Secure Stream Cipher Initialisation Processes

by

Ali Abdulaziz Alhamdan

Bachelor of Engineering (Electrical Engineering) *(KSU)* – 1997
Master of Information Technology *(QUT)* – 2009

Thesis submitted in accordance with the regulations for the Degree of Doctor of Philosophy

Information Security discipline
Electrical Engineering and Computer Science School
Science and Engineering Faculty
Queensland University of Technology

January 2014

Keywords

A5/1, Common Scrambling Algorithm Stream Cipher (CSA-SC), cryptanalysis, cryptography, differential attacks, diffusion phase, Dragon, Grain, initialisation flaws, initialisation process, initialisation vector (IV), LILI, loading phase, MICKEY, RC4, rekeying, resynchronisation attacks, Sfinks, shifted keystream, slid pairs, slide attacks, state convergence, stream cipher, symmetric cipher, Time Memory trade-off (TMTO) attacks, Trivium, weak key-IV combinations.

Abstract

Stream ciphers are symmetric key cryptosystems that are used commonly to provide confidentiality for a wide range of frame-based applications; such as mobile phone communication, pay TV transmission and Internet traffic. For these applications, stream ciphers are preferred for encryption due to the simplicity of implementation, security against known attacks, efficiency and high throughput.

In modern stream ciphers, the initialisation process (also known as resynchronisation) involves the use of publicly known material as well as the secret key. Unless it is carefully designed, this process may reveal some information about the secret key or may leave a stream cipher vulnerable for some attacks. Analysis of the initialisation process in the literature is limited and has not been addressed thoroughly.

The main objective of this research is to provide design recommendations for strengthening initialisation processes in modern stream ciphers. We achieve this by examining in-depth the initialisation process of three well-known stream ciphers: A5/1, Sfinks and the Common Scrambling Algorithm. Our reasons for choosing these algorithms are:

- These ciphers are broadly representative of modern stream ciphers.
- They cover a variety of loading processes
- A5/1 and the Common Scrambling Algorithm are both used widely.

We have examined the initialisation processes of these three ciphers

Design criteria provide to prevent these flaws as well as other flaws in the initialisation process of stream ciphers.

Contents

List of Tables

List of Figures

List of Algorithms

List of Abbreviations

ATSC	Advanced Television Systems Committee
BSW	Biryukov, Shamir and Wagner sampling
CSA-SC	Common Scrambling Algorithm - Stream Cipher
DES	Data Encryption Standard
DVB	Digital Video Broadcasting
DVB-CSA	Digital Video Broadcasting Common Scrambling Algorithm
ETSI	European Telecommunication Standards Institute
FSM	Finite State Machine
GPRS	General packet radio service
GSM	Global System for Mobile Communications (cellular phone technology)
ICT	Information and Communications Technologies
ITU	International Telecommunication Union
IV	Initial Value / Initialisation Vector
K	The secret key
KSA	Key Scheduling Algorithm
LFSR	Linear Feedback Shift Register
MPEG-2	Motion Picture Experts Group 2
NFSR	Nonlinear Feedback Shift Register
OTP	One-time pad
PRGA	Pseudo-Random Generation Algorithm
SSL	Secure Sockets Layer
TMTO	Time Memory Trade-Off
WEP	Wired Equivalent Privacy

List of Symbols

$x\&y$	AND Bitwise; x AND y
$x \oplus y$	XOR Bitwise; x XOR y
$x\|y$	OR Bitwise; x OR y
$x \boxplus y$	modular addition operation, e.g for Dragon (mod 2^{32}), and for CSA-SC (mod 2^4)
$x + y$	Normal addition operation
$x \cdot y$	Normal product operation
$x\\|y$	The concatenation of the binary strings x and y
$\bigoplus s_t^j$	XOR s_t, for j times
$\sum_i^n a_i$	The sum; $a_i + a_{i+1} + \ldots + a_n$
$\Pi_i^n a_i$	The product; $a_i \cdot a_{i+1} \ldots + a_n$
$y \gg x$	Shifting y to the right by x bits
$y \ll x$	Shifting y to the left by x bits
$y \ggg x$	Rotate y to the right by x bits
	$y \gg x \parallel y \ll (n - x)$; where n is the y length in bits
$y \lll x$	Rotate y to the left by x bits
	$y \ll x \parallel y \gg (n - x)$; where n is the y length in bits
$O(x)$	Big-O notation: worst case algorithm complexity
$x \bmod n$	The remainder when x is divided by n
S	The state of a shift register
s_t^i	The stage i at time t
$y_{j,t}$	The j-th word of a memory at time t
$y_{j,t}^i$	The i-th bit of $y_{j,t}$ word
$\mathbb{F}_n^m$	Galois Field of characteristic n with n^m elements
$x \in \mathbb{F}$	x in $\mathbb{F}$
z_t	The output keystream bit at time t
c_t	The output ciphertext bit/word at time t
$\log x$	Logarithm base 10 of a real $x > 0$
$\log_a x$	Logarithm base a of a real $x > 0$
k_i	The i^{th} key bit
v_i	The i^{th} IV bit

Declaration

The work contained in this thesis has not been previously submitted for a degree or diploma at any higher education institution. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made.

QUT Verified Signature

Signed: **Date:** 24/1/2014

Previously Published Material

The following papers have been published, and contain material based on the content of this thesis.

[1] Sui-Guan Teo, Ali Alhamdan, Harry Bartlett, Leonie Simpson, Kenneth Koon-Ho Wong and Ed Dawson. State convergence in the initialisation of stream ciphers. In Udaya Parampalli and Philip Hawkes, editors, *In Proceedings 16th Australasian Conference Information Security and Privacy, (ACISP 2011), Melbourne, Australia*, volume 6812 of *Lecture Notes in Computer Science (LNCS)*, pages 75-88. Springer, 2011.

[2] Ali Alhamdan, Harry Bartlett, Leonie Simpson, Ed Dawson, and Kenneth Koon-Ho Wong. State convergence in the initialisation of the Sfinks stream cipher. In Josef Pieprzyk and Clark Thomborson, editors, *In Proceedings 10th Australasian Information Security Conference (AISC 2012)*, Melbourne, Australia, volume 125 of Conference in Research and Practice in Information Technology (CRPIT), pages 27-32. Australian Computer Society, 2012.

[3] Ali Alhamdan, Harry Bartlett, Ed Dawson, Leonie Simpson and Kenneth Koon-Ho Wong. Slide Attacks on the Sfinks Stream Cipher. *In Proceedings of the 6th International Conference on Signal Processing and Communication Systems*, IEEE, Radisson Resort, Gold Coast, QLD, Dec. 2012.

[4] Ali Alhamdan, Harry Bartlett, Ed Dawson, Leonie Simpson and Kenneth Koon-Ho Wong. Slid pairs in the initialisation of the A5/1 stream cipher. In Clark Thomborson and Udaya Parampalli, editors, *In Proceedings of the 11th Australasian Information Security Conference (AISC 2013), Adelaide, Australian*, volume 138 of Conference in Research and Practice in Information Technology (CRPIT), pages 3-12. Australian Computer Society, 2013.

[5] Ali Alhamdan, Harry Bartlett, Ed Dawson, Leonie Simpson and Kenneth Koon-Ho Wong. Weak Key-IV Pairs in the A5/1 Stream Cipher. In Udaya Parampalli and Ian Welch, editors, *In Proceedings of the Australasian Information Security Conference (AISC 2014), Auckland, New Zealand*, volume 149 of Conference in Research and Practice in Information Technology (CRPIT), Australian Computer Society, to appear, 2014.

The following papers has been presented, and contain material based on the content of this thesis.

[6] Sui-Guan Teo, Harry Bartlett, Ali Alhamdan, Leonie Simpson, Kenneth Koon-Ho Wong and Ed Dawson. State convergence in bit-based stream ciphers. Cryptology ePrint Archive, Report 2013/096, 2013. http://eprint.iacr.org/.

[7] Ali Alhamdan and Harry Bartlett. State Space convergence in the A5/1 keystream generator. Presented at the rump session of ASIACRYPT 2010.

Acknowledgements

In the name of Allah, the most Generous and the most Merciful. All praise is due to Allah, for giving me inspiration and guidance along this journey. I have neither power nor ability to do anything without his blessings. My thanks to Allah is not complete if I do not thank all the people involved directly or indirectly with this research project.

Working towards a PhD is an arduous journey, but the guidance and encouragement of my supervisors are paramount. In this time, I wish to express my sincere gratitude, appreciation and acknowledgement to my principal supervisor Dr. Leonie Simpson and to my associate supervisors, Emeritus Professor Ed Dawson, Dr. Harry Bartlett and Dr.Kenneth Koon-Ho Wong. Without their enthusiastic support I would not have started or finished. I can not forget their constructive comments, encouragement and support to my work. I really enjoy working with them during my PhD research project.

I wish also acknowledge all of the academic and administrative staff of the Information Security Institute (ISI) for their support and encouragement. This institute has moved into "Information Security discipline" under the Institute for Future Environments in the Queensland University of Technology. Also, I wish to thank the administrative staff of the Electrical Engineering and Computer Science school (EECS). I would like to acknowledge my friends and colleagues, I would not name individual names here. Everyone contributed to me as person or to my research project.

I would like to acknowledge a fellow PhD student, Sui-Guan Teo for a joint paper with. We have published a paper "State convergence in the initialisation of stream ciphers". Two stream ciphers are analysed in this paper as follows: A5/1 and Mixer ciphers. A5/1 cipher is analysed by Ali and the Mixer is analysed by Sui-Guan.

Some experiments and results in this thesis require the use of the Computational Algebra System, MAGMA. Most of the results in this thesis would not have been possible without the computational facilities provided by the High Performance Computing (HPC) and Research Support Group, Queensland University of Technology, Brisbane, Australia. So, I would like like to extend my gratitude to the HPC staff for their efforts.

I would like to thank the two independent members of my internal examination panel: Dr. Joanne Hall and Dr. Seyit Camtepe, for their comments which helped to improve the quality of this thesis.

I must acknowledge the National Information Center at the Ministry of Interior in Saudi Arabia for granting me the scholarship and providing me the opportunity to do my PhD research. Without their financial help this research would not have been possible.

There is no words that can express my deepest gratitude to my parents. I am deeply indebted to them. Without their encouragement and prayers, I could not complete this long research journey. I spent several years away from them, while they need a special care and attention in their old age. Also, I wish to thank all of my brothers, sisters and all of my relatives. As well, I am greatly indebted to my late father in-law, mother in-law, brothers in-law and sisters in-law for their support and help.

This thesis would not have completed without the love and support of my wife, my lovely daughters and my little son. They spent many lonely hours and weekends as I endeavoured to complete my PhD. My warmest, heartfelt and deepest gratitude for them for their patience and understanding. I understand the difficulties that they faced during my PhD journey.

Chapter 1

Introduction

Stream ciphers are used in a wide range of real-time applications; such as the internet, pay TV and mobile phone transmissions; in order to provide confidentiality. That is, the information should not be accessible to unauthorised parties. However, before the cipher can be used to protect information, it must be initialised using secret key material and sometimes other publicly known material commonly referred to as an initialisation vector (IV) .

For some applications, the communication is divided into packets or frames, and initialisation is performed for each section of the communication. Generally the same secret key is used for the whole communication but with a different IV for each packet or frame. After initialisation is completed, a keystream is generated and used for encryption.

Stream ciphers are considered as an important class of encryption algorithms for several reasons. Stream ciphers in general are faster than block ciphers. They also have less complex hardware implementation. Stream ciphers have limited error propagation, and in some applications there is no error propagation [83, p. 191]. The security provided by stream ciphers depends on both the initialisation and keystream generation processes being secure.

Most cryptanalysts focus on the keystream generation phase, rather than considering both the initialisation and keystream generation phases. However, the security of the initialisation process is also important. The initialisation process must ensure that the keystreams produced using the same secret key but different IVs appear unrelated. A good initialisation process should ensure that each key-IV pair generates a distinct and unpredictable keystream. This is possible for recent proposals where the state size of the keystream generator is large enough. Also, the initialisation process (loading and diffusion phases) should not reveal any information about the secret key. The initialisation process should ensure that the key recovery attack is hard and the mathematical relationships between the key-IV pair and the keystreams are hard to establish as well.

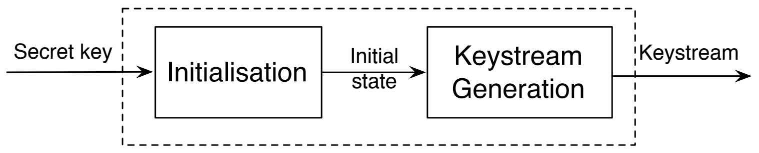
This research aims to determine whether the initialisation processes are secure at least against the known attacks. It focuses on the initialisation processes as a necessary step before keystream generation. The security of the initialisation processes is investigated and examined for different existing stream cipher proposals. Initialisation processes are examined against generic attacks to identify the features which contribute to successful attacks. This investigation covers the strengths and weaknesses of the initialisation processes to identify the flaws that may occur.

Efficiency of initialisation processes is another important point for some applications. For example, real-time applications such as mobile phones require efficient initialisation processes because rekeying is performed for each portion of data. For example, for mobile phone systems (such as GSM) , the rekeying process is performed every 4.6 milliseconds (using the same secret key and different IV) [24]. Efficient initialisation processes allow stream ciphers to perform fast rekeying using the secret key and multiple IVs to produce multiple keystreams without any noticeable delay between parties. In general, efficiency may involve energy consumption, area (footprint) or required memory and space, although these are beyond the scope of this research. In this research, we comment on the efficiency as a secondary concern and only related to the time required to complete the process.

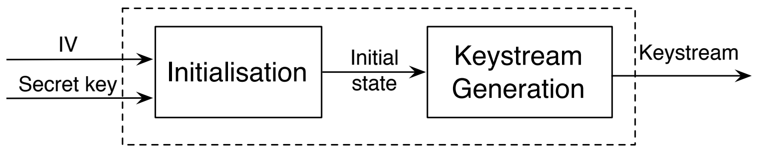
1.1 Motivation

Communications between parties through public channels require security. Global System for Mobile communications (GSM) , Internet and Pay TV, for example, all may require confidentiality. In many cases, this is provided by encryption using synchronous stream ciphers. A problem arises when the parties lose synchronisation. So in this case, the sender is required to resend the lost message. Therefore, the communication systems require techniques to maintain synchronisation. To address this problem efficiently, the requirement for reinitialisation (rekeying) appears to be especially important for real-time applications.

Prior to 2000 most stream cipher proposals utilised only a secret key to generate the keystream sequences. This is the case, for example, for the widely used RC4 stream cipher [91, p. 397]. Around that time, the problems of loss of synchronisation and the generation of whole keystream using only the secret key alone (vulnerable to TMTO) were identified with this practice of using secret key alone. Therefore, the initialisation process and the use of an initialisation vector (IV) was introduced to solve the practical problem of resynchronisation, allowing the generation of multiple keystream sequences from the same secret key with multiple IVs. Figure 1.1a illustrates the general models of keystream generators that use only a secret key to generate a keystream, and Figure 1.1b shows a keystream generator that uses both the secret key and an IV to generate a keystream.



(a) Using only secret key as input



(b) Using secret key and IV as inputs

Figure 1.1: General structure of keystream generator

The requirement to use both a secret key and IV is comparatively recent, and the security analysis of initialisation processes is less thoroughly addressed than that of the keystream generation process but it is no less important. The New European Schemes for Signatures, Integrity, and Encryption (NESSIE) project (between 2000 and 2003) [68] marks a time point for stream cipher design, where the use of an IV with the secret key became essential. The original call for primitives did not require the use of an IV with the stream cipher algorithms. However, requests to add rekeying proposals to submissions were made during the project. Since then, rekeying schemes have become mandatory for new stream cipher proposals. For example, the eSTREAM project (between 2004 to 2008) [50] sponsored by the European Network of Excellence for Cryptology

(ECRYPT) required that the initialisation vector (IV) be used and an initialisation process specified as an essential part of the submissions.

To date, there is a limited number of studies on the resynchronisation problem of the initialisation processes of stream ciphers. Resynchronisation attacks are applied for some stream ciphers: for Grain stream cipher in [10, 45, 73, 76, 108], and for Trivium stream cipher in [63, 64, 86]. Also, Hong and Kim [65] reported entropy loss in the internal state of MICKEY stream cipher after a number of iterations of the initialisation state update function.

1.2 Aims and Objectives

The general aim of this research project is to investigate the initialisation processes of stream ciphers, primarily in terms of security. The research considers both the *loading phase* (loading of secret key and IV) and the *diffusion phase* of the initialisation process. In more detail, the aims of this research project are as follows:

1. To identify common features and properties in the initialisation processes of the keystream generators for stream ciphers during the loading and diffusion phases.

2. To examine and determine the impact of these features on the security provided by performing an in-depth analysis of the initialisation processes of three stream ciphers to identify any potential flaws.

3. To propose design criteria for the initialisation process of stream ciphers to avoid the identified weaknesses.

1.3 Contributions and Achievements

The contributions that have been achieved in this thesis are as follows:

1.3.1 Analysis of the Initialisation Process of A5/1 Stream Cipher

In this contribution, the initialisation process of A5/1 stream cipher is analysed. Three flaws are identified: state convergence, the existence of slid pairs and weak key-IVs. This contribution helps to achieve aims 1 and 2 from Section 1.2.

State Convergence in A5/1 Stream Cipher

In prior work, Golić [54, 55] demonstrated the proportional reduction of the internal state space after one clock. In our research, we extended Golić's work by examining the reduction in the state space for further five iterations. We extrapolated to estimate the state space reduction after 100 iterations. The proportion of internal states which can be obtained is reduced after six iterations by approximately half.

This contribution appears in the following publication:

- Sui-Guan Teo, Ali Alhamdan, Harry Bartlett, Leonie Simpson, Kenneth Koon-Ho Wong and Ed Dawson. State convergence in the initialisation of stream ciphers. In Udaya Parampalli and Philip Hawkes, editors, *Proceedings 16th Australasian Conference Information Security and Privacy, (ACISP 2011), Melbourne, Australia*, volume 6812 of *Lecture Notes in Computer Science (LNCS)*, pages 75-88. Springer, 2011.

Slid Pairs in A5/1 Stream Cipher

The A5/1 stream cipher uses the same update function during both the initialisation and keystream generation processes. Additionally, each internal state obtained after any number of iterations, is a legitimate loaded state. This is used to show that slid pairs and shifted keystream can be obtained easily, even after only one or two clocks. An attack procedure and the attacking algorithm are presented using slid pairs and shifted keystream flaws for the A5/1 cipher. This attack targets secret key recovery using a ciphertext-only attack model.

This contribution appears in the following publication:

- Ali Alhamdan, Harry Bartlett, Ed Dawson, Leonie Simpson and Kenneth Koon-Ho Wong. Slid pairs in the initialisation of the A5/1 stream cipher. In Clark Thomborson and Udaya Parampalli, editors, *Proceedings of the 11th Australasian Information Security Conference (AISC 2013), Adelaide, Australia*, volume 138 of Conference in Research and Practice in Information Technology (CRPIT), pages 3-12. Australian Computer Society, 2013.

Weak Key-IV in A5/1 Stream Cipher

It is shown that the non-autonomous feedback mechanism during the key-IV loading phase may result in the contents of one, two or three registers containing all zeros. When two or three registers contain all-zeros, it is possible to perform key recovery using ciphertext-only attack.

This contribution will appear in the following publication:

- Ali Alhamdan, Harry Bartlett, Ed Dawson, Leonie Simpson and Kenneth Koon-Ho Wong. Weak key-IV Pairs in the A5/1 Stream Cipher. In Udaya Parampalli and Ian Welch, editors, *Proceedings of the Australasian Information Security Conference (AISC 2014), Auckland, New Zealand*, volume 149 of Conference in Research and Practice in Information Technology (CRPIT). Australian Computer Society, to appear, 2014.

1.3.2 Analysis of the Initialisation Process of Sfinks Stream Cipher

In this contribution, the initialisation process of the Sfinks stream cipher is analysed. Two flaws are identified: state convergence and the existence of slid pairs. This contribution helps to achieve aims 1 and 2 from Section 1.2.

State Convergence in Sfinks Stream Cipher

This contribution demonstrates that, even though the individual components of the state update function are one-to-one, the combination and interaction between these components may not be one-to-one and therefore state convergence can still occur.

This contribution appears in the following publication:

- Ali Alhamdan, Harry Bartlett, Leonie Simpson, Ed Dawson, and Kenneth Koon-Ho Wong. State convergence in the initialisation of the Sfinks stream cipher. In Josef Pieprzyk and Clark Thomborson, editors, *Proceedings 10th Australasian Information Security Conference (AISC 2012), Melbourne, Australia*, volume 125 of Conference in Research and Practice in Information Technology (CRPIT), pages 27-32. Australian Computer Society, 2012.

Slid Pairs in Sfinks Stream Cipher

The occurrence of slid pairs and shifted keystreams is investigated for the Sfinks stream cipher. Due to the padding pattern and the specification of the state update function, the occurrence of slid pairs is deferred until the 17th iteration. We demonstrate a modification to the padding pattern which is based on changing the content of just one stage. This modified version dramatically changes the time interval before slid pairs are obtained, and the probability of slid pairs and shifted keystreams.

The contribution appears in the following publication:

- Ali Alhamdan, Harry Bartlett, Ed Dawson, Leonie Simpson and Kenneth Koon-Ho Wong. Slide Attacks on the Sfinks Stream Cipher. In *Proceedings of the 6th International Conference on Signal Processing and Communication Systems*, IEEE, Radisson Resort, Gold Coast, QLD, Dec. 2012.

1.3.3 Analysis of the Initialisation Process of the CSA-SC

This contribution is obtained from analysis of the initialisation process of the Common Scrambling Algorithm Stream Cipher (CSA-SC). Two flaws have been identified in the CSA-SC: state convergence and the existence of slid pairs. This contribution helps to achieve aims 1 and 2 from Section 1.2.

State Convergence in CSA-SC

The state update function during the initialisation process is not one-to-one, because the registers are not updated autonomously. This leads to state convergence during the majority of the initialisation process.

Slid Pairs in CSA-SC

The occurrence of slid pairs and shifted keystreams in the CSA-SC is demonstrated. This occurs due to the padding pattern and the state update functions of both the initialisation and keystream generation processes. The highest probability to obtain slid pairs and shifted keystreams occurs after four clocking steps, which is $2^{-43.65}$.

1.3.4 Criteria for the Initialisation Process of Stream Ciphers

This contribution provides recommendations for designing the initialisation process of shift-register based stream ciphers so as to achieve secure and efficient stream ciphers. These recommendations are based on an in-depth analysis of the initialisation processes of such stream ciphers, taking into account our own contributions (as described in Sections 1.3.1, 1.3.2 and 1.3.3) as well as existing results in the public literature.

In particular, our analysis identifies features and properties of the loading and diffusion phases that can cause the flaws identified in our other contributions and in the existing literature, while the recommendations provide guidelines for avoiding these flaws and reducing the severity of their effects. In all, we provide eight recommendations for the design of the stream cipher initialisation processes. This contribution helps to achieve aims 2 and 3 from Section 1.2.

1.4 Thesis Outline

The organisation of this thesis is to present the literature review and then present the results from analysing the initialisation processes of three different ciphers: A5/1, Sfinks and CSA-SC. Each cipher is described and the initialisation process analysed in a separate chapter. Following this, a discussion of common flaws and possible mitigation strategies is provided. In more detail, this thesis is organised as follows:

Chapter 2. This chapter reviews the literature and background information that is the basis of this research. It introduces the terminology and notation used in this thesis. A general overview of stream ciphers and more detail on the initialisation processes are provided. After that, examples of initialisation processes from commonly used ciphers are discussed. Cryptanalysis of stream ciphers, generic attacks, flaws and existing attacks on the initialisation process are also discussed. It gives the necessary background for an understanding of the stream cipher initialisation processes.

Chapter 3. This chapter describes the specification of A5/1 stream cipher and the initialisation process thoroughly, and reviews the previous analysis on the initialisation process of this cipher. It investigates three flaws in the initialisation process of A5/1 as follows: state convergence, the existence of slid pairs and synchronisation attacks, and key-IV weaknesses. This chapter represents the first major contribution by analysing the initialisation process of the A5/1 stream cipher.

Chapter 4. This chapter describes the specification of the Sfinks stream cipher and the initialisation process thoroughly. It investigates two flaws in the initialisation process of Sfinks as follows: state convergence and the existence of slid pairs and synchronisation attacks. This chapter represents the second major contribution by analysing the initialisation process of Sfinks stream cipher.

Chapter 5. This chapter describes the specification of the Common Scrambling Algorithm Stream Cipher (CSA-SC) and the initialisation process thoroughly. It investigates two flaws in the initialisation process of CSA-SC as follows: state convergence and the existence of slid pairs and synchronisation attacks. This chapter is the third major contribution which analyses the initialisation process of CSA-SC stream cipher.

Chapter 6. This chapter provides an in-depth discussion of flaws and attacks on the initialisation process of stream ciphers. This discussion is based on the literature analyses in Chapter 2 and the investigations presented in Chapters 3, 4 and 5. This chapter discusses the causes of these flaws in terms of design choices in the loading and diffusion phases, or the combination of these phases. It also reviews briefly the weaknesses identified in initialisation process in the existing literature. Finally, it gives some recommendations for the initialisation processes to overcome or mitigate these flaws.

Chapter 7. This chapter concludes the research and gives some directions for future work in this area.

Chapter 2

Background

This chapter presents an overview of the topics relevant to secure stream cipher initialisation processes, including a review of the literature of stream ciphers, concentrating on the initialisation processes. This review provides the necessary information to support the research conducted in the thesis. It gives a general view of stream ciphers and their initialisation (rekeying) process. Also, it reviews the generic methods of attack that are used against stream ciphers and which are also useful to attack the initialisation process. Finally, several examples of initialisation processes from well known stream ciphers are provided, together with some existing analysis of their weaknesses.

Section 2.1 gives a general overview of the terminology that is used in this thesis. Following this, an overview of stream ciphers, types of stream ciphers and properties of keystream sequences are presented in Section 2.2. Section 2.3 discusses the phases of the initialisation process and gives a brief overview of the keystream generation process. Section 2.4 provides some examples of initialisation processes of stream ciphers. Section 2.5 gives an introduction to cryptanalysis of stream ciphers. Some generic attacks that are used for stream ciphers and applied to the initialisation process are discussed in Section 2.6. Section 2.7 discusses three generic flaws in the initialisation processes. Some existing attacks on the initialisation processes of ciphers from Section 2.4 are shown in Section 2.8. Section 2.10 presents a summary of this chapter.

2.1 Terminology and Notation

The following terminology and notation are used in this research project:

- *Plaintext:* The original intelligible data, such as a conversation or any data.
- *Keystream:* The sequence of random or pseudorandom bits that are generated by a keystream generator using a secret key and (optionally) an IV.
- *Ciphertext:* The output of the encryption algorithm for a given plaintext. For most stream ciphers, it is the combination of a plaintext and a keystream using the bitwise "XOR" operation. It is unintelligible information.
- *Encryption algorithm:* The procedure or method used to convert a plaintext to ciphertext using a secret key and (optionally) an IV.
- *Decryption algorithm:* The inverse of the encryption algorithm. It is the procedure or method used to convert a ciphertext to a plaintext using a secret key and (optionally) an IV.
- *Internal state:* Information stored within the components of a stream cipher, this information is updated during the operation of the cipher and is used to generate the keystream.
- *State size:* The total number of bits or words that can be held by a stream cipher component.
- *State space:* The total number of states that can be held by the components of the cipher is 2^s, where s is the state size (length).
- *Initialisation vector (IV):* (or frame number). Publicly known information which is used with the secret key (master key) to generate the session key that is used in turn to generate the keystream.
- *IV space:* The set of all possible IVs that can be generated from a specific number of bits (length of IV). If j is the length of IV, the IV space is 2^j.
- *Cryptographic keys:*

    - *Secret key (Master key):* An input to the encryption and decryption algorithms which is combined with the known IV to produce a session key. It is assumed that this input is known only to the sender and receiver.
    - *Initial state (Session key):* The internal state immediately following the initialisation process which is then used by the keystream generator to generate the keystream sequences.

- *Key Space (secret key space):* The set of all possible secret keys that can be generated from a specific number of bits (length of secret key). If the secret key length is l, then the key space is 2^l.
- *State update function:* The function that is used to update the content of the internal state at each clock.
- *Loading phase:* The process of loading the secret key and sometimes the IV also into the internal state of a stream cipher, resulting in a *loaded state* . This is the first phase of the initialisation process.
- *Diffusion phase:* The process of diffusing, mixing and expanding the secret key and IV over the internal state of a stream cipher. This is the second phase of the initialisation process. Eventually, the diffusion phase produces the *initial state* (session key). After that the keystream generation process can begin.

2.2 Stream Ciphers

Stream ciphers are a security mechanism used in a wide range of real-time applications to provide confidentiality. Stream ciphers are symmetric key ciphers. This means that the same secret key, IV and algorithm used for encryption must also be used to perform decryption.

Stream ciphers are defined by Rueppel [89, p. 1] “stream ciphers divide the plaintext into characters and encipher each character with a time-varying function whose time-dependency is governed by the internal state of stream cipher”. Stream ciphers are defined by Menezes et al [83, p. 191] as “stream ciphers encrypt individual characters (usually binary digits) of a plaintext message one at a time using an encryption transformation which varies with time”. Stallings [97, p.

189] reported that stream ciphers may be designed to operate on one bit at a time or in units larger than a byte at a time.

Stream ciphers can be classified as bit-based or word-based depending on the operation and output size of the key stream generator. The bit-based stream ciphers process a single bit per clock. Some stream ciphers are bit-based but can be modified to produce several keystream bits per clock, such as Grain stream cipher [59]. (It seems to be word-based in the output function.) Bit-based stream ciphers are the earliest design and implementation of stream ciphers [48] and many bit-based stream ciphers are based on shift registers [54]. Bit-based stream ciphers are more suitable for hardware implementation. Examples of bit-based stream ciphers are A5/1 [54], Trivium [43], LEX [22] and Grain [59]. Word-based stream ciphers are designed for faster software implementation as they can manipulate words per clock. Bit-based stream ciphers may be slower in software implementation [32]. The word-based stream ciphers produce an n-bit word at a time where the word size varies based on the structure of ciphers. This type of stream cipher is efficient for software implementation and provides a possible solution to the security efficiency trade-off [32]. There are several well-known algorithms that are word-based stream ciphers; for example, RC4 [91] commonly uses a word size of 8 bits, Turing [88] uses word size of 32 bits and the Dragon stream cipher [32] uses 32-bit words in the operational stages and generates 2 output words (64 bits) per clock. In software applications, the clock frequency is generally constant for a given processor (related to the clock rate) and the throughput is increased by increasing the word size.

The most common type of stream cipher is the binary additive stream cipher, which uses the "XOR" operation to combine the plaintext m_i and the keystream z_i to produce the ciphertext c_i; and vice versa, by combining the ciphertext with the keystream to recover the plaintext; as shown in Figure 2.1. This combining feature has two major advantages: it is very fast and the same algorithm can be used for both encryption and decryption without any modification. The encryption and decryption equations are as follows.

$$\text{For encryption,} \qquad c_i = z_i \oplus m_i, \quad 1 \leq i \tag{2.1a}$$

$$\text{For decryption,} \qquad m_i = z_i \oplus c_i, \quad 1 \leq i \tag{2.1b}$$

where: z_i is the individual keystream bit, m_i: the individual plaintext bit, c_i: the individual ciphertext bit, $\oplus$ is the XOR Boolean operation (addition (mod 2)) and i is the time index.

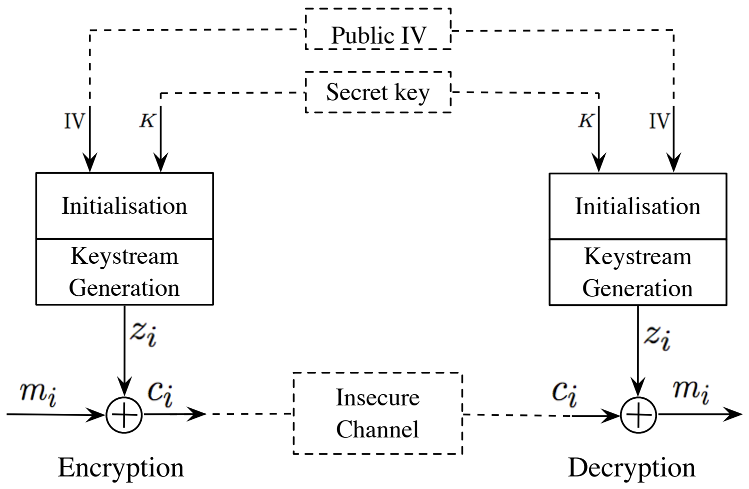
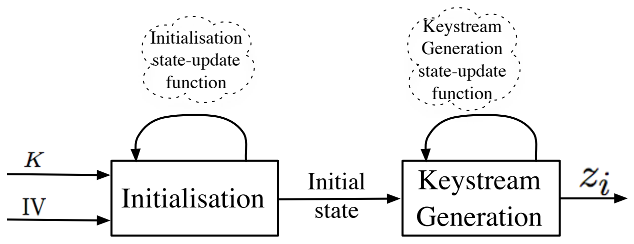


Figure 2.1: Binary additive stream cipher

Most recent stream ciphers consist of two main processes: initialisation process and keystream generation process. Figure 2.2 shows a general construction of recent stream ciphers using initialisation and keystream generation processes.



Initialisation state-update function
Keystream Generation state-update function
K
IV
Initialisation
Initial state
Keystream Generation
z_i

Figure 2.2: Stream cipher with initialisation and keystream generation processes

Most stream ciphers use a keystream generator to produce pseudorandom keystream sequences. The security provided by the cipher depends on these keystream sequences appearing to be random [20, 24]. Most modern keystream generators utilise two inputs: a secret key and a sequence of initialisation vectors (IVs) or frame numbers [67]. The IVs are assumed to be known information. The keystream generator loads the secret key and IV and performs some initialisation process in order to form the session key (initial state of the keystream generator) before any keystream can be produced. Suppose a secret key, K, and IV are used to generate a keystream z. We can represent z as $z = \{z_i\}_1^n = f(K, \text{IV})$, where n is the number of keystream bits and f is the cipher function.

2.2.1 Types of Keystream Generators

Keystream generators are commonly classified into synchronous and self-synchronous keystream generators. This classification is related to the dependence of the keystream sequences on the previous ciphertext as explained below.

A - Synchronous Stream Ciphers

Synchronous stream ciphers generate the keystream solely from the internal state (formed from the secret key and IV) and an update function which are independent of the plaintext or ciphertext [83, p. 193]. Encryption and decryption of synchronous stream ciphers are described by the following equations and illustrated in Figure 2.3.

$$\begin{array}{ll} \text{Keystream output} & \{z_i\}_1^n = f(K, \text{IV}) \\ \text{Ciphertext} & c_i = z_i \oplus m_i \end{array}$$

where m_i is plaintext bit, c_i is ciphertext bit, z_i is keystream bit, k is secret key, v is initialisation vector, f is the cipher function and n is the number of keystream bits.

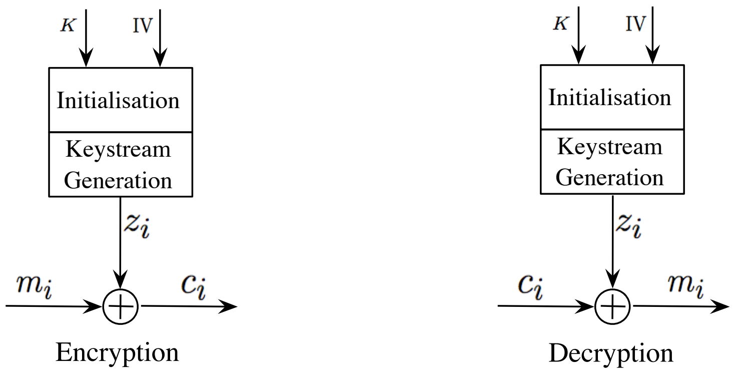


Figure 2.3: Encryption and decryption of synchronous stream cipher

A major advantage of synchronous stream ciphers is that there is no error propagation during the decryption processes, i.e. any bit/word errors that may occur in the ciphertext during transmission affect only the corresponding bit/word of the plaintext [37]. However, synchronous stream ciphers require both the sender and receiver to be synchronised to allow the message to be intelligible to the author and receivers; that is the keystream generator used by both encryptor and decryptor have simultaneously the same internal state during the encryption and decryption. If synchronisation is lost, the decryption processes will fail and need an additional technique for resynchronisation. For example, if bits/words are inserted or deleted from a ciphertext, then the decryption will fail and the synchronisation is lost [83].

Frequent Rekeying (Re-initialisation)

Rekeying is the process of using the secret key and multiple IVs to generate multiple keystream sequences. (That is to avoid using same keystream sequence with multiple messages.) Rekeying or re-initialisation process is the method that is used to solve the problem of synchronisation of the synchronous stream ciphers, to diffuse the secret key across the entire internal state and to generate multiple keystream sequences from one secret key associated with multiple IVs. It is used to avoid the requirement for a resynchronisation technique for the synchronous stream cipher. If the synchronisation is lost and the interval is small, then only that small amount of information before rekeying will be lost. This should not cause any significant degradation in the recovery of the original message. For example, A5/1 performs rekeying for a GSM conversation every 4.6 milliseconds [24]. This high performance rekeying stream cipher is competitive in practical applications and is preferred for real-time applications, such as mobile and wireless communications [32]. Rekeying is affected by the transmission packet size. For example, the packet size of Digital Video Broadcasting (DVB) is 184 bytes, Advanced Television Systems Committee (ATSC) is 208 bytes, General Packet Radio Service (GPRS) is 160, 240, 288 or 400 bits and GSM mobile phone is 228 bits. Figure 2.4 shows the general concept of rekeying for a keystream sequence using the same secret key, K, and multiple IVs.

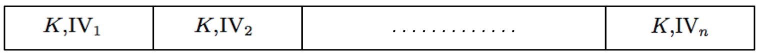


Figure 2.4: A keystream generated using the same secret key, K, and multiple IVs

Frequent rekeying process (re-initialisation) of stream ciphers must operate with the transmission speed between parties (sender and the receivers). Therefore, frequent rekeying of a system requires efficient initialisation process, especially for real-time applications. If the rekeying process is slow, then the stream cipher cannot encrypt or decrypt all the transmitted messages properly. However, the requirement of efficiency should not compromise the security of the initialisation process.

B - Self-Synchronous Stream Ciphers

Self-synchronous (or asynchronous) stream ciphers generate keystream as a function of the secret key, IV and a number of bits from the previous ciphertext [91, p. 198]. Figure 2.5 shows the general concept of self-synchronous stream ciphers. The self-synchronous stream cipher encryption function can be described as a function of the secret key, IV and the feedback of the ciphertext.

$$\begin{array}{ll} \text{Keystream output} & \{z_i\}_1^n = f((K, \text{IV}), \sigma_i) \\ \text{Ciphertext} & c_i = z_i \oplus m_i \end{array}$$

where, $\sigma_i = (c_{i-t}, c_{i-t+1}, \ldots, c_{i-1})$. From the above equations, it is obvious that the internal state is a function of the previous ciphertext, and the encryption processes depend on the previous ciphertext.

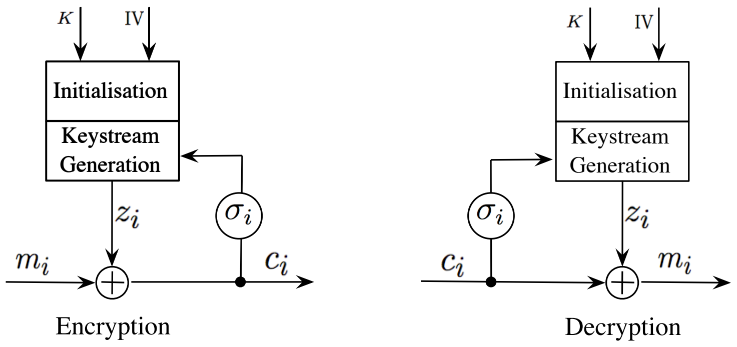


Figure 2.5: Encryption and decryption of self-synchronous stream cipher

Self-synchronous stream ciphers have the ability to automatically resynchronise themselves after a loss of synchronisation. That is, if ciphertext bits/words are deleted or inserted, the cipher loses the synchronisation; however, self-synchronous stream ciphers will automatically recover the synchronisation after several decryption steps. Self-synchronous stream ciphers propagate errors for a limited number of clocks. If a ciphertext bit is modified, then the decryption processes will affect a number of ciphertext sequences before synchronising itself. The drawback of the self-synchronous stream ciphers is that this type of cipher may be vulnerable to chosen ciphertext attack [107].

From the above argument, the characteristics of synchronous, self-synchronous stream ciphers and the requirement for frequent rekeying are highlighted. Based on the fact that the synchronous stream ciphers resist the chosen ciphertext attack, this thesis focuses on analysis of the initialisation process of the synchronous stream ciphers.

2.2.2 Properties of Keystream

Pseudorandom keystream sequences are intended to be indistinguishable from truly random sequences. However, as Schneier comments, "it is impossible to produce something truly random on a computer" [91, p. 44]. Therefore, standard properties of the keystream sequences are used to measure unpredictability of the pseudorandom keystream sequences. These include criteria for period, linear complexity and white noise statistics [41]. Meeting the relevant criteria for these properties is a necessary (but not sufficient) condition for generating pseudorandom keystream sequences. These properties and their effects on the security of keystream sequences are explained in the rest of this section.

- *Period.* A deterministic keystream generator produces pseudorandom sequences or sequences that eventually become periodic. The period of the keystream sequences should be longer than the length of the plaintext. If the keystream is periodic or repeated for another plaintext, it may be possible to recover the message by ciphertext-only attack [40].

- *Linear complexity.* Any periodic sequence can be constructed using a suitable chosen linear feedback shift register (LFSR). The linear complexity L of a sequence is the length of the shortest binary LFSR that generates the specified sequence. The details of the appropriate LFSR for a given sequence can be determined from $2L$ bits of known output sequence using the Berlekamp-Massey algorithm [14, 80]. Therefore, the linear complexity of the LFSR should be sufficiently large to avoid a reconstruction of the internal state [41].

- *White noise statistics.* The output sequences of random generators are statistically independent and unbiased [83]. A pseudorandom sequence generated by a keystream generator should mimic this behaviour. That is, it should be indistinguishable from a random sequence. In a pseudorandom sequence, occurrence of single, pairs, triples bits etc are uniformly distributed. The keystream should appear to be random in short and long subsequences. [41]. The output sequences should pass all the statistical tests of randomness [91]. There are five commonly used statistical tests to measure the security of pseudorandom keystream sequences as follows: (1) frequency test: the number of ones and zeros in any sequences are approximately the same; (2) serial test: (two-bit test), this test ensures that the number of occurrences of 00, 01, 10 and 11 are approximately the same; (3) poker test: if m is a positive integer and divides the keystream sequence Z into n non-overlapping parts of length m, the poker test will determine if the sequences of length m appear approximately the same number of times in each keystream sequence; (4) runs test: in a specific period, half of the runs should have a length of one (zero or one), a quarter of the runs should have a length of two (zeros, ones) and so on; and (5) autocorrelation test which checks the correlations within a sequence in a period p [83].

2.3 Initialisation Process

The initialisation process is a critical phase that must be performed before keystream generation can begin [60]. After that, the keystream generator can be used to produce keystream sequences. Figure 2.2 illustrates the sequencing of the initialisation and the keystream generation processes. For security, the initialisation process must not reveal any information about the secret key and must be secure against at least the known generic attacks. The initialisation process consists of two main phases, *loading* and *diffusion* and are described below.

2.3.1 Loading Phase

In most stream ciphers, in the *loading phase* the secret key and IV (if relevant) are loaded into the internal state to produce the *loaded state*. In some ciphers this is performed sequentially. For example the A5/1 stream cipher loads the secret key first, followed by the IV. Alternatively, some stream ciphers such as Grain and Trivium load both secret key and IV simultaneously into the internal state. The loading phase itself may use either a linear or nonlinear function. After the loading phase is performed, the diffusion phase is applied.

In some stream ciphers there is no specific format for the loaded state, such as the A5/1 stream cipher (so any internal state after a number of iterations is a legitimate loaded state). Some other ciphers have specific format (padding pattern), such as Trivium and Sfinks stream ciphers (so, any internal state after α iterations can be counted as a legitimate loaded state, only if it meets the loaded format). This padding system fills specific stages by predetermined values according to the cipher algorithm's design. The padding system should be considered in the security analysis.

2.3.2 Diffusion Phase

The *diffusion phase* consists of a specified number of iterations of the initialisation state update function to generate the *initial state* without producing any keystream bits. The objective is to diffuse the secret key and IV across the internal state. In some cases, the IV is loaded during the diffusion phase, such as the Common Scrambling Algorithm Stream Cipher [104]. The state update function during the diffusion phase is usually a nonlinear function. It may use Boolean functions or other nonlinear functions such as S-boxes. The purpose of this state update function is to make the derivation of the secret key from the initial state mathematically hard.

In general, the initialisation process should prevent secret key recovery from a known initial state (session key). In this case, if attackers have an ability to construct the initial state (session key) of a stream cipher, then the initialisation process should be sufficiently complicated to prevent the secret key recovery attack (except by exhaustive key search). If the key recovery attack is impossible, then the attackers must repeat their attack processes to find the session key each time when the cipher is rekeyed. Conversely, if an attacker is able to obtain the secret key, it can produce the keystream for any combination of the secret key and the known IVs. Therefore, initialisation processes are critical for stream ciphers.

2.3.3 Keystream Generation Process

Once the initialisation process is complete, the cipher is in its *initial state* that is the session key has been generated, and the keystream generation can begin. Keystream generation performs a specific number of clocks to generate a keystream sequence that should meet the required properties to be distinct and unpredictable. Keystream generation process consists of a state update function to update the content of the internal state and an output function to form the output keystream bits or words. The state update function of the keystream generation may be the same, different or have a degree of similarity to the state update function of the initialisation process (diffusion phase).

2.4 Examples of Initialisation Processes

This section reviews some stream ciphers that were submitted to the NESSIE [68] or eSTREAM [50] projects or that are well known for other reasons. This review gives a description of the ciphers, concentrating on the initialisation process. The initialisation processes of these stream ciphers have been analysed in the literature as presented in Section 2.8. We consider in this section the stream ciphers: RC4, Trivium, Grain, Dragon, MICKEY and LILI-II. As well, three stream ciphers A5/1 [30], Sfinks [29] and CSA-SC [15] will be presented in more detail in Chapters 3, 4 and 5 respectively. After we describe the other ciphers, we will include an explanation of our reasons for choosing these three ciphers.

2.4.1 RC4

Description

RC4 was designed in 1987 [97, p.191] by Ron Rivest for RSA Data Security, Inc. It is widely deployed in software applications such as secure sockets layer (SSL) due to its efficiency. RC4 is a word-based stream cipher with words of size n bits (mostly $n = 8$ bits) and a variable secret key-size. In the remaining of this section, we assume each word of RC4 is one byte in length (i.e. $n = 8$) as is commonly use. Let l denote the size of the secret key in bytes, commonly between 1 and 256 bytes. Let K denote the key and let each key byte be denoted as $k[i]$ for $i = 0, 1, \ldots, l-1$. The key is used indirectly to form the contents of an array, denoted S. The array S has size 2^n words. So the most common size of S is 256 bytes.

RC4 generates keystream sequences independently of the plaintext. The operation requires performing two phases: the key scheduling algorithm (KSA) and the pseudo random generator algorithm (PRGA). The KSA and PRGA are equivalent to the initialisation process for the internal state, S, and the keystream generation process (where each clock of the PRGA produces a keystream byte), respectively.

Key Scheduling Algorithm (KSA)

The initial process of RC4 is to fill the stages of array S with the integers from 0 to 255 in ascending order; that is $S[0] = 0, S[1] = 1, \ldots, S[255] = 255$. A temporary vector, T, of size 256 bytes is created and filled with the secret key. The secret key is transferred into the temporary vector T and may be repeated if it is necessary (if the key size is less than T size). Following this, a key dependent shuffling of S is performed. The secret key in the temporary array T is used to form the permutation of the S array only. Algorithm 2.1 illustrates the shuffling process of the key scheduling algorithm (KSA) in detail (which is applied on the array S).

```
 1: for $i = 0$ to 255 do
 2:     $S[i] = i$
 3:     $T[i] = k[i \bmod l]$
 4: end for
 5: $j = 0$
 6: for $i = 0$ to 255 do
 7:     $j = (j + S[i] + T[i]) \bmod 256$
 8:     Swap $(S[i], S[j])$
 9: end for
10: return $S$
```

Algorithm 2.1: RC4: Key Scheduling Algorithm (KSA)

Pseudo Random Generator Algorithm (PRGA)

After the key scheduling algorithm is complete, the pseudo random generator algorithm starts. The state update function is shuffling the bytes and then the output function selects a stage of S. The content of this stage is the output byte of the keystream as shown in Algorithm 2.2 (state update function is steps 4-6 and output function is steps 7 and 8). The keystream byte $S[u]$ is XORed with a plaintext byte using bit level XORing to produce a ciphertext byte.

```
Require: $S$
1: $i = 0$
2: $j = 0$
3: loop
4:     $i = (i + 1) \bmod 256$
5:     $j = (j + S[i]) \bmod 256$
6:     Swap $(S[i], S[j])$
7:     $u = (S[i] + S[j]) \bmod 256$
8:     print $S[u]$
9: end loop
```

Algorithm 2.2: RC4: keystream generation PRGA

2.4.2 Trivium

Description

The Trivium stream cipher was submitted to eSTREAM project, by De Cannière and Preneel [43]. Trivium has been selected to be in the final portfolio of the eSTREAM project. It is a hardware oriented bit-based synchronous stream cipher. It aims to provide flexible speed and space trade-off. Trivium takes an 80-bit secret key, K and 80-bit initial vector, IV, as input to a 288-bit shift register internal state as shown in Figure 2.6. It is designed to generate up to 2^{64} bits of keystream before the rekeying process. It consists of nonlinear feedback shift registers and a linear output function.

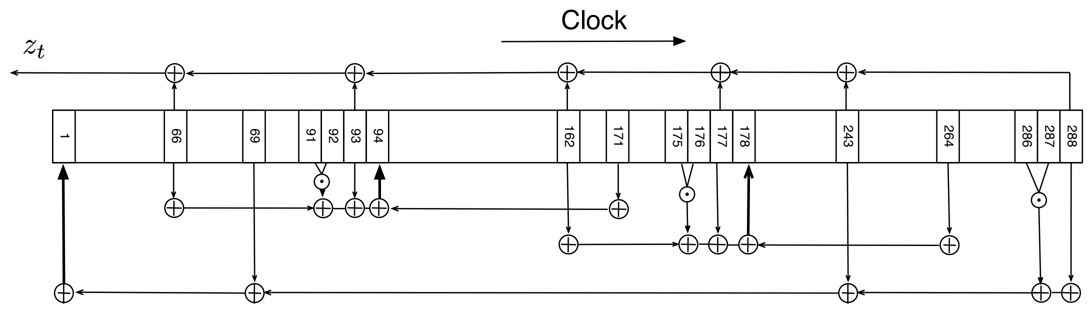


Figure 2.6: Trivium construction

Initialisation Process

During the loading phase, the 80-bit secret key and 80-bit IV are loaded into the 288-bit internal state in a linear manner and the remaining stages are set to zeros except the stages s_{286}, s_{287} and s_{288} which are set to all ones. Let the secret key bits be k_i, $1 \leq i \leq 80$, the IV bits be v_i, $1 \leq i \leq 80$ and the stages of internal state S be s_i, $1 \leq i \leq 288$. Algorithm 2.3 shows the loading and diffusion phases of Trivium. During the diffusion phase, the initialisation process performs 4 full cycles (4×288) before producing any keystream.

```
Require: $K$ and IV
 1: $(s_1, s_2, \ldots, s_{93}) \leftarrow (k_1, \ldots, k_{80}, 0, \ldots, 0)$
 2: $(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (v_1, \ldots, v_{80}, 0, \ldots, 0)$
 3: $(s_{178}, \ldots, s_{286}, s_{287}, s_{288}) \leftarrow (0, \ldots, 0, 1, 1, 1)$
 4: for $i = 1$ to $4 \cdot 288$ do
 5:     $t_1 \leftarrow s_{66} \oplus s_{91} \cdot s_{92} \oplus s_{93} \oplus s_{171}$
 6:     $t_2 \leftarrow s_{162} \oplus s_{175} \cdot s_{176} \oplus s_{177} \oplus s_{264}$
 7:     $t_3 \leftarrow s_{69} \oplus s_{243} \oplus s_{286} \cdot s_{287} \oplus s_{288}$
 8:     $(s_1, s_2, \ldots, s_{93}) \leftarrow (t_3, s_1, \ldots, s_{92})$
 9:     $(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (t_1, s_{94}, \ldots, s_{176})$
10:     $(s_{178}, s_{179}, \ldots, s_{288}) \leftarrow (t_2, s_{178}, \ldots, s_{287})$
11: end for
12: return $S$
```

Algorithm 2.3: Trivium initialisation process

Keystream Generation

The keystream generation uses 15 specific bits to update three bits. The output keystream bit is generated by XORing six bits from the internal state using a linear output function. This process is repeated until $n \leq 2^{64}$, where the maximum keystream bits that can be generated by a key-IV pair is 2^{64} bits. Algorithm 2.4 illustrates the keystream generation process.

2.4.3 Grain v1

Description

The Grain v1 stream cipher was submitted to eSTREAM project, by Hell, Johansson and Meier [59] and was selected in the final portfolio of the eSTREAM

```
Require: $S$
1: for $i = 1$ to $n$ do
2:     $t_1 \leftarrow s_{66} \oplus s_{93}$
3:     $t_2 \leftarrow s_{162} \oplus s_{177}$
4:     $t_3 \leftarrow s_{243} \oplus s_{288}$
5:     $z_i \leftarrow t_1 \oplus t_2 \oplus t_3$
6:     $t_1 \leftarrow t_1 \oplus s_{91} \cdot s_{92} \oplus s_{171}$
7:     $t_2 \leftarrow t_2 \oplus s_{175} \cdot s_{176} \oplus s_{264}$
8:     $t_3 \leftarrow t_3 \oplus s_{69} \oplus s_{286} \cdot s_{287}$
9:     $(s_1, s_2, \ldots, s_{93}) \leftarrow (t_3, s_1, \ldots, s_{92})$
10:    $(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (t_1, s_{94}, \ldots, s_{176})$
11:    $(s_{178}, s_{179}, \ldots, s_{288}) \leftarrow (t_2, s_{178}, \ldots, s_{287})$
12: end for
13: print $z_i$
```

Algorithm 2.4: Trivium keystream generation

project. It is a hardware oriented bit-based synchronous cipher. It aims to provide flexible speed up to 16 bit/clock based on the available hardware. Grain takes an 80-bit secret key, K and 64-bit initial vector, IV, as input to a 160-bit internal state as shown in Figure 2.7. The main components of Grain are two feedback shift registers and one output function. One of the registers is a linear feedback shift register (LFSR) with polynomial function $f(x)$, and the second register is a nonlinear feedback shift register (NFSR) with polynomial function $g(x)$. Each shift register has a size of 80 bits. The output function $h(x)$ is a nonlinear function with five-bit inputs (four bits from the LFSR and a bit from the NFSR).

Initialisation Process

The Grain stream cipher performs the initialisation process before producing any keystream bits. Let the secret key bits be denoted k_i, $0 \leq i \leq 79$ and the IV bits be denoted v_i, $0 \leq i \leq 64$. Firstly, the loading phase is performed. The 80-bit secret key is loaded into the nonlinear feedback shift register (NFSR). Let the stages of NFSR be b_i where $b_i = k_i$ for $0 \leq i \leq 79$. The 64 bits of IV are loaded into 64 bits of the linear feedback shift register (LFSR). Let the stages of LFSR be s_i where $s_i = v_i$ for $0 \leq i \leq 63$. The remaining bits of the LFSR are filled by ones, $s_i = 1$ for $64 \leq i \leq 79$. Secondly, the diffusion phase proceeds by clocking the LFSR and NFSR registers 160 times before producing any keystream, and the output bit z_i is fed back and XORed to the input of both the LFSR and NFSR. The general concept of Grain is illustrated in Figure 2.7 with the polynomial functions of $f(x)$ and $g(x)$. The update function of the LFSR and NFSR, $h(x)$ and z_i are defined as follows:

$$\begin{aligned}
s_{i+80} &= s_{i+62} \oplus s_{i+51} \oplus s_{i+38} \oplus s_{i+23} \oplus s_{i+13} \oplus s_i \\
b_{i+80} &= s_i \oplus b_{i+62} \oplus b_{i+60} \oplus b_{i+52} \oplus b_{i+45} \oplus b_{i+37} \oplus b_{i+33} \oplus b_{i+28} \oplus b_{i+21} \oplus \\
&\quad b_{i+14} \oplus b_{i+9} \oplus b_i \oplus b_{i+63}b_{i+60} \oplus b_{i+37}b_{i+33} \oplus b_{i+15}b_{i+9} \oplus \\
&\quad b_{i+60}b_{i+52}b_{i+45} \oplus b_{i+33}b_{i+28}b_{i+21} \oplus b_{i+63}b_{i+45}b_{i+28}b_{i+9} \oplus \\
&\quad b_{i+60}b_{i+52}b_{i+37}b_{i+33} \oplus b_{i+63}b_{i+60}b_{i+21}b_{i+15} \oplus \\
&\quad b_{i+63}b_{i+60}b_{i+52}b_{i+45}b_{i+37} \oplus b_{i+33}b_{i+28}b_{i+21}b_{i+15}b_{i+9} \oplus \\
&\quad b_{i+52}b_{i+45}b_{i+37}b_{i+33}b_{i+28}b_{i+21} \\
h(x) &= x_1 \oplus x_4 \oplus x_0x_3 \oplus x_2x_3 \oplus x_3x_4 \oplus x_0x_1x_2 \oplus x_0x_2x_3 \oplus x_0x_2x_4 \oplus \\
&\quad x_1x_2x_4 \oplus x_2x_3x_4 \\
z_i &= \textstyle\sum_{k\in A} b_{i+k} h(s_{i+3}, s_{i+25}, s_{i+46}, s_{i+64}, b_{i+63}) \\
\text{where} &\quad x_0, x_1, x_2, x_3, x_4 = s_{i+3}, s_{i+25}, s_{i+46}, s_{i+64}, b_{i+63} \\
\text{where} &\quad A = (1, 2, 4, 10, 31, 43, 56)
\end{aligned}$$

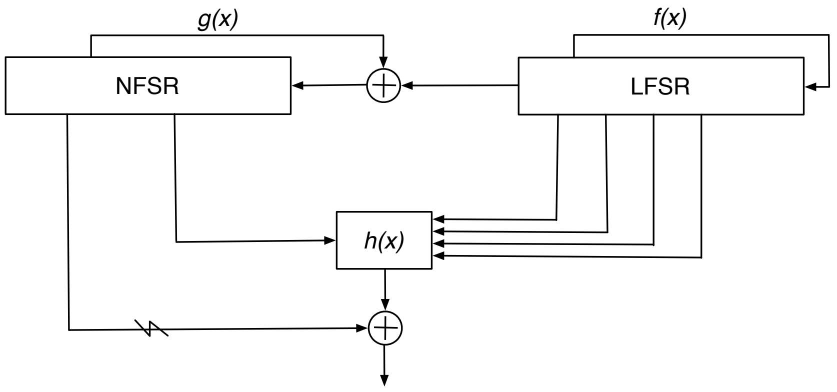


Figure 2.7: Grain construction

Keystream Generation

After the initialisation process is completed, the output bit is no longer used as feedback into both NFSR and LFSR registers. Now Grain v1 is ready to generate the keystream bits z_i. Grain is a bit-based cipher, it generates a bit per clock. Hell et al [59] reported that the period of Grain v1 is at least $2^{80} - 1$ with probability $1 - 2^{-80}$.

Comments on Grain v0 and Grain-128

Grain v1 [59] is released in response to several attacks [13,70,81] on Grain v0 [58]. Both Grain v0 and Grain v1 have the same internal state size (same registers 80-bit LFSR and 80-bit NFSR). They use an 80-bit secret key and a 64-bit IV as input to the ciphers. So, the general structure and components are similar. The bit positions of the feedback and output functions were adjusted according to the new cipher. The actual clocking mechanism also remains unchanged.

Grain-128 [60] was designed later, by Hell, Johansson, Maximov and Meier and submitted to the eSTREAM project as well as an enhancement of the Grain stream cipher. Grain-128 uses 128-bit secret keys and 96-bit IVs. It was designed with a larger secret key and IV length to resist time-memory-data trade-off attack (as they mentioned the 128-bit key is a minimum key length in secure applications). It works similarly to the previous version except for some modifications in the polynomial equations. One bit per clock is produced in the keystream generation. The output can be increased up to 32 bits per clock by adding some hardware components.

2.4.4 Dragon

Description

The Dragon stream cipher was submitted to eSTREAM project, by Chen, Henricksen, Millan, Fuller, Simpson, Dawson, Lee and Moon [32,39]. It is a sofware oriented word-based synchronous stream cipher. Dragon aims to provide high speed throughput by generating two words (64 bits) every clock. (It uses two variable lengths of secret key and initialisation vector, with 256 bits or 128 bits for both as input to a nonlinear feedback shift register (NFSR) 1024-bit internal state.) Dragon is designed to use an F function which consists of three parts: premixing, substitution and post-mixing using G and H functions, two S-boxes (S_1 and S_2) and a 64-bit memory (M). The F function maps six input words to six output words (each word is 32 bits). Algorithm 2.5 shows the specification of the F function where $\oplus$ denotes XOR, $\boxplus$ denotes addition ($mod\ 2^{32}$) and $||$ denotes concatenation. After that, Dragon starts the initialisation process before producing any keystreams using the nonlinear function and multiple iterations to give a high throughput and secure keystream generation.
The G and H functions are:

```
Require: ${a, b, c, d, e, f}$
 1: Pre-mixing Layer:
 2: $b = b \oplus a$; $d = d \oplus c$; $f = f \oplus e$
 3: $c = c \boxplus b$; $e = e \boxplus d$; $a = a \boxplus f$
 4: S-box Layer:
 5: $d = d \oplus G_1(a)$; $f = f \oplus G_2(c)$; $b = b \oplus G_3(e)$
 6: $a = a \oplus H_1(b)$; $c = c \oplus H_2(d)$; $e = e \oplus H_3(f)$
 7: Post-mixing Layer:
 8: $d' = d \boxplus a$; $f' = f \boxplus c$; $b' = b \boxplus e$
 9: $c' = c \oplus b$; $e' = e \oplus d$; $a' = a \oplus f$
10: return ${a', b', c', d', e', f'}$
```

Algorithm 2.5: Dragon: F function

$$\begin{aligned}
G_1(x) =& S_1(x_0) \oplus S_1(x_1) \oplus S_1(x_2) \oplus S_2(x_3)\\
G_2(x) =& S_1(x_0) \oplus S_1(x_1) \oplus S_2(x_2) \oplus S_1(x_3)\\
G_3(x) =& S_1(x_0) \oplus S_2(x_1) \oplus S_1(x_2) \oplus S_1(x_3)\\
H_1(x) =& S_2(x_0) \oplus S_2(x_1) \oplus S_2(x_2) \oplus S_1(x_3)\\
H_2(x) =& S_2(x_0) \oplus S_2(x_1) \oplus S_1(x_2) \oplus S_2(x_3)\\
H_3(x) =& S_2(x_0) \oplus S_1(x_1) \oplus S_2(x_2) \oplus S_2(x_3)
\end{aligned}$$

Initialisation Process

Dragon [32, 39] can use 256 bits or 128 bits for both the secret key and IV as mentioned above. The 1024-bit internal state is formed by register containing the secret key and IV plus the memory. The initialisation process uses the F function extensively with 16 iterations. Rekeying (reinitialising) is performed at least once every 2^{64} bits of the output keystream. Algorithm 2.6 and Figure 2.8 illustrate all the initialisation process of the Dragon stream cipher. For the key-IV in Algorithm 2.6, $\bar{x}$ and x' denote the complement and the swapping of the upper half and the lower half of x respectively.

Keystream Generation

Dragon has an internal state with a size of 1024 bits divided into 32 words. During each cycle, six words (word numbers 0, 9, 16, 19, 30, 31) are used as input for the F. The memory is used as a counter. Each clock, Dragon produces

```
Require: $\{K, IV\}$ (256-bit) or $\{k, iv\}$ (128-bit)
1:  $W_0\|\ldots\|W_7 = K\|K \oplus IV\|\overline{K \oplus IV}\|IV$ (256-bit) or
2:  $W_0\|\ldots\|W_7 = k\|k' \oplus iv'\|iv\|k \oplus iv'\|k'\|k \oplus iv\|iv'\|k' \oplus iv$ (128-bit)
3:  $M = 0x0000447261676F6E$
4:  for $i = 1$ to $16$ do
5:      $a\|b\|c\|d = (W_0 \oplus W_6 \oplus W_7)$
6:      $e\|f = M$
7:      $(a', b', c', d', e', f') = F(a, b, c, d, e, f)$
8:      $W_0 = (a'\|b'\|c'\|d') \oplus W_4$
9:      for $i = 7$ to $1$ do
10:         $W_i = W_{i-1}$
11:     end for
12:     $M = e'\|f'$
13: end for
14: return $\{W_0\|\ldots\|W_7\}$
```

Algorithm 2.6: Dragon's initialisation process

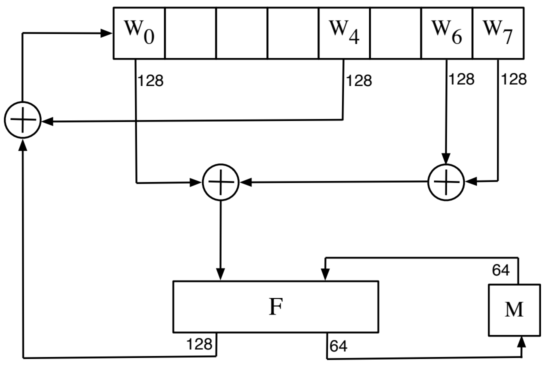


Figure 2.8: Initialisation of Dragon

an output word, z, (64 bits), updates the state and increases the value of memory by one. Algorithm 2.7 shows the general principle of keystream generation.

2.4.5 MICKEY v1

Description

The MICKEY v1 stream cipher was submitted to eSTREAM project, by Babbage and Dodd [6]. It is a hardware oriented bit-based synchronous cipher. MICKEY v1 takes an 80-bit secret key, K and an IV between 0 and 80 bits in length as inputs to a 160-bit internal state (two shift registers) as shown in Figures 2.9. MICKEY is designed to generate up to 2^{40} bits of keystream before

```
Require: ${B_0\|\ldots\|B_{31}, M}$
 1: $(M_L\|M_R) = M$
 2: $a = B_0, b = B_9, c = B_{16}, d = B_{19}, e = B_{30} \oplus M_L, f = B_{31} \oplus M_R$
 3: $(a', b', c', d', e', f') = F(a, b, c, d, f, e)$
 4: $B_0 = b', B_1 = c'$
 5: for $i = 31$ to 2 do
 6:     $B_i = B_{i-2}$
 7: end for
 8: $M = M + 1$
 9: $z = a'\|e'$
10: return $\{k, B_0\|\ldots\|B_{31}, M\}$
```

Algorithm 2.7: Dragon's keystream generator

the rekeying process.

MICKEY v1 consists of two shift registers R and S. Register R is a linear feedback shift register that is clocked based on the value of CB_R. If CB_R is 0, register R is clocked in the standard way. When CB_R is 1, register R is clocked $2^{40} - 23$ times. Register S is a nonlinear shift register that is clocked based on the value of CB_S. Let $r_0, r_1, \ldots, r_{79}$ and $r'_0, r'_1, \ldots, r'_{79}$ denote the stages of register R before clocking and after clocking respectively. Let $s_0, s_1, \ldots, s_{79}$ and $s'_0, s'_1, \ldots, s'_{79}$ denote the stages of register S before clocking and after clocking respectively. The feedback tap positions of register R is defined as: $TAP\_R = \{0,$ 2, 4, 6, 7, 8, 9, 13, 14, 16, 17, 20, 22, 24, 26, 27, 28, 34, 35, 37, 39, 41, 43, 49, 51, 52, 54, 56, 62, 67, 69, 71, 73, 76, 78, 79$\}$. For the state update function fo register S, Let $\hat{s}_0, \hat{s}_1, \ldots, \hat{s}_{79}$ denote intermediate variables. Define four sequences as follows: $COMP0_i$ and $COMP1_i$ for $1 \le i \le 78$, and $FB0_i$ and $FB1_i$ for $0 \le i \le 79$, as shown in Table 2.1.

Initialisation Process

MICKEY v1 performs the initialisation process before producing any keystream bits. At the beginning, all stages are set to be zeros. Then, the IV (v_j, for $0 \le j \le 80$) and 80-bit secret key are loaded into the 160-bit internal state. Let the secret key bits be denoted k_i, $0 \le i \le 79$ and the IV bits be denoted v_i, $0 \le i \le j - 1$. Firstly using a nonlinear function, the IV bits are loaded into both registers R and S. Following that the key bits are loaded in the same manner. MICKEY v1 performs 80 clocks for both registers and the output bits are discarded. The initialisation process is performed as follows:

Table 2.1: The four sequences $COMP0_i, COMP1_i, FB0_i$ and $FB1_i$

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
$COMP0_i$		0	0	0	1	1	0	0	0	1	0	1	1	1	1	0	1	0	0	1	0	1	0	1	0	1	0
$COMP1_i$		1	0	1	1	0	0	1	0	1	1	1	1	0	0	1	0	1	0	0	0	1	1	0	1	0	1
$FB0_i$	1	1	1	1	0	1	0	1	1	1	1	1	1	1	1	0	0	1	0	1	1	1	1	1	1	1	1
$FB1_i$	1	1	1	0	1	1	1	0	0	0	0	1	1	1	0	1	0	0	1	1	0	0	0	1	0	0	1

i	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53
$COMP0_i$	1	0	1	1	0	1	0	0	1	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0	0	0	0
$COMP1_i$	1	1	0	1	1	1	1	0	0	0	1	1	0	1	0	1	1	1	0	0	0	0	1	0	0	0	1
$FB0_i$	1	1	0	0	1	1	0	0	0	0	0	0	1	1	1	0	0	1	0	0	1	0	1	0	1	0	0
$FB1_i$	1	0	0	1	0	1	1	0	0	0	1	1	0	0	0	0	0	1	1	0	1	1	0	0	0	1	0

i	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
$COMP0_i$	1	0	1	0	0	1	1	1	1	0	0	1	0	1	0	1	1	1	1	1	1	1	1	1	0	
$COMP1_i$	0	1	1	1	0	0	0	1	1	1	1	1	1	0	1	0	1	1	1	0	1	1	1	1	0	
$FB0_i$	1	0	1	1	1	1	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	1	1	0	1	0
$FB1_i$	0	0	1	0	0	1	0	0	1	0	1	1	0	1	0	1	0	0	1	0	1	0	0	0	1	1

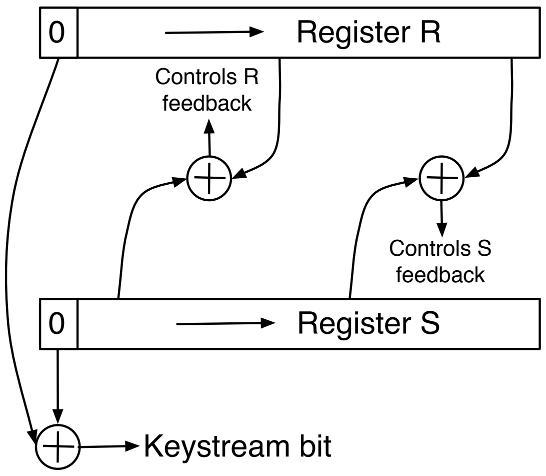


Figure 2.9: MICKEY stream cipher

- Loading phase:
  - Initialise both registers R and S to all-zeros contents,
    $r_i = s_i = 0$ for $0 \leq i \leq 79$.
  - Load the IV bits to both registers R and S simultaneously,
    $CLOCK\_KG(R, S, 1, v_i)$, for $0 \leq i \leq j - 1$.
  - Load the key bits to both registers R and S simultaneously,
    $CLOCK\_KG(R, S, 1, k_i)$, for $0 \leq i \leq 79$.
- Diffusion phase:
  - $CLOCK\_KG(R, S, 1, 0)$, for $0 \leq i \leq 79$.

```
Require: $R$ and $S$ to execute $CLOCK\_KG(R, S, MIX, In\_B)$
 1: $CB\_R = s_{27} \oplus r_{53}$
 2: $CB\_S = s_{53} \oplus r_{26}$
 3: if MIX =1 then
 4:     $IB\_R = In\_B \oplus s_{40}$
 5: else
 6:     $IB\_R = In\_B$
 7: end if
 8: $IB\_S = In\_B$
 9: $CLOCK\_R(R, IB\_R, CB\_R)$
10: $CLOCK\_S(S, IB\_S, CB\_S)$
```

Algorithm 2.8: Clocking the overall generator

```
Require: $R$ to execute $CLOCK\_R(R, IB\_R, CB\_R)$
 1: $FB\_R = r_{79} \oplus IB\_R$
 2: for $i = 1$ to 79 do
 3:     $r'_i = r_{i-1}$
 4: end for
 5: $r'_0 = 0$
 6: for $i = 0$ to 79 do
 7:     if $i \in TAP\_R$ then
 8:         $r'_i = r'_i \oplus FB\_R$
 9:     end if
10: end for
11: if $CB\_R = 1$ then
12:     for $i = 0$ to 79 do
13:         $r'_i = r'_i \oplus r_i$
14:     end for
15: end if
```

Algorithm 2.9: Clocking register R

Keystream Generation

During the keystream generation, the state update function no longer uses s_{40} for the feedback of register R. Each clock, MICKEY v1 generates a keystream bit as follows: $z_i = r_0 \oplus s_0$.

$CLOCK\_KG(R, S, 0, 0)$, for $0 \leq i \leq 79$.

Comments on the MICKEY v2 and MICKEY-128

Hong and Kim [65] demonstrated a state entropy loss weakness (state convergence) in MICKEY v1. To overcome this weakness, the designers proposed an

```
Require: $S$ to execute $CLOCK\_S(S, IB\_S, CB\_S)$
 1: $FB\_S = s_{79} \oplus IB\_S$
 2: for $i = 1$ to 78 do
 3:     $\hat{s}_i = s_{i-1} \oplus ((s_i \oplus COMP0_i) \cdot (s_{i+1} \oplus COMP1_i))$
 4: end for
 5: $\hat{s}_0 = 0$
 6: $\hat{s}_{79} = s_{78}$
 7: if $CB\_S = 0$ then
 8:     for $i = 0$ to 79 do
 9:         $s'_i = \hat{s}_i \oplus (FB0_i \cdot FB\_S)$
10:     end for
11: else
12:     for $i = 0$ to 79 do
13:         $s'_i = \hat{s}_i \oplus (FB1_i \cdot FB\_S)$
14:     end for
15: end if
```

Algorithm 2.10: Clocking register S

improved cipher, referred to as Mickey v2 [8]. Mickey v2 is one of the stream ciphers in the final eSTREAM project portfolio, for the second profile (hardware). Register lengths of R and S were increased to be 100 bits instead of 80 bits for each register. The tap positions and the feedback and clocking were adjusted according to the new proposal. The general structure and components are similar to Mickey-v1 and the clocking mechanism remains unchanged.

MICKEY-128 v1 [5] was designed to use a 128-bit secret key. This cipher proposal has similar design and clocking mechanism principles to MICKEY v1 with two registers each 128-bit, where the clocking and feedback taps were relocated accordingly. Later, another improvement of MICKEY-128 v1 was proposed and referred as MICKEY-128 v2 [7]. MICKEY-128 v2 has increased register lengths to be 160-bit for registers R and S instead of 128-bit each. This is the only difference.

2.4.6 LILI-II

Description

The LILI-II stream cipher was designed by Clark, Dawson, Fuller, Golić, Lee, Millan, Moon, and Simpson [35] which is from the LILI family together with LILI [95] and LILI-128 [38] (was submitted to NESSIE project [68]). It is hardware and software oriented, and a bit-based synchronous cipher. LILI-II takes 128-bit key and 128-bit IV as input to a 255-bit internal state (two shift registers) as shown in Figure 2.10. It was designed to have a period of 2^{255}.

Let k_i and v_i denote the key and IV bits for $0 \leq i \leq 127$. There are two registers LFSR$_c$ and LFSR$_d$ of lengths 128 and 127 bits respectively. LFSR$_c$ is clocked normally and the LFSR$_d$ is controlled by a function f_c. This function f_c takes two inputs from the LFSR$_c$ to calculate the output $c(t)$ at time t, where $c(t) \in \{1, 2, 3, 4\}$. The output Boolean function f_d takes twelve bits from register LFSR$_d$ as inputs to produce the output bit. Let c_i and d_j denote the stages for registers LFSR$_c$ and LFSR$_d$ for $0 \leq i \leq 127$ and $0 \leq j \leq 126$ respectively.

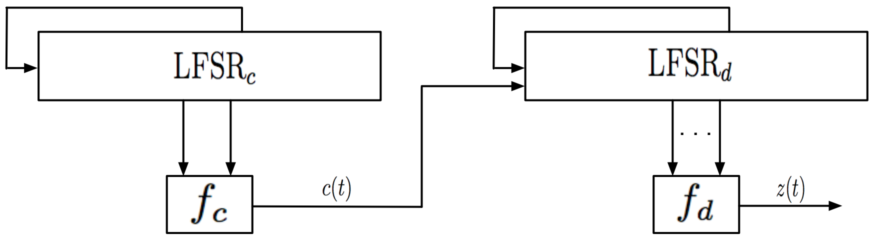


Figure 2.10: LILI-II stream cipher

Initialisation Process

LILI-II performs the initialisation process as follows:

- Loading LFSR$_c$, the secret key, k, and the IV, v, are XORed to generate the loaded state of LFSR$_c$ as follows, $c_i = k_i \oplus v_i$, for $0 \leq i \leq 127$.
- Loading LFSR$_d$, the secret key, k, and the IV, v, are XORed to generate the loaded state of LFSR$_d$ by ignoring k_0 and v_{79} as follows:, $d_i = k_{i+1} \oplus v_i$, for $0 \leq i \leq 126$.
- Generates 255 keystream bits, the first 128 bits are loaded to register LFSR$_c$, and then 127 bits are loaded to register LFSR$_d$.
- Generates another 255 keystream bits, the first 128 bits are loaded to register LFSR$_c$, and then 127 bits are loaded to register LFSR$_d$.

Keystream Generation

Now, the LILI-II stream cipher has completed the initialisation process and the keystream generation can begin. The keystream generation process uses the

same state update function from the initialisation process. The keystream bit is formed from the f_d function as mentioned above:
$z_i = f_d(d_0,\, d_1,\, d_3,\, d_7,\, d_{12},\, d_{20},\, d_{30}, d_{44},\, d_{65},\, d_{80},\, d_{96},\, d_{122})$

2.4.7 Why A5/1, Sfinks and CSA-SC

We first discuss the rationale behind the selection of the three ciphers A5/1, Sfinks and the Common Scrambling Algorithm Stream Cipher (CSA-SC) as subjects for in-depth analysis of the initialisation processes. These are all commonly used ciphers which take different approaches to the loading and diffusion phases of initialisation and there has been a limited analysis of security of their initialisation processes. Also, these three ciphers use different state, key and IV sizes, and perform different approaches of key and IV loading processes.

Most stream ciphers use shift registers in combination with non-linear functions. So, we choose these ciphers because they have the same structure as most other stream ciphers. In common with most stream ciphers, these three ciphers use shift registers with variety of structures and nonlinear functions. They cover both cases when state size is less and larger than the total of the key and IV sizes. They cover three different type of loading phases and initialisation processes: with majority clocking scheme, update function with delay and loading the IV during the diffusion phase. Moreover, they cover autonomous and non-autonomous operations for both loading process and update function. Therefore, the recommendations and outcomes from these analyses are relevant to most stream ciphers that are based on shift registers.

A5/1 Stream Cipher

The A5/1 [30] cipher is a well known stream cipher which is used widely to provide confidentiality for GSM mobile phone communications. It is used by about 130 million customers in Europe [24]. The keystream generation process of A5/1 has been heavily analysed, (see [12,19,53,82]), but there was only limited analysis of the initialisation process prior to this research [24,54]. The description of A5/1 and its initialisation processes are given in Section 3.1.

Sfinks Stream Cipher

The Sfinks stream cipher was submitted to eSTREAM project, the ECRYPT call for stream cipher proposals in April 2005, by Braeken, Lano, Mentens, Preneel and Verbauwhede [29]. The Sfinks initialisation process is not well studied in the public literature. The individual components of the state update function of Sfinks are one-to-one, but during the initialisation process their combination is not one-to-one. This is the central point in choosing this cipher to be investigated. The description of Sfinks and its initialisation processes are given in Section 4.1.

Common Scrambling Algorithm Stream Cipher

The Digital Video Broadcasting Common Scrambling Algorithm (DVB-CSA) is specified by the European Telecommunications Standards Institute (ETSI) [51] . It has been used for scrambling and encrypting the MPEG-2 transport stream (digital television or pay TV) since 1994. The DVB-CSA consists of a cascade of block and stream ciphers. In this thesis, we examine the stream cipher in DVB-CSA which is referred to as CSA-SC [16]. There is some analysis of the keystream generation process of the CSA-SC [74,96,104,105], but the initialisation process of CSA-SC is not well studied in the public literature. The description of CSA-SC and its initialisation processes are given in Section 5.1.

Table 2.2 presents the summary of characteristics of the ciphers that have been discussed in this section. It also includes three ciphers A5/1, Sfinks and CSA-SC which will be discussed and analysed in great detail in Chapters 3, 4 and 5. It shows some of the important components and comparison among these ciphers.

Table 2.2: Summary of well known stream cipher proposals

Stream Cipher	Cipher Based	Word size (bits)	Key size (bits)	IV size (bits)	State size (bits)	Storage type	Iterations before generation	Output size (bits)	rekeying time (bits)	Similarity of functions*
RC4	byte	1 byte	l-byte	-	2064	Array	0	8	n/a	n/a
LILI-II	bit	1	128	128	255	LFSR	2	1		same.
Grain v1	bit	1	80	64	160	LFSR/NFSR	160	1-16		same
Grain-128	bit	1	128	96	256	LFSR/NFSR	256	1-32		same
MICKEY v2	bit	1	80	0-80	200	LFSR	99	1	2^{40}	modi.
MICKEY-128	bit	1	128	0-128	320	LFSR	159	1	2^{64}	modi.
Trivium	bit	1	80	80	288	NFSR	1152	1	2^{64}	modi.
Dragon	word	32	128/256	128/256	1088	NFSR/memory	16	64	2^{64}	diff.
A5/1	bit	1	64	22	64	LFSR	100	1	228	same
Sfinks	bit	1	80	80	256	NFSR	128	1		modi.
CSA-SC	word	4	64	64	89	NFSR	32	2		modi.

Note: * The similarity of state update functions of initialisation and keystream generator is as follows.
same: the state update functions are similar
modi.: the state update functions have a degree of modification
diff.: the state update functions are completely different

2.5 Cryptanalysis of Stream Ciphers

Attacking stream ciphers usually aims to either distinguish between stream ciphers, to find internal state, to recover secret key or message recovery. These attacks are discussed in more detail in Section 2.5.2. Attackers exploit any available information, methods or tools that may assist the attacking process.

Certain assumptions are made when analysing any encryption algorithm. From the general viewpoint of cryptanalysts, the stream cipher algorithms and initial vectors (IVs) are assumed to be known. The challenge is to break a stream cipher given the algorithm and also any IVs that are used but assuming that the secret key is unknown.

For more specific situations there are some assumptions about each attack method. For example, slide attacks assume the self-similarity of iterations of the stream cipher [25]. In another example, the algebraic attack assumes the attacker has gained knowledge of the stream cipher algorithm and some keystream bits [3]. Consequently, there are general assumptions for cryptanalysis and specific assumptions for each cryptanalysis method as well as for each application.

The complexity of attacks and cryptanalysis of stream ciphers are usually measured by the following parameters:

- *Memory complexity.* The amount of memory that is required to hold information for attacking. Memory may be required for pre-computation and the attack itself such as in the Time Memory Time Trade-Off attack (TMTO), which is discussed later.
- *Time complexity.* The required time to perform an attack.
- *Data complexity.* The amount of ciphertext/plaintext that is required to execute an attack.

The main focus of a cryptanalyst is to develop and find an attack approach with the lowest possible complexity (memory, time and amount of data). In general, there are trade-offs between these parameters. For example, TMTO attack is a good example of a trade-off attack.

2.5.1 Attacking Models

Attacking models specify the type of information which can help an attacker to break a stream cipher. The model describes the information that can be accessed by an attacker. The attacking models for stream ciphers are based on Kerckhoffs' [69] and Shannon's principles [92] where the description of ciphers are publicly known and the secrecy depends on the secret key only. The attacking models are classified into the following types based on which information is known.

- *Ciphertext-only attack.* An attacker is able to access only the ciphertext from a stream cipher.
- *Known-plaintext attack.* In this model, an attacker has some pairs of plaintext-ciphertext that use the same secret key.
- *Chosen-plaintext attack.* An attacker is able to choose a random plaintext to be encrypted in order to obtain the corresponding ciphertext. The secret key is assumed to be unknown.
- *Chosen-ciphertext attack.* An attacker is able to decrypt a chosen ciphertext to obtain the corresponding plaintext assuming the secret key is unknown.

A stream cipher which is broken by a ciphertext-only attack is considered a weak stream cipher. Due to the rapid development of technology, stream ciphers should be continually under analysis to ensure that they are still secure or to determine their limitation of application.

2.5.2 Aims of Attackers

Attackers may have one or more of the attacking aims or goals. These aims can be classified as follows.

- Distinguishing attack [87]: Since the keystream sequence, which is an output of a stream cipher, is claimed to be random for a specific period, distinguishing attack is a generic attack that is used to determine which stream cipher has been used. Distinguishing attack exploits some security flaws to build a relationship between a keystream sequence and the stream cipher used. Even though a distinguishing attack alone does not reveal information about the secret key or the initial state (session key), it demonstrates some weakness in the stream cipher. It may be a part of an attacking process. It may be used with another attack (may be generic attack) to reduce the complexity of attacking or reveal information about the secret key or the initial state. Therefore, most stream cipher design prescribes the maximum number of keystream bits to be generated before rekeying the system by a new IV. This determines the maximum possible keystream that can be generated from a single key-IV pair to reduce the possibility of any attack.

- State recovery attack [94]: The state recovery attack aims to recover the initial state of a stream cipher, which is known as session key recovery attack. If an attacker is able to recover the initial state, then the entire keystream for this session can be generated easily. Attacker needs to perform the attacking process for each session key. Ciphers which are vulnerable for state recovery attack are considered to be insecure.

- Key recovery attack [31, 93]: Key recovery attack is the main purpose of attackers. If an attacker is able to recover the secret key, then keystream can be generated for any known IV easily and the stream cipher is completely broken. Therefore, modern stream ciphers designers aim to make key recovery attack hard from a given session key recovery attack. So, the initialisation process plays a role like a security guard.

- Message recovery attack [40]: In this type of attack, an attacker may have the ability to recover the message without any knowledge of the secret key. It is based on the properties of the massage, such as the redundancy and XORing of messages as shown in [40].

2.6 Generic Attacks on the Initialisation Process

This section presents some generic attacks for stream ciphers that are used also to attack the initialisation process of stream ciphers.

2.6.1 Brute Force Attack

For the initialisation process, we aim to find the secret key (master key). Brute force attack is a naive attack on ciphers, where all possible secret keys are tried until a secret key which produces the keystream is found. This is also known as exhaustive search. However, most stream ciphers are designed so that it is impractical to use the brute force attack due to the time required. In practice, the average number of trials of brute force attack is half the size of the key space. For example, if the secret key length is l then the total key space is 2^l and half of the key space is 2^{l-1}.

Most attacks are claimed to be better than the brute force attack, that means the time complexity of an attack is claimed to be less than the complexity of the brute force attack. So, brute force attack is sometimes used to evaluate the effectiveness of a new attack.

2.6.2 Time Memory Data Trade-Off Attack

Time-Memory Trade-off (TMTO) was introduced in 1980 by Hellman [62] as a generic attack on block ciphers and applied to the Data Encryption Standard (DES). TMTO is a general technique to invert the one-way function. The effort required lies somewhere between exhaustive search of the search space and the look-up table using two phases, a pre-computation phase and a real-time phase. The pre-computation phase aims to build tables of random states x_i and their corresponding keystream sequences y_i where $f(x) = y$, where x_i represents either the secret key or the internal state (as input), y_i is the keystream (as output) and f is the function that maps input to the output. This function f could be the initialisation update function of the stream ciphers or the state update function of the keystream generation and the output functions. These tables are sorted into increasing order of the y_i.

Time-Memory Trade-off (TMTO) has also been applied to stream ciphers. Independent works by Golić [54] and Babbage [9] applied the TMTO to keystream generators of stream ciphers to perform the internal state recovery attack. Moreover Biryukov and Shamir [23] applied the above works of Hellman and Golić-Babbage to stream ciphers to recover the internal state as well. In these three independent analyses, the researchers recommended that the internal state size should be at least double the size of the secret key.

The complexity of the time memory data trade-off attack can be described in terms of the pre-computation and online parameters. The pre-computation terms are as follows: P is the time that is required to construct the lookup table (pre-computation time), and M is the size of memory required to construct and store the lookup table. During the attacking time (online phase), D is the amount of keystream that is required to perform the TMTO attack, and T is the online time that is required to search in the lookup table. From the above notation, the attack complexity is based on both T and M, where P is pre-computation time and the attacker is assumed to have unlimited time for preparation.

Now, we will focus on the treatment of initialisation processes in TMTO. Hong and Sarkar [66, 67] revised the TMTO to be applied for stream ciphers. They aimed to perform secret key recovery attack for stream ciphers that use secret key and initial vector (IV). Hong and Sarkar used the Biryukov and Shamir TMTO for a search space $N = 2^{l+j}$, where l and j are the key and IV sizes in bits, respectively.

$$T = M = 2^{\frac{1}{2}(l+j)} \quad \text{with} \quad D = 2^{\frac{1}{4}(l+j)} \tag{2.2}$$

Thus, if an attacker has access to a keystream of length $D = 2^{\frac{1}{4}(l+j)}$, then that attacker can recover a single (k, v) pair with time and memory complexity of $T = M = 2^{\frac{1}{2}(l+j)}$. If $j < l$ then the time and memory complexity is less than the exhaustive search. In order to resist Hong and Sarkar's TMTO attack, the IV size should be at least equal to the key size.

Dunkelman and Keller [46] applied the TMTO attacks for stream ciphers to recover the secret key. They assumed that the IVs are publicly known information. They proposed to construct lookup tables based on the IVs. So, an attacker waits for a set of IVs to capture the keystream to perform the attack according to the following trade-off curve:

$$2^{2(l+j)} = TM^2D^2 \tag{2.3}$$

There is no restriction for the parameters of the Dunkelman and Keller TMTO attack such as $T \geq D^2$. If $T = M = D$ then the complexity of the attack is less than the exhaustive search as long as $2^{2j} < 2^{3l}$. In order to resist this attack, they recommended that the IV size should be at least 1.5 times the key size, $j \geq 1.5l$.

2.6.3 Differential Attacks

A differential attack is a generic cryptanalysis method introduced in 1990 by Biham and Shamir in an attack on block ciphers [21]. It was applied to stream ciphers in 2004 by Muller [85]. The attacker considers multiple keystreams obtained with a slight difference in the inputs (usually a difference in the IVs).

The differential attack on stream ciphers examines the behavior of the initialisation and keystream generation processes for a differential in the inputs. It analyses how differences in the inputs (either the key or IV, or internal state) affect the output (either internal state or keystream). Biham and Dunkelman [20] describe three situations as follows:

1. A difference in the key or IV generates a difference in the internal state, S, $(\Delta k, \Delta v) \longrightarrow \Delta S$, with probability of p_1.
2. A difference in the internal state, S, generates a difference in the internal state after α clocks, $\Delta S_t \longrightarrow \Delta S_{t+\alpha}$.
3. A difference in the internal state, S, generates a difference in the keystream, Z, $\Delta S \longrightarrow \Delta Z$ with probability of p_2.

Considering these inputs to the keystream generator and combining steps (1) and (2) in the above, we obtain the following:

- A difference in the key or IV generates a difference in the keystream, $(\Delta k, \Delta v) \longrightarrow \Delta Z$, with probability of $p_1 \cdot p_2$.

For the keystream generators for stream ciphers an important attack scenario to consider is where an attacker can access multiple keystreams formed using the same secret key but with different known IVs; as (k, v) and (k, v'), for Δv. Then the difference in the two keystreams is ΔZ. When Δv and ΔZ are known, then the relationship between the two keystreams may reveal some information about the secret key.

Differential cryptanalysis involves choosing two different inputs to a keystream generator or cipher system and observing the difference between the outputs. The difference may be used to distinguish the stream cipher or to reveal some information about the secret key or the internal state. Stream ciphers can be protected against differential attacks by designing the initialisation, keystream generation processes and the output carefully. The IVs should be mixed correctly with the secret key. These processes should result in unpredictable differences in the internal state or keystream. Muller [85] stated that the initialisation process can provide protection against differential attacks by mixing nonlinear and linear functions.

2.7 Flaws in the Initialisation Process

In the following section, we present some potential flaws in the initialisation process of stream ciphers that can be used with some known generic attacks to disclose information about the secret key or the encrypted message. These flaws are state convergence and the existence of slid pairs and of weak Key-IV combinations. These three flaws are discussed and investigated because they happen commonly in shift register based stream ciphers. They occur due to some common structural features and properties in the initialisation processes functions. Flaws in the initialisation processes need more research and investigation. Although this research focuses on these three common flaws, there may be other flaws or new forms of attacks exploit a specific type of flaw (which may be discovered in the future). Moreover, other stream ciphers that are not based on shift registers may also have weaknesses related to the initialisation. For example a widely used stream cipher, RC4, is a stream cipher based on a dynamic table. The initialisation process of other types of stream cipher design should be analysed carefully. In this thesis, we investigate only shift registers based designs.

2.7.1 State Convergence

Modern stream ciphers take a secret key and an IV as input to the initialisation process of the keystream generator to produce an initial state (session key). If the state size is equal to or larger than the total of the key and IV sizes, then each key-IV combination should map to a distinct internal state. If the state size is shorter than the key and IV sizes, then there is a reduction of key-IV space (compression). However even if a cipher satisfies this state size condition, it can still experience a reduction of the effective state size during either or both of the initialisation and keystream generation processes. This effect is known as state convergence.

State convergence occurs when two or more states are mapped to the same state after α iterations for $\alpha > 0$. That is, state convergence occurs when the state update function is not one-to-one . State convergence may occur during the initialisation process, during keystream generation, or both. This may reduce the effective key-IV size and leave the stream cipher vulnerable to attacks such as distinguishing attacks [87], time-memory-data trade-off attacks [23] or other ciphertext-only attacks [40].

2.7.2 Analysis of State Convergence

State convergence can be investigated by examining the state update transition. It may not be straightforward to decide whether the stream cipher experiences state convergence or not. Therefore, in some cases more than one approach to investigate the convergence is required. These approaches are described below:

Forward Direction

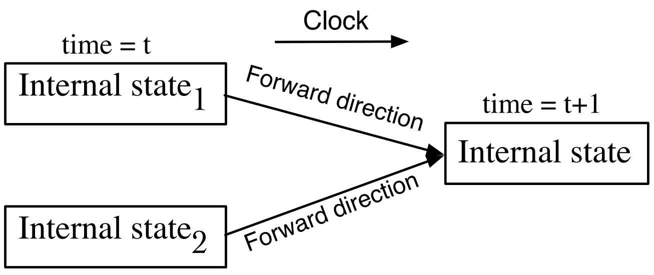
The forward direction follows the normal clocking operation (state update function) in the stream cipher initialisation process. This approach examines the state space at time $t + \alpha$, given the state space at time t, for $\alpha \geq 1$ iterations, as shown in Figure 2.11a. If there are a number of states at time t that give the same state at time $t + \alpha$, then state convergence occurs. This approach is applied for the investigation of state convergence during the initialisation and keystream generation processes of the Common Scrambling Algorithm Stream Cipher as outlined in Chapter 5.

Reverse Direction

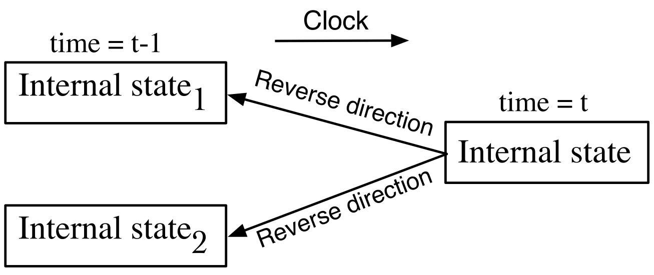
This approach considers the inverse of the state update function, F^{-1}, where F denotes the state update function. If a state is given at time t, then the reverse direction investigation examines the number of pre-images of this state at time $t - 1$ (or more generally, $t - \alpha$) as shown in Figure 2.11b. If the state update function is one-to-one then there is only one pre-image for each state. If it is not one-to-one then there could be none, one or multiple pre-images for some states.

This approach is used to examine state convergence in the A5/1 stream cipher in Chapter 3.

The theoretical analysis of state convergence contributes to determine the relationship between the multiple states that result in the same state after α iterations. Based on this relationship between these multiple states, it is possible in some ciphers to identify the relationship between multiple key-IV pairs that result in the same initial state (and same keystream). This analysis leads to investigate the consequences of state convergence as shown later.



(a) Forward direction



(b) Reverse direction

Figure 2.11: Two states converge to one state

Experimental Work

Experimental work may be performed to investigate state convergence using computer simulation for either the forward or reverse clocking. In most cases, exploration of the entire state space or key-IV space is not feasible. Experimental simulation gives an estimation of the number of distinct internal states for the target cipher after a number of iterations (or distinct keystream sequences). Simulation is performed using either randomly selected inputs to the real cipher or exhaustive search over inputs of a scaled down version of the target stream cipher.

Simulation using the random inputs to the real ciphers requires a large random number of inputs. For example, this approach has been used for 10^8 random states of A5/1 to compute the number of distinct initial states obtainable after 100 iterations during the diffusion phase in [24] using the forward direction. Also, it has been used previously in the reverse direction for 2^{20} random states of the MICKEY stream cipher, in [65].

The scaled down version of a cipher must represent most of the features and properties of the target stream cipher. In this approach an exhaustive search over all possible key-IV pairs is performed. This approach has been used for a scaled version of the A5/1 stream cipher in [2].

Consequences of State Convergence

With respect to the key and IV used with the stream ciphers, there are three cases of state convergence that can occur. These are as follows:

Case 1 Two input pairs have the same secret key but different IVs. During initialisation sate convergence occurs and the pairs (K, IV) and (K, IV$'$) produce the same initial state and consequently the same keystream.

Case 2 Two input pairs have different secret keys and different IVs. During initialisation sate convergence occurs and the pairs (K, IV) and (K', IV$'$) produce the same initial state and consequently the same keystream.

Case 3 Two input pairs have different secret keys but the same IV. During initialisation sate convergence occurs and the pairs (K, IV) and (K', IV) produce the same initial state and consequently the same keystream.

For an attacker the most beneficial of these is Case 1. The situation described in Case 1 can occur between separate frames of a single message, particularly on a frame based channel, where the same secret key is used with multiple IVs in a communication between two parties. For practical attacks on synchronous stream ciphers, Case 1 is the most important flaw, as it may leak information about the secret key.

2.7.3 Slid Pairs and Shifted Keystream

The slide attacks are known-plaintext attacks against iterative ciphers in which successive iterations of the state update function are identical. This property means that the applicability of this attack is independent of the number of iterations used. This generic attack was specified for block ciphers by Biryukov and Wagner in 1999 [25] and 2000 [26]. The attack is based on the related key attack, which was introduced by Biham in 1994 [18]. The first step in this direction of attack was by Grossman and Tuckerman in 1978 [56]. Slide attacks have also been applied to stream ciphers that are based on block ciphers such as LEX [106] and WAKE-ROFB [25] and more recently to other stream ciphers, such as Grain [42, 73, 108] and Trivium [86]. In the stream ciphers' application, slide attacks are sometimes revered to as slid pairs attacks, resynchronisation attacks [73, 106] or related key chosen IV attacks [76].

In the case of stream ciphers, during initialisation a key-IV pair is loaded into an internal state, to form a loaded state. The state is then updated for α iteration(s) during the diffusion phase of the initialisation process. The state update function of the initialisation process defines a cycle of transitions of the internal state. Therefore, some $(K,\ \text{IV})$ pair represents a point on such a cycle. If a state that is obtained from this process can also be obtained as a loaded state for another key-IV pair then the two loaded states form a slid pair that are in the same cycle. Consequently, different key-IV pairs can be found for stream ciphers that sometimes produce phase shifted keystream [25, 42, 73, 86, 108], since stream ciphers use finite state machines (with predefined state update functions) to perform the initialisation and keystream generation processes.

Figure 2.12 illustrates the initialisation and keystream generation processes resulting in two keystream sequences shifted by α bit(s). The relationship between the two keystreams z and z' is given by Equation 2.4.

$$\begin{aligned} &(K,\ \text{IV}) \Longrightarrow z \\ &(K',\ \text{IV}') \Longrightarrow z', \text{ where } z' = z \lll \epsilon \cdot \alpha \end{aligned} \tag{2.4}$$

where ϵ is a positive constant that depends on the design of the output function of the stream cipher. For bit based stream cipher, $\epsilon = 1$.

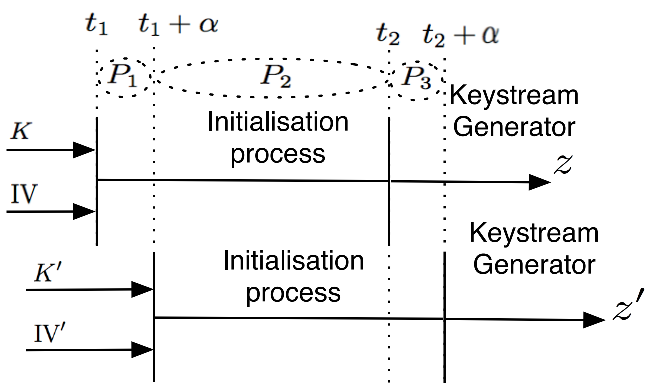


Figure 2.12: Slid pairs and stream ciphers

The probability of obtaining a slid pair that results in a correspondingly shifted keystream depends on three probabilities; P_1, P_2 and P_3. These are as follows. The probability of getting a legitimate loaded state with (K', IV′) pair after α iterations of the initialisation process is P_1. To obtain a shifted version of the keystream from a given slid pair, the state updates for the final $t_2 - (t_1 + \alpha)$ clocks of diffusion phase with (K, IV) must give the same result as the first $t_2 - (t_1 + \alpha)$ clocks with (K', IV′) with probability of P_2. Likewise, the state updates of the initialisation process of the last α iterations for the (K', IV′) should be same as the state update function during the keystream generation process for (K, IV) with probability of P_3. Therefore, the total probability for obtaining shifted keystream from a randomly chosen key-IV is the product of these three probabilities, as shown in Figure 2.12. For most stream ciphers, $P_2 = 1$.

When a key-IV pair (K', IV′) produces a loaded state that can also be obtained from another key-IV pair (K, IV) after a number of iterations α of the initialisation state update function, we refer to these two states as a slid pair. The keystream generated by the pair (K', IV′) may then be a phase-shifted version of the keystream generated by (K, IV), shifted by $\epsilon \cdot \alpha$ bits. This occurs when the following properties hold for the initialisation process and key stream generation:

a) Iterations of the initialisation process are the same as each other.

b) Iterations of the keystream generation process are the same as each other.

c) State update functions for both the initialisation and keystream generation processes have a degree of similarity with one another.

Most stream ciphers follow the (a) and (b) conditions above, but there is a wide variety in the extent to which condition (c) applies. Slid pairs can be expressed as $(K,\ \text{IV}), (K',\ \text{IV}'), \alpha\}$, where $(K,\ \text{IV})$ is a key-IV pair that produces another loaded state with key-IV pair $(K',\ \text{IV}')$ after α iterations of the initialisation state update function.

2.7.4 Analysis of Slid Pairs and Shifted Keystream

Slid pairs may occur in stream ciphers. This can be investigated by exploring the state update function using the forward or reverse directions as shown in Figure 2.13.

Forward Direction:

For a given loaded state, the state update function is clocked (in the forward direction) until the internal state is in the format required for a loaded state. This new loaded state is generated after applying the clocking mechanism for $\alpha \geq 1$ clock. If this occurs, the clocking for both loaded states must be further examined to determine whether they give a shifted keystream. This type of analysis should consider the similarity between the state update functions for both the initialisation and keystream generation processes.

Reverse Direction:

In this analysis a bit shifted version of a keystream (shifted by one or more bits) is given. Then the task is to clock the cipher in reverse direction to investigate the corresponding loaded state. If the obtained state is a legitimate loaded state, then the two pairs are slid pairs.

The theoretical analysis of slid pairs reveals the relationship between the multiple key-IV pairs that generate slid pairs with a difference of α clocks (iterations). So, based on this analysis of slid pairs and shifted keystreams, it is possible to identify the relationship between these multiple key-IV pairs that result in shifted keystreams. This analysis leads to an investigation into the consequences of slid pairs as outlined later.

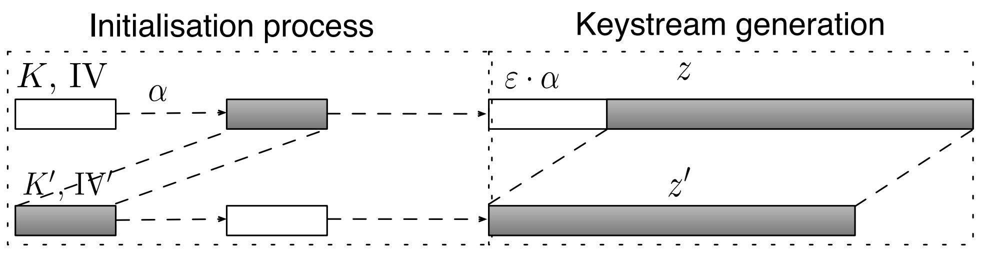


Figure 2.13: Slid pairs in stream ciphers

Experimental Work

The experimental work requires computer simulation for the initialisation and keystream generation processes in either the forward or reverse clocking direction. In most cases, simulation of the entire state space or key-IV space is not feasible. Therefore, simulation can be applied using a large number of randomly selected key-IV pairs. Experimental simulation gives a probabilistic estimation of the actual number of slid pairs and shifted keystreams.

Consequences of of Slid Pairs

Slid pairs can be used as the basis of an attack on some stream ciphers. Similar to the conditions for state convergence, there are three possible cases of key-IV combinations that produce slid pairs in stream ciphers as follows:

Case 1 Two input pairs which have the same secret key but different IVs produce out-of-phase keystream sequences. That is (K, IV) and (K, IV$'$) produce z and z', respectively.

Case 2 Two input pairs which have different secret keys and different IVs produce out-of-phase keystream sequences. That is (K, IV) and (K', IV$'$) produce z and z', respectively.

Case 3 Two input pairs which have different secret keys but the same IV produce out-of-phase keystream sequences. That is (K, IV) and (K', IV) produce z and z', respectively.

where $z' = z \ll \epsilon \cdot \alpha$, for some positive number ϵ and α is the number of iterations between the loaded states in the slid pairs.

Whenever slid pairs generate shifted keystream, the corresponding ciphertexts can be combined to reveal some of the message content; for example, consider messages m_1 and m_2 where m_1 is encrypted using z and a second message m_2 is encrypted using z' to give encrypted frames, f_1 and f_2 respectively. Then:

$$(f_1 \ll \epsilon \cdot \alpha) \oplus f_2 = (m_1 \ll \epsilon \cdot \alpha) \oplus m_2$$

As shown in [40], it is possible to attack an XOR combination of the form $(m_1 \ll \alpha) \oplus m_2$ using the redundancy of the plaintext. Furthermore, when this occurs for Case 1 (same secret key with different IVs) this may leak information about the common secret key. This is the case for A5/1 cipher, as shown in Section 3.3. From a cryptographic perspective, Case 1 is the most important flaw for attackers, because this situation can occur between separate frames of a single message on a frame based channel such as GSM, where the same secret key is used for each frame in a communication with different IV's.

2.7.5 Weak Key-IV Combinations

For some stream ciphers, certain key-IV pairs may lead to internal states in which one or more of the component registers have all zero contents. It may lead to constant or distinguishable keystream as will be shown for A5/1. If this occurs at the end of the initialisation process and the component is autonomous during the keystream generation, then the component may remain in an all-zero state. Then the complexity of keystream generation is effectively reduced. This may in turn lead to serious security flaws. Such key-IV pairs are known as weak key-IV pairs. This phenomenon has previously been observed in Grain v0, v1 and 128 [108] and we will consider it in Chapter 3 for the A5/1 cipher.

2.7.6 Analysis of Weak Key-IV Pairs

Weak key-IV pairs may occur in stream ciphers due to the impact of external inputs or feedbacks to a shift register. This can be investigated by exploring the influence of the external inputs using the forward or reverse directions.

Forward Direction

For a given loaded state, the state update function is clocked forward until the external input is completed (where there is no more external input). This investigation may end in a state where one or more registers contain all-zero values.

This result may be at the end of the loading phase or diffusion phase of the initialisation process. If a key-IV pair results in one or more registers containing all-zero values, then this key-IV pair can be considered as a weak pair.

Reverse Direction

Analysis in the reverse direction, either a loaded state or initial state is specified with one or more registers containing all-zero values. Then the task is to perform reverse clocking for the total number of iterations to obtain the candidate key and IV. If the candidate key and IV are correct, the loaded state format is obtained. Then the key-IV can be considered as a weak pair.

The theoretical analysis of weak key-IV combination results in the identification of the relationship between secret key bits and IV bits that generate one or more components of a cipher containing all-zero values. Based on this relationship between the key and IV bits, it is possible in some ciphers to distinguish keystreams produced from weak key-IV pairs and can be attacked. This analysis leads to investigate the consequences of weak key-IV pairs as discussed later.

Experimental Work

As discussed for the previous flaws, the experimental work requires computer simulation related to both forward and reverse clocking direction. This can be performed using a number of random inputs, or a scaled down version of the real cipher. Such simulations will result in the estimation of the probability of weak key-IV pairs that result in one or more registers containing all-zero values.

Consequences of Weak Key-IV Pairs

Any change in the properties of the keystream may leave the keystream vulnerable to some attacks, such as guess and determine attacks. This keystream is considered a weak keystream sequence. Weak key-IV pairs result in an initial state that produces a weak keystream sequence: one that is distinguished easily, it can be attacked with low complexity. This may reveal some or all of the secret key bits.

2.8 Existing Attacks on Initialisation Processes

This section presents some existing analysis of the initialisation processes stream ciphers described in Section 2.4. Some of these analyses or flaws are used with other generic attacks such as Time-Memory-Data Trade-off attacks. This section presents some examples that have been reported in the literature. Some of these examples show serious security attacks and flaws, and others demonstrate an undesirable feature in the initialisation process, but cannot be considered as serious attacks.

2.8.1 RC4 Analysis

Mantin and Shamir [79] discussed some statistical weaknesses of RC4 using the second output byte of the RC4 system. The second output word of RC4 is biased to have a value 0 with an approximate probability of $N = 2^{n-1}$ and the probability is 1/128 for $n = 8$. Denoting the first permutation or round during the pseudo random generator algorithm (PRGA) of $S[x]$ as $S_0[x]$, then they show that if the $S_0[2] = 0$ and $S_0[1] \neq 2$, then the second output byte is 0 with a probability of 1. The authors advise that, for future implementation of RC4 the first two output words be discarded and they recommend discarding the first N output words for more security. Later, Klein proved that an attack will work even if the first 256 bytes of the secret key stream are not used [72].

Fluhrer et al. [52] presented several weaknesses in the key scheduling algorithm (KSA) of RC4. From a small known number of secret key bits, a large number of states and output bits can be recovered. Therefore, RC4 has an unacceptable property in that these weak secret keys affect a large number of the output bits. Furthermore, a high correlation between some bits of the secret key and some bits of the output stream illustrates the propagation of a weak part of the key to some part of the output.

Another weakness related to the *initialisation processes* of some applications of RC4 [52] is when a secret key and initialisation vector (IV) are concatenated and used as a secret key for the RC4 stream cipher in a wired equivalent privacy (WEP) application. If an attacker can obtain the first word output corresponding to each IV, then the attacker can construct the secret key. Finally, Fluhrer et al. [52] stated that the RC4 is completely broken and it is not secure in most operation modes.

Based on the weaknesses that have been reported by Fluhrer et al. [52], Stubblefield et al. [99] tried to capture real encrypted packets to apply the attack. They claimed that they were able to recover the 128-bit secret key of RC4 that is used to protect a network (WEP application). Based on [52], Mantin [78] achieves a practical recovery attack of secret key and IV on RC4 stream cipher using fault injection, where the IV is concatenated with the secret key to generate the keystream for WEP application.

2.8.2 Trivium Analysis

For Trivium, the same state update function is used during both the initialisation and keystream generation processes. This function is invertible. Therefore, if an attacker can obtain the internal state at any time point, then the state update function can be inverted and the generator clocked back until the loaded state is obtained. The secret key can be directly extracted from the loaded state. This property implies that the state recovery attacks on Trivium are also key recovery attacks. As De Canniere and Preneel [43] report, there is not currently an attack which obtains the internal state from the keystream.

Hojsík and Rudolf [63] applied differential fault analysis to the Trivium stream cipher. Based on the chosen-chiphertext attack scenario, they supposed that an attacker can alter a random bit in the internal state to observe the difference in the keystream. Each bit fault in the internal state reveals a number of equations that can be solved. They showed that an injection of 43 bits in random position of the internal state can disclose the content of the internal state, and then the secret key can be obtained, as outlined above.

In an improvement to the previous work, Hojsík and Rudolf [64] applied the floating model for Trivium to transform the polynomial equations to linear equations. They found that an attack requires 3.2 one-bit fault injections on average to recover the internal state using 800 keystream bits. In the best scenario, 2 one-bit fault injections are enough to reconstruct the internal state, and consequently to obtain the secret key. They claim that if they inject 5 bits, they can reveal the internal state with a probability of success equal to one.

Priemuth-Schmid and Biryukov [86] investigated slid pairs for Trivium, and found that for a key-IV pair, (K, IV), there exists a related (K', IV') pair which generates a 111-bit shifted keystream, for more than 2^{39} key-IV pairs. They found slid pairs and solved the system of equations for 111, 112, 113, 114 and

115-bit shifted keystreams. (The Trivium padding pattern used in the loading phase ensures a slid pair may not occur until after 111 clocks.) To find the slid pairs, they generated a system of equations in terms of the key and IV variables. This system of equations is solved by guessing some variables and then checking the solutions.

2.8.3 Grain Analysis

Kuçuk [73] demonstrated a weakness of the Grain stream cipher by applying a slide attack on Grain 1.0 (Grain with secret key size 80) and called it a slide resynchronization attack. For any key-IV pair (K, IV) there exists a related (K', IV$'$) pair which generates a 1-bit shifted keystream with probability of 2^{-2}. The similarities of the state update functions of initialisation and keystream generation processes showed *a weakness in the initialisation process.* As Kuçuk mentioned, this attack does not achieve an efficient key recovery attack for Grain 1.0; it may need more research to recover the secret key. The attack is based on the construction of Grain 1.0 as follows. Firstly, the initialisation process and keystream generator are similar to each other. Secondly, the initialisation process is similar to the keystream generation, except the output of the initialisation process is fed back and XORed to the input of both the LFSR and NFSR. This paper shows that the internal state of the LFSR and NFSR after one clock will be equal to the previous states with a shift of one bit for both the LFSR and NFSR if $b_0^{160} \oplus h(x) = 0$, where b_0^{160} is content of b_0 of the NFSR after 160 clocking steps of the initialisation process and $h(x)$ is the value of the output function at the same clocking steps (160) .

Lee et al [76] extended the work of Kuçuk [73] to attack Grain v1 and Grain-128. This attack is performed for secret key recovery with $2^{22.59}$ chosen IVs, $2^{26.29}$ keystream and complexity of $2^{22.90}$ for Grain v1. For Grain-128, the secret key recovery attack requires $2^{26.59}$ chosen IVs, $2^{31.39}$ keystream and complexity of $2^{27.01}$.

De Cannière et al [42] reported a general form of the probability of slid pairs for Grain stream cipher for a key-IV pair, (K, IV), there exists a related (K', IV$'$) pair which generates an n-bit shifted keystream with probability of 2^{-2n}. They analysed Grain 1.0 using a generic differential attack. This analysis recovers one key in 2^9 keys using two related keys and 2^{55} chosen IV pairs.

Zhang and Wang [108] demonstrated some sliding weak key-IV combination that apply to Grain v0 [58], Grain v1 [59] and Grain-128 [60]. They define weak key-IVs, that result in zeros for all the stages of the LFSR, during the initialisation process. A reverse direction is used during the initialisation process to examine the existence of weak key-IVs. The process starts from the initial state (with LFSR contains zeros) and clocks backward using a specific algorithm (it is called Algorithm 1) to find the loaded state at time $t = 0$. They used the weak key-IVs to perform a distinguishing attack using $2^{64}, 2^{64}$ and 2^{96} weak key-IVs, with $2^{12.6}, 2^{44.2}$ and 2^{86} keystream bits, and $2^{15.8}, 2^{47.5}$ and $2^{104.2}$ operation, respectively for Grain v0, Grain v1 and Grain-128. Then, they apply algebraic attack with weak key IVs to recover the secret key for all of Grain v0, Grain v1 and Grain-128.

Bjørstad [27] uses Time-Memory/Data Trade-Off (TMTO) to attack Grain with an internal state of 160 bits and secret key size of 80 bits. Due to this choice of state size and key size, it is infeasible to apply the standard time-memory trade-off directly to Grain v1. Firstly, Grain has a large complexity resistance against TMTO and the computation complexity is not lower than the exhaustive search $O(2^{80})$. Bjørstad incorporates the advantages of both the Biryukov, Shamir and Wagner (BSW) [24] sampling and the properties of output function $h(\cdot)$. The nonlinear function $h(\cdot)$ is computed and masked with seven bits from the NFSR. The paper utilizes the distance between the two masking bits of the NFSR, which are n_{i+10} and n_{i+31}, to derive that the sampling resistance of Grain is 2^{-18}. It shows that the TMTO recovers the state of Grain v1 in online time and memory complexity $O(2^{71})$, precomputation complexity $O(2^{106.5})$ and used known keystream complexity $O(2^{53.5})$. These results show the precomputation complexity is greater than the exhaustive secret key search complexity; however, the complexity of the online computation is less than the complexity of the exhaustive search.

Banik et al [10] reported some results on the related key-IV pairs (slid pairs) on Grain v1 [59], Grain-128 [60] and Grain-128a [1]. They reported an algorithm to find a related key-IV pair (K', IV$'$) given a key-IV pair (K, IV).

There are two recent papers: the first paper by Ding and Guan [45] has shown a related key attack on Grain-128a [1]. Their result requires a complexity of $2^{96.322}$, 2^{96} chosen IVs and $2^{103.613}$ keystream bits. As they reported, the probability of success of this attack is 0.632. In a second paper by Banik et al [11] demonstrated slid pairs in Grain-128a, which may occur after 32 clocks, which is expected, where the padding is all ones except one bit is zero. They presented a key recovery attack with $\gamma \cdot 2^{32}$ related keys and $\gamma \cdot 2^{64}$ chosen IVs to generate $32 \cdot \gamma$ nonlinear equations, where γ can be estimated to be 2^8.

2.8.4 Dragon Analysis

The initialisation process and keystream generation use the F function with S-boxes, H and G functions for ease of implementation, efficiency and high nonlinearity. Dragon has an internal state space of 1088 bits (which is the combination of 1024-bit internal state and 64-bit memory) for high security [32, 39]. The expected upper bound of the period is $2^{512+64} = 2^{576}$ and the lower bound is 2^{64}. The maximum interval of the keystream before rekeying is 2^{64}. Therefore, Dragon's design prevents collision attacks [32, 39] as mentioned by the authors.

From the F function and its components, during the initialisation process, four words from the output of the F function updates the internal state, and two words form the memory value. During keystream generation, two words are used to produce a 64-bit word of keystream. The F function using H and G functions with 16 iterations gives high diffusion and prevents high probability of differentials. [32, 39].

The size of the internal state is larger than double the secret key size; hence, the TMTO is infeasible [49]. Dragon has been designed with consideration given to the initialisation process. It uses a secret key with different IVs producing different keystreams using 16 rounds and the nonlinear function F.

Since the F function is used during the initialisation and keystream generation processes, there is a number of studies that tried to linearise the F function. These studies tried to distinguish the Dragon stream cipher [33, 34, 49]. This is the only analysis that is related to the initialisation process of Dragon stream cipher. It is an attractive area for future work.

2.8.5 MICKEY Analysis

Hong and Kim reported a state entropy loss in the MICKEY v1 stream cipher which is state convergence [65]. They report that the state update function is not one-to-one and experimented with 2^{20} randomly chosen states (which is based on the C-language `rand()`) to find only about 70.63% have pre-images after one clock. So, after one clock, there may be zero, one, two or four pre-images for any state. They did the examination for two consecutive clocks and they found that about 56.17% have pre-image after two clocks.

Helleseth et al [61] continued the work of Hong and Kim [65] on MICKEY v2 [8] and MICKEY-128 v2 [7] and examined the existence of slid pairs in MICKEY v2. They performed reverse clocking to MICKEY v2 and MICKEY-128 v2 to calculate the number of backward states for 0 to 15 clocks. They demonstrated two key-IV pairs that generate shifted keystreams (shifted by 1 bit). Based on this analysis, they presented two scenarios of attack based on some assumptions. For more details see [61].

2.8.6 LILI-II Analysis

Biham and Dunkelman [20] reported minor multiple Key-IV pairs result in one loaded state during the loading phase in the initialisation process (which is Key-IV compression). The operation of the loading phase is only XORing the secret key and the IV as specified previously to produce the loaded state of the two registers LFSR_d and LFSR_c, which is pure linear function. Based on this loading process, Biham and Dunkelman [20] noted that if two key-IV pairs (K, IV) and (K', IV') that satisfy the relationship $K \oplus K' = \text{IV} \oplus \text{IV}' = \{1\}^{128}$, then these two key-IV pairs will produce the same loaded state for registers LFSR_d and LFSR_c and consequently will result in the same keystream. Recently, Teo [100] reported that state convergence does not exist in the LILI family of stream ciphers.

2.9 Tools

This section describes the tools used to implement the stream ciphers and to investigate the flaws discussed in Section 2.7. In this thesis both the C programming language and MAGMA software package are used as tools for computer simulation and exploration.

C Programming Language

C programming or C-code is a computer language that is used to implement many stream ciphers, such as the ciphers that were submitted to eSTREAM project [50] and NESSIE project [68]. In this research, we make use of the software implementations published by the authors of these ciphers. Additional software in C has been created for simulation and analysis to identify and demonstrate particular flaws.

Analysis of the flaws may require specific inputs to the stream cipher, or to some components, to study the impact on the internal state or keystream sequence. Therefore, we have adapted some codes to enable examination of specific key-IV pairs or internal states. This adaptation does not change the main function or purpose of the stream cipher algorithm. Computer simulation is also used for statistical analyses as a technique to study or monitor the behaviour of the analysed ciphers under varying conditions.

MAGMA Package

Another tool that is used in this research is a computer algebra system known as MAGMA [28]. MAGMA is a powerful tool for implementing the ciphers, generating systems of equations and solving the system of equations. As part of the process of investigation, the computer simulations using the MAGMA computational algebra system require the support of the High Performance Computing (HPC) facility at QUT due to the resources required to perform the computer simulations.

2.10 Summary

Recently, specification of the initialisation process has became an essential part of any stream cipher design. During the eSTREAM project, one of the requirements for stream cipher submission is that the use of the IV as essential input to the cipher [50]. In contrast, there are some traditional stream ciphers that include initial value (IV) and initialisation process as part of the keystream generator such as A5/1 and CSA-SC stream ciphers.

Initialisation process can introduce significant problems in stream cipher applications. This process may leak some information or be exploited by attackers. It may make stream ciphers vulnerable by revealing some information about the secret key. From the literature review, it is clear that the initialisation process of stream ciphers have not yet been thoroughly studied. It is desirable to have a secure and efficient initialisation process that resist at least the known attacks. The initialisation process should possess desirable properties that make them simultaneously secure and efficient.

As mentioned above, there are three stream ciphers for which the initialisation process will be discussed, and analysed in later chapters, as follows: Chapter 3 investigates the initialisation process of A5/1 stream cipher. Chapter 4 analyses the initialisation process of Sfinks stream cipher. Chapter 5 considers the initialisation process of the Common Scrambling Algorithm Stream Cipher (CSA-SC). Following that, Chapter 6 provides recommendations for the initialisation process based on the three cases and previous analyses of initialisation processes of stream ciphers.

Chapter 3

Analysis of A5/1 Stream Cipher Initialisation Process

The privacy of mobile telephone communications is protected by the A5/1 stream cipher. A5/1 is used by about 130 million customers in Europe. A general structure of A5/1 was leaked in 1994 and the exact design was revealed in 1999, when it was reverse engineered by Briceno [30]. Therefore, A5/1 has received the attention of cryptographers and has been analysed in several papers such as [12, 19, 24, 53–55] but mainly looking at keystream generation rather than the initialisation process.

A5/1 stream cipher is a bit-based cipher that takes a 64-bit secret key and 22-bit IV (frame number) as inputs into a 64-bit internal state (in three shift registers). Each telephone conversation uses one secret key with multiple IVs. Each IV is used to generate a 228-bit keystream. Then the initialisation process (rekeying) is repeated with another IV to generate another 228-bit keystream sequence. The telephone converstion is sent as a sequence of frames every 4.6 milliseconds. Each frame (228-bit keystream) represents the communication between two parties A and B, so each keystream frame consistes of 114 bits to communicate from A to B and another 114 bits to communicate from B to A.

In Section 2.7, some aspects of the security flaws of initialisation process for stream ciphers have been demonstrated theoretically. This chapter investigates these flaws in A5/1; a cipher which uses a linear state update function during loading phase and the same nonlinear state update function for both diffusion phase and keystream generation. The output function as well is a purely linear function, namely the XOR function. The nonlinearity of the keystream is derived from the nonlinear state update function during the diffusion phase and keystream generation.

The initialisation process of A5/1 [30] is analysed with respect to the occurrence of the following problems: State convergence, slid pairs and weak key-IV pairs. The results of these investigations are presented. Although some of these problems also occur during keystream generation, the keystream generation process is beyond the scope of this research.

This chapter is organized as follows. Section 3.1 describes the specification of A5/1 and its initialisation process. Section 3.2 presents the analysis of the state convergence during the initialisation process of A5/1. Section 3.3 shows the slid pairs and synchronisation attacks in the A5/1 and presents results and examples to support the theoretical analysis. Weak key-IV pairs are described with details in Section 3.4. This chapter is summarised in Section 3.5 and presents the security impact of this analysis.

3.1 Specification of A5/1 Stream Cipher

A5/1 [24, 30, 54] is a bit based stream cipher which uses three binary feedback shift registers, denoted A, B and C, with lengths of 19, 22 and 23 bits respectively, giving a state size of 64 bits. Each shift register has a primitive feedback polynomial. Let S denote the internal state of A5/1 and let S_A, S_B and S_C denote the internal states for each register A, B and C, respectively. Let $s^i_{a,t}$ denote the content of i^{th} stage of register A at time t, for $0 \leq i \leq 18$, $s^i_{b,t}$ denote the content of i^{th} stage of register B at time t, for $0 \leq i \leq 21$ and $s^i_{c,t}$ denote the content of i^{th} stage of register C at time t, for $0 \leq i \leq 22$.

A secret key of 64 bits is used for all frames in a given conversation, and a 22-bit frame number is used as the IV for each frame. Let k_i denote the secret key bit for $0 \leq i \leq 63$ and v_i denote the IV bit for $0 \leq i \leq 21$. The three registers are regularly clocked during loading of the key and IV (frame number), while a majority clocking mechanism is used for the diffusion phase and for keystream generation. The use of majority clocking implicitly introduces nonlinearity to the keystream sequence. This is the only nonlinear operation performed.

To implement the majority clocking scheme, each register has a clocking tap:

stages $s^{8}_{a,t}$, $s^{10}_{b,t}$ and $s^{10}_{c,t}$. The contents of these stages determine which registers will be clocked for the next iteration at time $(t+1)$: those registers for which the clock control bits agree with the majority value are clocked. For example, if $s^{8}_{a,t}=0$, $s^{10}_{b,t}=1$ and $s^{10}_{c,t}=0$, then the majority value is 0 and registers A and C are clocked at time $(t+1)$. Thus, either two or three registers are clocked at each step. Overall, each register is clocked with probability of $\frac{3}{4}$. Figure 3.1 shows a pictorial diagram of the A5/1 stream cipher, with the clocking taps and feedback functions indicated for each register.

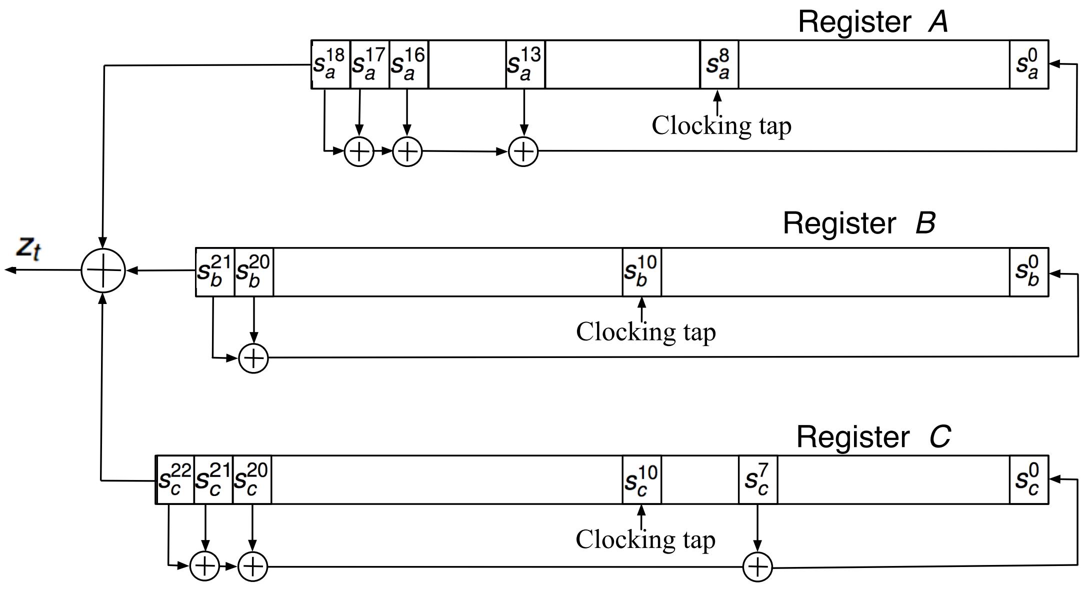


Figure 3.1: A5/1 stream cipher

3.1.1 Initialisation Process

The initialisation process transfers the 64-bit secret key and 22-bit frame number (IV) into the internal state and performs 100 iterations to produce the 64-bit initial register state. Once this initial state is obtained, keystream generation can begin. The initialisation process is performed in two phases, which we refer to as *loading* and *diffusion* phases.

Loading Phase

At the beginning, all stages of the three registers are set to zero. Each linear feedback shift register is regularly clocked 64 times as each key bit, k_i, is XORed with the register feedback to form the new value of stages s_a^0, s_b^0 and s_c^0. Following this, each register is regularly clocked 22 times as the IV is loaded in the same manner [24]. At the end of the loading phase, the registers are in the *loaded state*. Note that the state update function during the loading phase is entirely linear. That is, the contents of each stage in each register are independent linear combinations of key and IV bits.

Diffusion Phase

The diffusion phase involves performing 100 iterations of the initialisation state update function using the majority clocking scheme. At the end of diffusion phase the cipher is in its *initial state* and is ready for keystream generation.

3.1.2 Keystream Generation

Keystream generation comprises 228 iterations using the same majority clocking rule used during the diffusion phase. In each iteration, the keystream bit is obtained by XORing the output bit of the three registers $z_t = s_{a,t}^{18} \oplus s_{b,t}^{21} \oplus s_{c,t}^{22}$.

The analysis of the A5/1 stream cipher is based on the behaviour of the majority clocking schema. The following sections show three cryptographic flaws in the initialisation process of the A5/1.

3.2 State Convergence

The A5/1 stream cipher has a 64-bit internal state and uses a 64-bit secret key and a 22-bit IV to form a loaded state. So, there are 2^{64} possible internal states. Since the loading phase is linear, it is clear that each loaded state can be obtained from multiple key-IV pairs.

As the total size of the secret key and IV for A5/1 ($64+22 = 86$ bits) exceeds the 64 bit state size, a degree of key-IV space reduction occurs during the loading phase of initialisation. In fact, as the state-update function is linear during the

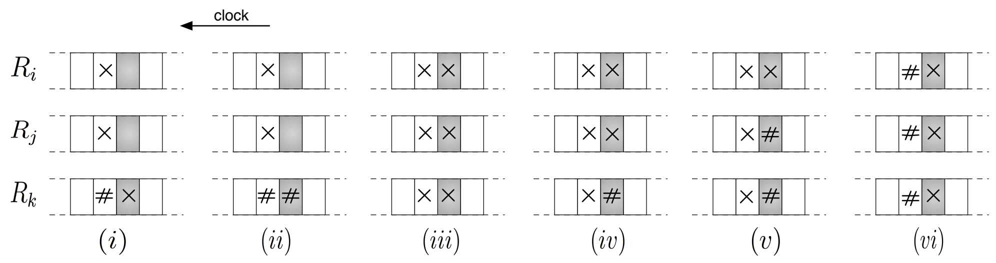


Figure 3.2: A5/1 pre-image of Golić's cases

loading phase, it can be shown that there are 2^{22} key-IV pairs corresponding to each possible loaded state. In addition to this issue, state convergence also occurs during the diffusion phase and keystream generation process.

3.2.1 Previous Analysis

Few previous analyses of A5/1 focussed specifically on the effect of state convergence during initialisation. Two papers that deal with this topic as part of a broader analysis are Golić [55] (based on [54]) and Biryukov, Shamir and Wagner [24].

Golić [55] considered the inverse mapping for the majority clocking function and identified some states with no pre-image and which therefore cannot be reached from any loaded state in a single iteration. He demonstrated that these states comprise $\frac{3}{8}$ of the loaded states of the system. Thus, the usable state space shrinks by a factor of $\frac{5}{8}$ (from 2^{64} to $5 \times 2^{61} \approx 2^{63.32}$) at the first iteration of the diffusion phase. Golić also identified some states with unique pre-images and others with up to four pre-image states. Figure 3.2 presents a graphical summary of the six cases identified by Golić. In this figure, (R_i, R_j, R_k) is any permutation of the set $\{A, B, C\}$ of registers and the shaded stage in each register is its clocking tap. The symbol $\times$ represents either 0 or 1, while $\#$ represents the complement of $\times$; a blank square represents a bit which can take either value.

The proportion of loaded states for each case in Figure 3.2 is presented in Table 3.1, along with the corresponding number of pre-images. Note that the case identified as (i) cannot be clocked back to any valid state. That is, states of this form cannot be reached after the first iteration of the initialisation state update function.

Biryukov, Shamir and Wagner [24] also provide convergence estimates when exploring the efficiency of their attack. They report that, of 10^8 randomly chosen

Table 3.1: Proportions of states in each of Golić's cases

Case	*(i)*	*(ii)*	*(iii)*	*(iv)*	*(v)*	*(vi)*
Proportion of states	$\frac{3}{8}$	$\frac{3}{8}$	$\frac{1}{32}$	$\frac{3}{32}$	$\frac{3}{32}$	$\frac{1}{32}$
Number of pre-images	0	1	1	2	3	4

states, only about 15% can be clocked back 100 times. That is, 85% of states could not be reached by a 100 iteration forward clocking process.

Alhamdan [2] performed an exhaustive experimental evaluation for a scaled-down version of the A5/1 stream cipher with three LFSRs and a majority clocking arrangement, but only a 15-bit internal state (LFSR lengths of 4, 5 and 6 bits were used). This scaled-down version uses a 15-bit secret key and a 5-bit IV (as frame number) and is loaded in the same as manner A5/1 with majority clocking performed for 100 iterations during the diffusion phase. All possible loaded states were used and the number of remaining distinct states for each iteration in the diffusion phase was recorded. The first three lines in Table 3.2 show the summary of the three previous works; Golić [55], Biryukov, Shamir and Wagner [24] (labelled as BSW) and Alhamdan [2]. The table also includes figure for two other analyses that are discussed in Section 3.2.2.

Table 3.2: Comparison between the analysis of inaccessible states

	Cumulative proportion of state reduction							
No of clock	1	2	3	4	5	6	10	100
Golić* [55]	0.375	-	-	-	-	-	-	-
BSW$^{\circ}$ [24]	-	-	-	-	-	-	-	0.85
Alhamdan$^{\bullet}$ [2]	0.375	0.422	0.439	-	0.466	-	0.524	0.81
KT* [71]	0.375	0.578	0.689	0.767	0.826	-	-	-
This work*	0.375	0.422	0.439	0.453	0.466	0.479	-	0.95$^{\diamond}$

* Theoretical analysis

$^{\circ}$ based on 10^8 randomly simulated states

$^{\bullet}$ Exhaustive search for a scaled-down version

$^{\diamond}$ based on exponential extrapolation

3.2.2 Extension of Existing Work

In A5/1, nonlinear operations in the state-update function are introduced during the diffusion phase via majority clocking. However, this majority clocking introduces state convergence. Although this convergence continues into keystream generation, in this research the focus is on the initialisation process only. As noted above, this effect was reported by Golić [54,55] and quantified to some extent by Biryukov, Shamir and Wagner [24]. This research extends these results to a larger number of iterations.

Golić's results demonstrate that the majority clocking process is not one-to-one and that state convergence can occur in one iteration. This work extends Golić's logic to identify the states which cannot be reached after each of the first six iterations of the diffusion phase. It shows that state convergence continues with each iteration, though not uniformly at each iteration, contrary to Golić's assumptions [55]. Some of the inaccessible states we identified for multiple iterations are presented in Figure 3.3. This work will use the previous notations (R_i, R_j, R_k) is any permutation of the set $\{A, B, C\}$ of registers in Figure 3.1 and the shaded stage in each register is its clocking tap. The symbol $\times$ represents either 0 or 1, while # represents the complement of $\times$; a blank square represents a bit which can take either value.

Now, the reasoning used to identify states that are inaccessible after two iterations is sketched. The term "downstream" is referred to the stages in Figure 3.2 and 3.3 that are to the left of the clocking stages. By reversing the logic of the majority clocking process, the following conditions apply when we invert an iteration:

1. A state obtained by clocking a pair of registers must have the contents of the stages immediately downstream of the clocking bit in these registers identical in value to one another, and different in value from the clocking bit of the third register.

2. A state obtained by clocking all three registers must have the contents of the stages immediately downstream of the clocking tap identical in all three registers.

For Figure 3.2, we note that Condition 1 applies to Case (ii), Condition 2 applies to Case (iii), both conditions apply to cases (iv), (v) and (vi), but neither applies

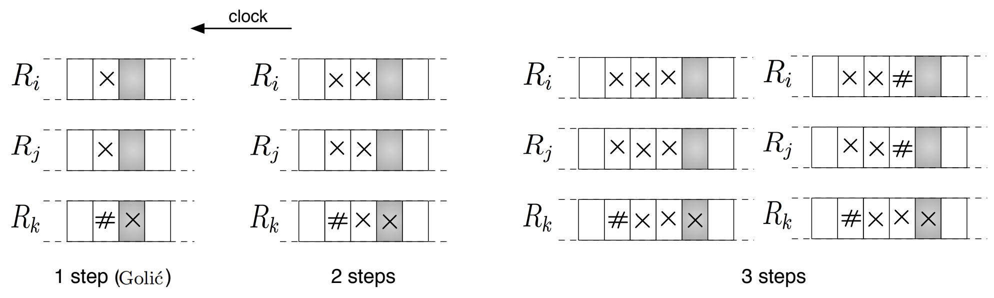


Figure 3.3: Inaccessible states for various numbers of iterations (steps)

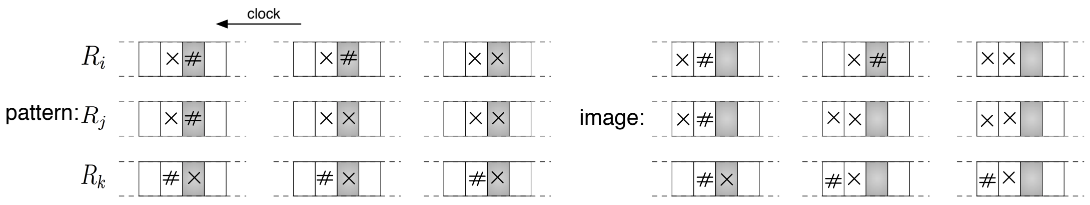
to Case (i). In cases (iv), (v) and (vi), Condition 1 applies to different numbers of the three possible pairs of registers.

Applying this logic to the pattern labelled “2 steps” in Figure 3.3 shows that such a state can arise only by clocking a combination of registers that includes register R_k in Figure 3.3. But this implies that any previous state belongs to Case (i) of Figure 3.2 (possibly with additional values specified among the clocking bits). Since Case (i) cannot be reached by the first clocking step, this “2 steps” state cannot be reached at the subsequent clocking step. (Note: it can, however, be reached by the first clocking step, since Case (i) is a valid loaded state.)

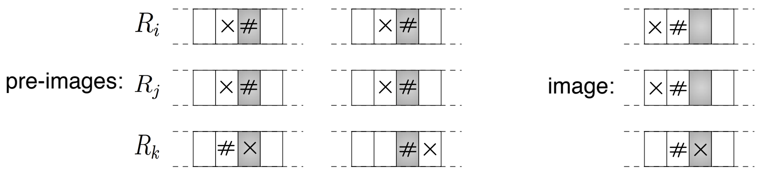
This pattern is the only inaccessible pattern at this step. Any state which is inaccessible after two iterations must clock back only to states that were inaccessible after the first step. So all such states must be contained in the image space (under clocking) of Case (i) above. This image space can be found by completing the unspecified values in Case (i) in all possible ways and applying the clocking rule to each (see Figure 3.4a). When this is done, we find that many of the image states are accessible, as they have multiple pre-images, some of which are accessible (see Figure 3.4b for an example). If we discard these states and retain those which can clock back only to Case (i), we find that the pattern presented above is indeed the only new inaccessible pattern at the second step.

A similar process can be followed to identify inaccessible patterns after α iterations. There is a branching tree of patterns for these inaccessible states: as well as the two “3 step” patterns presented in Figure 3.3, there are five distinct patterns at the fourth iteration, 17 at the fifth iteration and many more at each subsequent iteration. Table 3.3 presents the cumulative proportion of inaccessible states (out of all possible loaded states) after each of the first six iterations, together with the corresponding proportion and number of accessible states.

The process that is used to identify inaccessible patterns for $\alpha = 2$ iterations



(a) Results of clocking Case (i) forwards



(b) Possible pre-images for one of these results

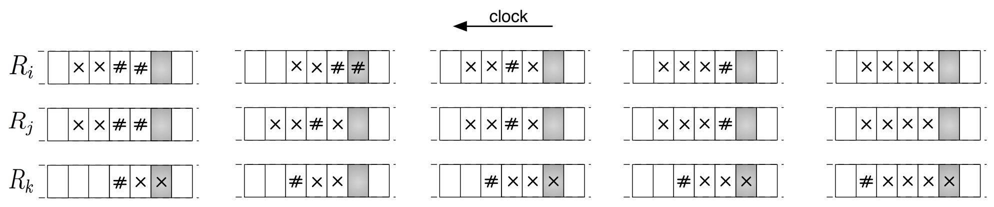
Figure 3.4: Determining inaccessible states at the second step

Table 3.3: Proportion of available states after α iterations

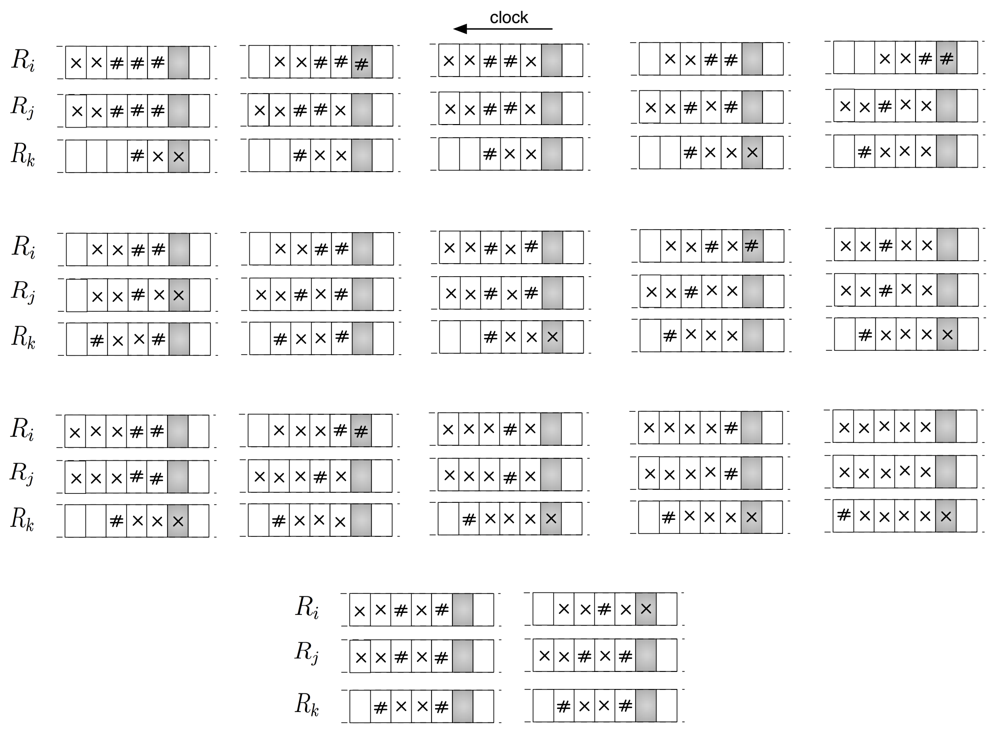
α (number of iterations)	1	2	3	4	5	6
new proportion inaccessible	$\frac{3}{8}$	$\frac{3}{64}$	$\frac{9}{512}$	$\frac{57}{4096}$	$\frac{423}{32768}$	$\frac{6453}{524288}$
cumulative proportion inaccessible	0.375	0.422	0.439	0.453	0.466	0.479
proportion accessible	0.625	0.578	0.561	0.547	0.534	0.521
number of accessible states	$2^{63.322}$	$2^{63.209}$	$2^{63.165}$	$2^{63.129}$	$2^{63.094}$	$2^{63.061}$

is to clock the inaccessible pattern (i) in Figure 3.2 by one clock forward. This will result in four states, then examine these four states for which one cannot be obtained from another accessible pre-image. The state that has only one pre-image ended with pattern (i) of Figure 3.2 will be inaccessible state at this clock $\alpha = 2$. This process is applied for any number of α to determine the inaccessible patterns. Figures 3.5a and 3.5b show the inaccessible patterns for $\alpha = 4$ and 5 respectively.

The number and complexity of the patterns obtained so far indicates that obtaining a general expression for the number of accessible states after a given number of iterations is not a simple task for large values. Extrapolating from the known values in Table 3.3 provides an approximation. Using an exponential extrapolation based on the proportion of accessible states as reported above for 2–6 iterations, we obtain an approximation of the proportion of accessible



(a) For $\alpha = 4$



(b) For $\alpha = 5$

Figure 3.5: Inaccessible patterns

states after 100 iterations of around 5% of the number of loaded states. The extrapolation is based on a linear regression fit to the logarithm of the proportion accessible [4]. Figure 3.6 presents the exponential extrapolation for the 100 steps and the actual results for 0–6.

As a result of the extended work, for small numbers of iterations, Alhamdan's work for scaled-down version of A5/1 [2] aligns very closely with those reported for this work on the acutal A5/1 cipher. Table 3.2 shows a summary of the previous works and the current work. As reported by Alhamdan [2], the proportion of distinct states after 100 iterations is 19.2% of the original loaded state. This is close to the experimental result of Biryukov, Shamir and Wagner's which is 15% [24]. Our extrapolation based on the results in Table 3.3 for the proportion

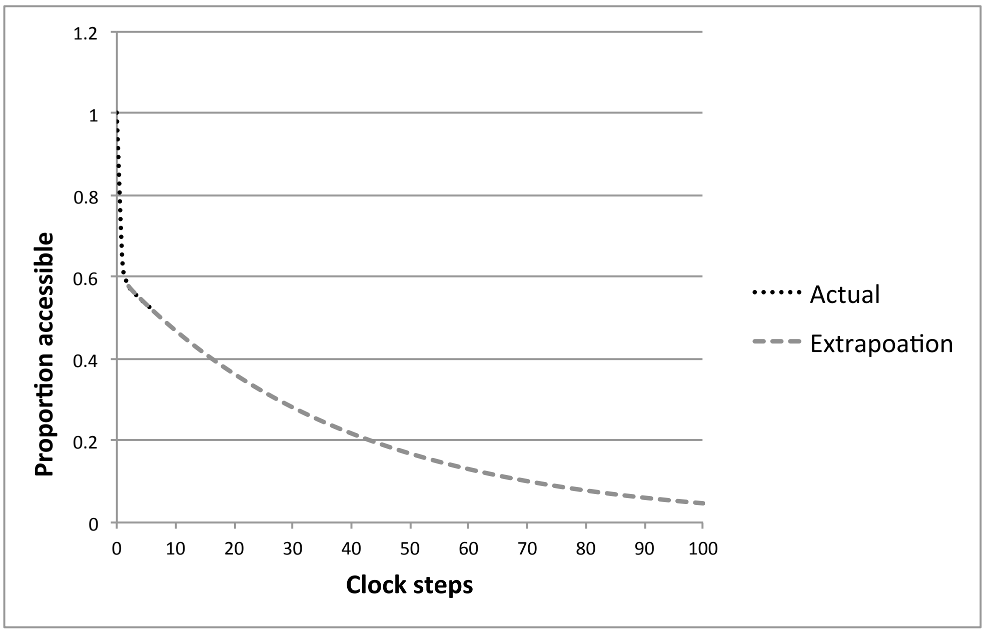


Figure 3.6: Exponential extrapolation for the proportion of accessible states after 100 steps

of accessible states is 5%.

Another Recent Work

A recent paper by Kiselev and Tokareva [71] analysed the reduction of key space during the state-update function of A5/1 for additional clocks. They reported on the reduction of the key space over up to 8 clocks. They tried to extend Golić's [55] work to determine the effective key space reduction in each of the first 8 clocks. Their results are also reported in Table 3.2 (labelled as KT). Their results for the number of inaccessible states after the first clock are consistent with previously reported results, but the results for further clock steps do not match the results presented above. Their results of further steps overestimate the number of inaccessible states compared to the available results. This discrepancy arises because these authors have assumed that any state accessible from the first inaccessible state is also inaccessible, whereas many of these states can actually be reached by clocking from other accessible states as well from the inaccessible states. Thus, these authors have included many accessible states in their claimed list of inaccessible states, for each clock step beyond the first one. This is demonstrated in Figure 3.7, where state (a) is a sub-case of one of Kiselev-Tokareva templates (the top left template of their Figure 4). This state can be obtained by clocking from the inaccessible state (b), as would be expected from Kiselev-Tokareva construction. However, it can also be obtained by clocking from the accessible state (c) (which corresponds to Figure 3.2 (v)). Thus, state (a) is clearly accessible after 2 clocks and should not be counted in their estimate of the number of inaccessible states after 2 clocks.

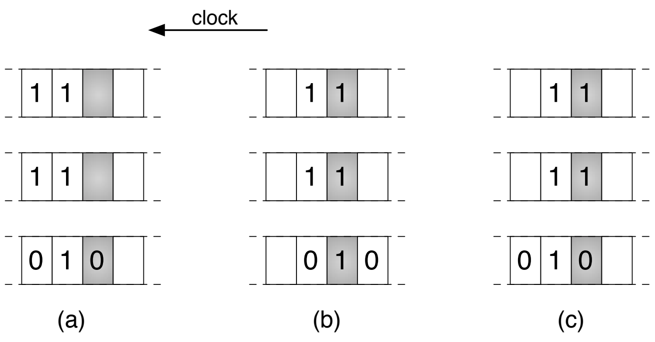


Figure 3.7: Example of Kiselev and Tokareva's template

3.3 Slid Pairs and Synchronisation Attacks

For stream ciphers, it is sometimes possible to find different key-IV pairs that produce phase shifted keystream [42, 86, 108]. This is clearly the case for A5/1. As the number of possible internal states is the same as the number of loaded states, it is clear that any internal state obtained after any number of iterations α, is also a legitimate loaded state. That is, for any key-IV pair and any $\alpha > 0$, it is always possible to find a second key-IV pair such that the loaded state from the second pair can be obtained from the loaded state of the first pair by applying α iterations of the (diffusion) state update function. That is, the loaded states corresponding to these two key-IV pairs from a slid pair separated by α clocks. Further, since the update functions during diffusion and keystream generation of A5/1 are identical, this slid pair will always produce keystream sequences which are out of phase by α bits.

Recall the operation of GSM system and A5/1 stream cipher, which use a single secret key and up to 2^{22} different IV's for each conversation. The initialisation process (rekeying process) is performed every 4.6 milliseconds with each initial state then used to generate a 228-bit keystream. As noted in Section 2.7.3, slid pairs can occur in three ways; the same secret key with distinct IVs, for distinct secret keys with distinct IVs, and for distinct secret keys with the same IV. For a practical attack, the most important case is that with the same secret key and multiple IVs, as this applies for a single conversation. Therefore, we restrict our analysis to that; we look for slid pairs in which two loaded states were generated from the same secret key but necessarily with different IV's. From analysis of this case, we show that it may be possible to perform a practical attack on A5/1 to determine the secret key used in encrypting the targeted conversation.

3.3.1 Analysis of Slid Pairs

During the Loading Phase

As the operation of the LFSRs during the loading phase is linear, it can easily be represented in terms of matrix operations . An analysis of the matrices involved then enables us to identify the conditions under which a slid pair can occur.

The autonomous operation of each LFSR can be described in terms of a matrix equation [77] as follows. Suppose that the stages of the register are denoted as $s^0, s^1, \ldots s^d$, the update coefficient are $c_0, c_1, \ldots c_d$ and that the update function can be represented as

$$\begin{aligned} s^0_{t+1} &= c_0 s^0_t \oplus c_1 s^1_t \oplus \ldots c_d s^d_t \\ s^j_{t+1} &= s^{j-1}_t \quad 1 \leq j \leq d \end{aligned}$$

Putting $S = [s^0 \ s^1 \ldots s^d]^\intercal$, the register update (clocking) operation can be represented by the equivalent matrix equation

$$S_{t+1} = TS_t \qquad \text{where } T = \begin{bmatrix} c_0 & c_1 & \ldots\ldots & c_{d-1} & c_d \\ 1 & 0 & \ldots\ldots & 0 & 0 \\ 0 & 1 & \ldots\ldots & 0 & 0 \\ \vdots & \vdots & \ddots \, \ddots & \vdots & \vdots \\ 0 & 0 & \ldots\ldots & 0 & 0 \\ 0 & 0 & \ldots\ldots & 1 & 0 \end{bmatrix}$$

is the state transition matrix of the register.

During the loading of the key into the register, the register state update function is given by

$$S_{t+1} = TS_t \oplus \sigma k_t \tag{3.1}$$

where $\sigma = [1\,0\ldots0]^\intercal$ indicates that the new key bit is XORed into the feedback of the LFSR. If we take $t = \tau$ to indicate the register state before loading commences and iterate this process several times, from $t = \tau$ to $t = \tau + l$ (where l denotes the key length), we have

$$\begin{aligned}
S_{\tau+1} &= TS_\tau \oplus \sigma k_0 \\
S_{\tau+2} &= T(TS_\tau \oplus \sigma k_0) \oplus \sigma k_1 = T^2 S_\tau \oplus T\sigma k_0 \oplus \sigma k_1 \\
&\ \ \vdots \\
S_{\tau+l} &= T^l S_\tau \oplus T^{l-1}\sigma k_0 \oplus T^{l-2}\sigma k_1 \oplus \ldots \oplus T\sigma k_{l-2} \\
&\quad \oplus \sigma k_{l-1} \\
&= T^l S_\tau \oplus NK
\end{aligned}$$

where $N = [T^{l-1}\sigma \quad T^{l-2}\sigma \ldots T\sigma \quad \sigma]$ and $K = [k_0 \quad k_1 \ldots k_{l-2} \quad k_{l-1}]^\intercal$.

The above analysis can be extended easily to cases such as A5/1, which have three LFSRs, by denoting the states of the three registers as S_A, S_B and S_C, and their state transition matrices as T^A, T^B and T^C, and defining the state transition matrix of the combined system in matrix block form as

$$T = \begin{bmatrix} T^A & 0 & 0 \\ 0 & T^B & 0 \\ 0 & 0 & T^C \end{bmatrix} \text{ acting on } S = \begin{bmatrix} S_A \\ S_B \\ S_C \end{bmatrix}$$

Likewise, denoting the σ and N matrices of each register as σ^A, σ^B, σ^C and N^A, N^B, N^C, the combined σ and N matrices for the whole system can be defined as

$$\sigma = \begin{bmatrix} \sigma^A \\ \sigma^B \\ \sigma^C \end{bmatrix} \quad \text{and} \quad N = \begin{bmatrix} N^A \\ N^B \\ N^C \end{bmatrix}$$

With these modifications, the equation above, $S_{\tau+l} = T^l S_\tau \oplus NK$, is also valid for the combined system. Noting that $S_\tau = [0\,0\ldots0]^\intercal$ for the registers of A5/1, this equation reduces to $S_{\tau+l} = NK$.

A similar analysis can also be undertaken for loading the IV bits into the LFSRs. For a 64-bit key and a 22-bit IV, we have

$$\begin{aligned} S_{\tau+64} &= NK \\ S_{\tau+86} &= T^{22}NK \oplus MV \end{aligned} \tag{3.2}$$

where $M = [T^{21}\sigma \quad T^{20}\sigma \ldots T\sigma \quad \sigma]$ and $V = [v_0 \quad v_1 \ldots v_{20} \quad v_{21}]^{\mathsf{T}}$.

During the Diffusion Phase

Now set $\tau = -86$, so that S_0 represents the loaded state of the system, and consider the behaviour of A5/1 during the diffusion phase. During this phase, the registers of A5/1 are clocked using a majority clocking rule, so for a single iteration, there are four different cases to be considered. These are:

Case 1 All registers are clocked

Case 2 Registers A and B are clocked

Case 3 Registers A and C are clocked

Case 4 Registers B and C are clocked

For each of these cases, there will be a different state transition matrix (T_x say) for the system, as follows:

for Case 1: $T_x = T$, as defined previously

$$\text{for Case 2: } T_x = T_{ab} = \begin{bmatrix} T^A & 0 & 0 \\ 0 & T^B & 0 \\ 0 & 0 & I \end{bmatrix}$$

$$\text{for Case 3: } T_x = T_{ac} = \begin{bmatrix} T^A & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & T^C \end{bmatrix}$$

$$\text{for Case 4: } T_x = T_{bc} = \begin{bmatrix} I & 0 & 0 \\ 0 & T^B & 0 \\ 0 & 0 & T^C \end{bmatrix}$$

Now suppose that we are looking for slid pairs with a slide distance of $\alpha = 1$ in which both loaded states arise from the same secret key. We have

$$S_1 = T_x S_0 \quad \text{with} \quad S_0 = T^{22}NK \oplus MV$$

Now if S_1 is also a loaded state for the same key K and different IV, V' , we have

$$S_1 = T^{22}NK \oplus MV'$$

and hence

$$\begin{aligned} MV \oplus MV' &= S_0 \oplus S_1 \\ \text{or} \quad M\Delta &= (I \oplus T_x)S_0 \\ &= (I \oplus T_x)(T^{22}NK \oplus MV) \end{aligned} \tag{3.3}$$

where $\Delta = V \oplus V' = [\delta_0\ \delta_1 \ldots \delta_{21}]^\intercal$.

For each of these cases, we can use Equation 3.3, together with the conditions guaranteeing the relevant type of clocking, to determine a set of conditions on the various bits of Δ, K and V that must be satisfied in order for a slid pair to occur in the manner described above.

The state transition matrices form the new state contents of the three registers after α iteration(s) as shown by the following equations. There are 4 possible state transition matrices to transfer from the current state to the next one. Therefore, the total number of possible state transition matrices from the beginning of diffusion phase at $t = 0$ to the required number of iteration(s) α can be determined as $2^{2\alpha}$.

$$\text{For } \alpha = 1, \quad M\Delta = (T_x \oplus I)S_0$$

where T_x can be T, T_{ab}, T_{ac} or T_{bc} for each of the four different cases respectively.

$$\text{For } \alpha = 2, \quad M\Delta = (T_xT_y \oplus I)S_0 \tag{3.4}$$

where each of the T_x and T_y can be T, T_{ab}, T_{ac} or T_{bc}, resulting in 16 cases

$$\text{For } \alpha = 3, \quad M\Delta = (T_xT_yT_z \oplus I)S_0$$

where each of the T_x, T_y and T_z can be T, T_{ab}, T_{ac} or T_{bc}, resulting in 64 cases

Equation 3.3 is analysed using Gaussian Elimination to find the conditions mentioned above. We start with a matrix (M) with dimension (64×22) and a vector (Δ) of 22 elements in the left side and in the right side a matrix ($T^{22}N||M$) with dimension (64×86) and a vector ($K||V$) of 86 elements. Gaussian Elimination is applied to the M matrix to determine the relationship between deltas $\{\delta_0, \delta_1 \ldots \delta_{21}\}$ and the key-IV bits as shown in Appendix A.1. This is obtained from the top 22 rows. This relationship forms the second IV$'$. The last 42 rows are the bases of the conditions that are required to get slid pairs. It consists of 42 equations of 86 variables, with each equation equal to zero.

The following process was used to obtain the results presented below:

- Use MAGMA to generate the relationship between key bits and IV bits (system of equations) that result in slid pairs and shifted keystreams, as discussed above.
- Identify a single secret key and some of the corresponding IVs that result in slid pairs and shifted keystream by 1 and 2 bits (can be applied for another shifted bits)
- Use the A5/1 cipher to generate the keystreams using this secret key and the corresponding IVs to demonstrate the practical shifted keystreams.

3.3.2 Result of Analysis

This section focuses on finding slid pairs of A5/1 for $\alpha = 1$ and 2 only. The analysis is performed by examining Equation 3.3 for each clocking case. For each clocking case, we find slid pairs for the same secret key (where the secret key is fixed for both slid pairs) with different IV's and describe the relationship between the first IV and second IV′. This relationship is key dependent. In addition, the secret key is expressed as a relationship between some key bits and other bits from both the key and the first IV.

For $\alpha = 1$

Keystream sequences shifted by one bit can be obtained from two different key-IV pairs (same key with different IV's). Based on the analysis of Equation 3.3 described above, Table 3.4 shows the number of key and IV bits that must be specified to form a 64-bit secret key that results in 1-bit shifted keystream for each of the four cases. Table 3.5 shows two examples of slid pairs (in hex) for a secret key with two different IV's that generate keystream sequences that are shifted by one bit. Note: the bits {0} and {1} are the shifted bits (in binary) between these two keystream sequences in each case. Note also that the final byte of keystream only contains three or four bits, and that these are treated as MSBs in each case. The keystream length is 228 bits and due to the one bit shift, the two sequences contain a common 227-bit sequence.

Table 3.4: Slid pairs after 1 clock

cases	1	2	3	4
free key bits	20	22	21	20
Involved IV bits	4	22	22	22

Table 3.5: Two keystreams shifted by 1 bit generated from the same secret key and different IVs

key	0x2D37B6F7292DFFFB
IV1	0x200000
IV2	0xE05A00
Keystream1	{0}0x5E449A6F3414F3CD76F567275D31CFE1A4F4AE4F4D3C954D3CB124D9A
Keystream2	0x5E449A6F3414F3CD76F567275D31CFE1A4F4AE4F4D3C954D3CB124D9A
key	0xF77832CC89EFFFFB
IV1	0x200000
IV2	0x4001A4
Keystream1	{1}0xF798818F32A6B4772F5B2E55B8808541301E49CA76B11BC46F65C1494
Keystream2	0xF798818F32A6B4772F5B2E55B8808541301E49CA76B11BC46F65C1494

For Case 1, a 64-bit secret key is formed from 20 free key bits (k_{42}, k_{44}, k_{45}, k_{47} to k_{63}) and 4 IV bits ($v_0, v_3, v_{11} \oplus v_{13}$). These 24 bits specify the remaining 44 key bits using the equation in Appendix A.1. The rest of the IV bits are free to be chosen and do not affect the secret key. Therefore, by choosing all possible values for the 20 free key bits and the 22 IV bits, the total number of slid pairs for Case 1 is 2^{42} and the probability that a randomly chosen key satisfies these equations for a given IV is 2^{-44}.

The total number of slid pairs in each of Cases 2, 3 and 4 can be calculated similarly, and the total numbers of slid pairs are 2^{44}, 2^{43} and 2^{42} respectively. Therefore, the total number of slid pairs after 1 clock (for the 4 cases combined) is 2^{45}. Likewise, the probability that a randomly chosen key satisfies the equations for any of these cases (for a given IV) is found to be 2^{-41}.

For $\alpha = 2$

Keystream sequences that are shifted by two bits can again be obtained from two different key-IV pairs (same key with different IV's). We determine the conditions under which this occurs by analysing Equation 3.4. As discussed above, there are 16 possible alternatives for the term $T_x\ T_y$ in Equation 3.4: we denote the corresponding cases in the analysis by i_j, where i refers to the case associated with T_y (the first clock step) and j refers to the case associated with T_x (the second clock step).

Based on the analysis of Equation 3.4, Table 3.6 shows the number of key and IV bits that must be specified to form a 64-bit secret key that results in a 2-bit shifted keystream for each of these 16 cases. Table 3.7 shows an example of a slid pair in hex for a secret key with two different IV's that generate sequences which are shifted by two bits. Note: the bits {01} are the shifted bits (in binary) between these 2 keystream sequences. The keystream length is 228 bits and due to the shift of two bits, the two keystream sequences contain a common 226 bit sequence.

Table 3.6: Slid pairs after 2 clocks

cases	1_1	1_2	1_3	1_4	2_1	2_2	2_3	2_4	3_1	3_2	3_3	3_4	4_1	4_2	4_3	4_4
Free key bits	18	18	18	18	18	22	18	18	18	18	21	18	18	18	18	21
Involved IV bits	7	22	22	22	22	22	22	22	22	22	22	22	22	22	22	22

Table 3.7: Two keystreams shifted by 2 bit generated from the same secret key and different IVs

key	0x1DCCC463432BFFFB
IV1	0x200000
IV2	0xC36D70
Keystream1	{01}0x916F6D486AF626F3247A77C97846CAED1D6D35B95D712F89B6B11EEB0
Keystream2	0x916F6D486AF626F3247A77C97846CAED1D6D35B95D712F89B6B11EEB1

Slid pairs after 2 clocks can occur in 16 different cases as shown in Table 3.6. Numbers of slid pairs can be calculated in a manner similar to that used for $\alpha = 1$. We found that the total number of slid pairs for $\alpha = 2$ (for the 16 cases combined) is $2^{25}+ 2^{40}+ 2^{40}+ 2^{40}+ 2^{40}+ 2^{44}+ 2^{40}+ 2^{40}+ 2^{40}+ 2^{40}+ 2^{43}+ 2^{40}+ 2^{40}+ 2^{40}+ 2^{40}+ 2^{43} \approx 2^{45.46}$. Likewise, the probability that a randomly chosen key satisfies the equations for any of these cases (for a given IV) is found to be $2^{-40.54}$.

Similarly, this work can be extended for a greater number of iterations to analyse the occurrence of the slid pairs in the A5/1 cipher such as for $\alpha = 3, 4, \ldots$. The number of cases to be analysed increases as the value of α increases to $2^{2\alpha}$ as mentioned in Section 3.3.1.

3.3.3 Attack Procedure

Since A5/1 has a 64-bit key and a 64-bit internal state, it is not feasible to simply guess the whole secret key that generates the keystream and check whether the guess is correct or not. However, if it is possible to identify the occurrence of a slid pair, the resulting relationship can be used to reduce the number of key bits that need to be guessed, forming the basis for an attack. We outline the procedure for such an attack.

For each conversation there is a secret key and a series of IV's (frame numbers). Initialisation with a new IV (rekeying) is performed after every 228 bits of keystream, that is, every 4.6 milliseconds. The total time elapsed to use all 2^{22} possible frame numbers is around 5 hours and 22 minutes. The first example in Table 3.5 shows that a slid pair for that specific secret key occurs after 2.51 minutes.

For the attack, we focus on the most useful attack scenario for an attacker:

ciphertext-only. This scenario assumes the attacker is able to get enough encrypted speech (ciphertext) and also that the IV's (frame numbers) are known.

We describe the algorithm for one bit shifted versions of keystream. However, it can be extended to other shifts as well. The algorithm depends on identifying where an unknown key K is used for two encrypted frames with known IV's, IV and IV$'$, such that the resulting keystreams are shifted versions of one another. We want to identify this from knowledge of encrypted frames alone. If we XOR an encrypted frame and a one-bit shifted version of a second encrypted frame where the two keystreams are out of phase by one, the result is the XOR combination of two plaintext frames. The XOR combination of plaintext frames can be easily identified due to the redundancy of plaintext [40]. This is critical in Step 2 of the four step algorithm presented below. Note that our aim is to find the secret key rather than to decrypt an individual frame. The attack algorithm is as follows:

Attack Algorithm

Step 1: Divide the encrypted speech (ciphertext) into separate frames. Each ciphertext frame corresponds to a different IV.

Step 2: Obtain $f_i \oplus (f_j \ll 1)$ for every available pair of frames for $0 \leq i \neq j \leq 2^{22}$ and use the redundancy of speech to determine whether a one bit shifted keystream is present, where f_i and f_j are the i^{th} and j^{th} encrypted frame respectively.

- If so, note the IV's of the relevant frames.
- If not, the algorithm fails for $\alpha = 1$ for the entire conversation.

Step 3: If shifted keystream has been identified, find a candidate key, k^*, by guessing and checking each possible key, as follows:

For each of cases 1, 2, 3, and 4 do:

- Guess the free key bits in the relevant set of equations in Appendix A.1.
- Use these free key bits and the known IV's to obtain the remaining key bits using the equations of Appendix A.1.
- Use the candidate key K^* with any available IVs to generate keystreams for several frames using the A5/1 algorithm.

- Try to decrypt the frames from the previous step using the generated keystreams.
  - If decryption is successful, the candidate key for this conversation is equivalent to the actual key.
  - If not, repeat the process for another guess.

Step 4: Use the secret key with known IV's to decrypt the entire intercepted ciphertext.

Comments on algorithm:

In Step 2: it is possible that multiple pairs of frames will be identified due to the state convergence in A5/1.

In Step 3: if two encrypted frames have been identified as shifted keystream, then the guessing process should cover the four Cases 1, 2, 3 and 4 to find the correct secret key. As shown in Table 3.4, it requires guessing 20, 22, 21, 20 bits for Cases 1, 2, 3 and 4 respectively. For each guess, the guessed key bits should be substituted into the relevant equations in Appendix A.1 to find the remaining key bits. At this time, it is not clear whether the candidate key is correct or not. The proposed key must be verified by decrypting an encrypted frame. Random frames are chosen with known IVs to generate keystreams using the proposed secret key. If the keystreams decrypt the frames correctly then the candidate key is correct, if not use another guess until the correct key is found or all candidates are exhausted.

Note that the above algorithm is for 1-bit shifts. If this fails, then it is possible to try for $\alpha = 2$ or greater. The process is the same with a slight modification in Step 2 only, so that $f_i \oplus (f_j \ll \alpha)$

Attack Complexity

For Step 2: The complexity of our attack depends essentially on the number of comparisons required in Step 2. If there are N frames in the targeted conversation, then up to $N \cdot (N-1)$ comparisons must be performed and each comparison must be done in both directions $(f_i \oplus (f_j \ll 1))$ and $((f_i \ll 1) \oplus f_j)$.

For a conversation involving all 2^{22} possible IVs (2^{22} frames $\approx$ 5 hours 22 minutes), there will be up to $2^{22}(2^{22}-1) \approx 2^{44}$ comparisons and the success probability (the probability that the secret key used in this conversation forms a slid pair using two of these IVs) is approximately 2^{-19}. For a shorter conversation, the probability that a slid pair will occur among the available frames decreases in proportion to the number of comparisons performed, so with N frames of conversation the probability of success will be approximately $N \cdot (N-1) \cdot 2^{-63}$. Table 3.8 gives some examples of the number of comparisons and the probability of success for various lengths of conversation.

For Step 3: If Step 2 of the attack is successful in finding a slid pair of the kind we are seeking, then Step 3 will require up to $44 \times 2^{23} \approx 2^{28.46}$ additional calculations to find the correct secret key.

Table 3.8: Complexity of various length of conversation

No of frame	time	Comparison complexity	Probability of success	Guessing required
2^{14}	1min 16sec	2^{28}	2^{-35}	$2^{28.46}$
2^{16}	5min 2sec	2^{32}	2^{-31}	$2^{28.46}$
2^{18}	20min 6sec	2^{36}	2^{-27}	$2^{28.46}$
2^{20}	1h 21min	2^{40}	2^{-23}	$2^{28.46}$
2^{22}	5h 22min	2^{44}	2^{-19}	$2^{28.46}$

Similarly for $\alpha = 2$, the probability that the secret key used in this conversation forms a slid pair using one of the 2^{22} possible IVs is approximately $2^{-18.51}$. For shorter conversation, the probability that a slid pair may occur among the available frames is approximately $N \cdot (N-1) \cdot 2^{-62.51}$.

In a known-plaintext attack [47], the attacker has a number of plaintext and ciphertext pairs, and the keystream sequences are known, $z = m \oplus f$. Therefore, the attacker can check directly for the presence of shifted keystream instead of using the redundancy property of the messages and Step 3 is unchanged. The total complexity is almost the same as the ciphertext-only attack scenario, except for the time for checking redundancy.

3.4 Weak Key-IV Combinations

The keystream generator of A5/1 can be considered to operate nonautonomously during the loading phase, where the value of the new bit of each register during the loading phase depends on both the feedback and an external value (which is the key or IV bit). During the diffusion phase and keystream generation process, the operation can be considered as an autonomous operation, as there is no external input to the shift registers. The nonautonomous operation during the loading phase is the most crucial operation for this analysis. However, considering the operation of each register, they operate independently during the loading phase. Therefore, they operate dependently during the diffusion phase and keystream generation, where each register is clocked depending on the value in the register clocking tap and the majority value for the clocking tap values of the three registers.

After completing the loading phase, the loaded state may have one or more registers which are all-zero values. If two or three registers contain all-zero values, then the generated keystream during this specific initialisation will be constant either zeros or ones. This flaw results in sending a message in clear text if the keystream is zeros or the complement of the message if the keystream is all ones. In the case where exactly one register is all-zero values, this will reduce the effective size of state space. This section thoroughly analyses the flaw of having one or more registers which are all-zero values to give the relationships between key and IV and the probability of recovering the secret key.

Recalling Equation 3.2, we can set $\tau = -86$, so that S_0 represents the loaded state of the system, and consider the behaviour of the loading phase of A5/1. The terms $T^{22}NK$ and MV are XORed together. So, they can be represented by concatenating $T^{22}N$ and M and multiplying by a new vector KV, where $KV = [k_0 \quad k_1 \ldots k_{62} \quad k_{63} \quad v_0 \quad v_1 \ldots v_{20} \quad v_{21}]^\intercal$ as follows

$$S_0 = [T^{22}N||M]KV \tag{3.5}$$

We can apply Equations 3.2 and 3.5 to each register of the A5/1. Let S_A, T_A, N_A and M_A represents the required matrices for register A. Similarly, matrices can be written for registers B and C as follow S_B, T_B, N_B, M_B and S_C, T_C, N_C, M_C respectively. The new equations for each register can be written as follows:

$$\begin{aligned} S_{A,\tau+86} &= T_A^{22}N_AK \oplus M_AV & \quad\text{and}\quad & S_{A,0} &= [T_A^{22}N_A||M_A]KV \\ S_{B,\tau+86} &= T_B^{22}N_BK \oplus M_BV & & S_{B,0} &= [T_B^{22}N_B||M_B]KV \\ S_{C,\tau+86} &= T_C^{22}N_CK \oplus M_CV & & S_{C,0} &= [T_C^{22}N_C||M_C]KV \end{aligned}$$

This analysis investigates the effect on the security when the loading phase results in a loaded state with one or more registers containing all-zero values. If the contents of any register is all-zero after the loading phase, it will remain so for the whole of the diffusion and keystream generation phases since the register has no external input during these phases. Thus, the register contents will not be changed until the next rekeying. This leads us to consider three scenarios: three registers containing all-zeros values, two registers containing all-zeros values and one register containing all-zeros values, as discussed later. As a result, for each scenario, the relationships between keys and IVs are identified to obtain the weak key-IV pairs.

The procedures used to obtain the following result are:

- Use MAGMA to generate the relationship between key bits and IV bits (system of equations) that result in one, two and three registers that contain all-zero values at the end of the loading phase.
- Identify a secret key and one of its corresponding IVs that result in one, two and three registers that contain all-zero values
- Use the real A5/1 cipher to generate the loaded, initial states, and keystream sequence to illustrate the practical impact of the weak key-IV in the contents of registers and the keystream sequences.

3.4.1 Three Registers all Zeros

This section focuses on the situation of three registers containing all-zero values after completing the loading phase. These three registers will be clocked and produce all zero keystream bits continuously until the next rekeying. Thus the output keystream bit obtained by XORing these three zeros will also be zero.

From Equation 3.5, the behaviour of these three registers is expressed in three terms $T_A^{22}N_A||M_A$, $T_B^{22}N_B||M_B$ and $T_C^{22}N_C||M_C$. Under the assumption that these three registers will have all-zero contents in each stage after performing the loading phase, we analyse the system of equations. Equation 3.5 is analysed using Gaussian Elimination to find the conditions mentioned above. We start with assuming these three terms ($T_A^{22}N_A||M_A$, $T_B^{22}N_B||M_B$, $T_C^{22}N_C||M_C$) are equal to zeros. This process generates the relationship between key and IV bits that result in freezing the three cipher's registers. We have 64 equations with 86 variables, and hence 22 variables can be chosen freely.

Based on the analysis of Equation 3.5, Table 3.9 shows an example of weak key-IV that produces fixed keystream which contains all-zeros, where the keystream is presented in hex. The three registers A, B and C contain all-zero values after performing the loading phase due to the nonautonomous operation. The conditions and the probability of the weak key-IV's depend on the 22 IV free bits.

Table 3.9: Example of weak key-IV (to freeze three registers)

key		0010000000100000100111001101110110000001001101111001000000100011
IV		1110000000000000000000
loaded state	A	0000000000000000000
	B	0000000000000000000000
	C	00000000000000000000000
initial state	A	0000000000000000000
	B	0000000000000000000000
	C	00000000000000000000000
Keystream		0x000000000000000000000000000000000000000000000000000000000

A 64-bit secret key, which results in freezing the three registers, is formed from 22 IV free bits (v_0, v_1 to v_{21}). These 22 IV free bits specify the 64 key bits using the system of equations in Appendix A.2. Therefore, by choosing all possible values of the 22 IV free bits, the total number of weak key-IV pairs is 2^{22} and the probability that a randomly chosen key satisfies these equations (for a given IV) is 2^{-64} (For a set of 2^{22} IVs, the probability is 2^{-42}).

Attack Procedure

Since A5/1 has a 64-bit key and a 64-bit internal state, it is not feasible to simply guess the whole secret key that generates the keystream and check whether the guess is correct or not. However, if it is possible to identify the occurrence of a weak key-IV where the three registers are zeros, this enables an attacker to calculate the exact secret key directly from the known IV. Note that this procedure focuses on finding the secret key rather than on decrypting an individual frame.

In the attack scenario, we focus on the most useful attack which is ciphertext-only attack. This scenario requires getting enough length of the encrypted speech (ciphertext) and assuming the IV's (frame numbers) are known. The following algorithm represents the attacking procedure for this scenario of the weak key-IV of the A5/1.

Attack Algorithm

When the initial state of the three registers are all-zeros, the keystream is all zeros as well. For this type of keystream, the required steps to identify the secret key are as follows:

Step 1: Divide the encrypted speech (ciphertext) into separate frames. Each ciphertext frame corresponds to a single IV.

Step 2: If any encrypted frame is intelligible and seems to be a plaintext, this indicates that the keystream is all zeros.

Step 3: Run the following algorithm on this frame, assuming that the keystream is generated by three registers containing all-zeros values.

- Use the known IV to calculate the potential secret key using the equations in Appendix A.2 for three registers containing all-zeros. In this scenario there is no guessing.
- Use the potential secret key with other IV's to generate keystreams for the corresponding frames using the A5/1 algorithm.
- Combine the generated keystreams with decrypt the encrypted frames that correspond to the IV's.
  - If the encrypted frames are decrypted successfully, then the secret key has been identified, and can be used with the known IV's to decrypt the entire intercepted ciphertext.
  - If not, apply the attack procedure outlined in Section 3.4.2, this applies when two registers contain all-zero values.

Attack Complexity, if the three registers contain all-zeros, then it is possible to directly calculate the secret key from the given IV using equations in Appendix A.2 for three registers contain all-zeros. The probability of occurrence of this type of keystream is 2^{-64}.

3.4.2 Two Registers all Zeros

This section focuses on the situation in which two registers contain all-zero values after performing the loading phase. As the clocking stage in each of these two registers will contain a zero, the majority value will be zero. Hence, these two registers will be clocked every time. The third register will be clocked until the content of its clocking stage has value “1”. Since the diffusion phase consists of 100 clocking steps before producing any keystream bits and the largest register has only 23 stages, this process will ensure that the third register will be in its steady state before the keystream generation begins.

The keystream bit z is obtained by XORing the contents of the left most stage of each registers s_a^{18}, s_b^{21} and s_c^{22} for registers A, B and C respectively as $z_t = s_{a,t}^{18} \oplus s_{b,t}^{21} \oplus s_{c,t}^{22}$. The left most stage of the non-zero register will be fixed, and could contain either 0 or 1, while the other two registers contain only zeros. The value in this stage is the value of the key stream bit. Thus the keystream has constant value for the entire frame.

Considering the operation of the LFSR during the loading phase is linear and non-autonomous where the new bit depends on both the feedback and key or IV bit, it is easy to relate the contents of the three registers after the loading phase to the key and IV bits that were loaded into these registers. From Equation 3.5, the behaviour of these three registers is expressed in three terms $T_A^{22}N_A||M_A$, $T_B^{22}N_B||M_B$ and $T_C^{22}N_C||M_C$. Our analysis is conducted under the assumption that we are looking for two registers which have zeros as content in each stage of these two registers after performing the loading phase. Specifically in the three cases:

Case 1 Registers A and B contain all-zeros

Case 2 Registers A and C contain all-zeros

Case 3 Registers B and C contain all-zeros

Equation 3.5 is analysed using Gaussian Elimination to find the conditions mentioned above. We start with the terms $(T_A^{22}N_A||M_A,\ T_B^{22}N_B||M_B)$, $(T_A^{22}N_A\ ||M_A,\ T_C^{22}N_C||M_C)$ and $(T_B^{22}N_B||M_B,\ T_C^{22}N_C||M_C)$. Setting each term equal to zeros and applying Gaussian Elimination will give us the conditions and relationships between key and IV bits that apply to these weak key-IV's. For the three scenarios, we have 41, 42, 45 equations respectively with 86 variables, and hence 45, 44 and 41 variables can be chosen freely.

Conditions and probabilities of the weak key-IV pairs depend on the number of available free bits. Table 3.10 shows the number of key and IV free bits that must be chosen to form a 64-bit secret key that results in a weak key-IV. This type of weak key-IV freezes two registers and produces fixed keystream for the entire frame.

Table 3.10: Number of free bits for weak key-IV of each case of two register all-zero

cases	1	2	3
key free bits	23	22	19
Involved IV bits	22	22	22

Case 1: It is possible to obtain a weak key-IV pair for A5/1 by freezing two registers A and B. Based on the analysis of Equation 3.5 described above, Table 3.11 shows two examples of weak key-IV that produce fixed keystream (either zeros or ones) where the keystream is presented in hex. The two registers A and B contain all zeros after performing the loading phase due to the nonautonomous operation. The keystream bits are formed as a copy from the output bit (last bit) of the register C. Note that the bold and underlined bits are the clocking control bits and the output bits respectively of the content of registers A, B and C.

For Case 1, a 64-bit secret key is formed from 23 key free bits (k_{41}, k_{42}, k_{43} to k_{63}) and 22 IV bits (v_0, v_1 to v_{20}, v_{21}). These 45 bits specify the remaining 41 key bits using the system of equations in Appendix A.2. Therefore, by choosing all possible values for the 23 key free bits and the 22 IV bits, the total number of weak key-IV for Case 1 is 2^{45} and the probability that a randomly chosen key satisfies this system of equations (for a given IV) is 2^{-41}. (For a 2^{22} IVs, the

Table 3.11: Two examples of weak key-IV (Case 1)

key		0110100101000010100110111011101001001011011111111111111111111001
IV		1110000000000000000000
loaded state	A	00000000**0**0000000000
	B	0000000000**0**00000000000
	C	1111111011**0**010110111001
initial state	A	00000000**0**000000000$\underline{0}$
	B	0000000000**0**0000000000$\underline{0}$
	C	1111111101**1**00101101110$\underline{0}$
Keystream		0x000000000000000000000000000000000000000000000000000000000
key		1000001000101110111010101011001101100110111111111111111111111001
IV		1100000000000000000000
loaded state	A	00000000**0**0000000000
	B	0000000000**0**00000000000
	C	1110110111**0**111110100111
initial state	A	00000000**0**000000000$\underline{0}$
	B	0000000000**0**0000000000$\underline{0}$
	C	0111011011**1**01111101001$\underline{1}$
Keystream		0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

probability is 2^{-19}).

Case 2: Similarly to Case 1, it is possible to freeze another pair of two registers A and C. So, it is possible to obtain a weak key-IV pair for A5/1 from this condition. Based on the analysis of Equation 3.5 described above, Table 3.12 shows two example of weak key-IV that produces fixed keystream either zeros or ones, where the keystream is presented in hex. The two registers A and C became zeros after performing the loading phase due to the nonautonomous operation. The keystream bits are copied from the output bit of register B. Note that the bold bits are the clocking tap and the underlined bits are the output bits.

For Case 2, a 64-bit secret key is formed from 22 key free bits (k_{42}, k_{43} to k_{63}) and 22 IV bits (v_0, v_1 to v_{21}). These 44 bits specify the remaining 42 key bits using the system of equations in Appendix A.2. Therefore, by choosing all possible values for the 22 key free bits and the 22 IV bits, the total number of weak key-IV for Case 2 is 2^{44} and the probability that a randomly chosen key satisfies this system of equations (for a given IV) is 2^{-42}. (For a range of 2^{22} IVs, the probability is 2^{-20}).

Case 3: Similarly to Case 1 and 2, it is possible to freeze the last combination of two registers B and C to obtain a weak key-IV. Based on the analysis of Equation 3.5 described above, Table 3.13 shows two examples of weak key-IV

Table 3.12: Two examples of weak key-IV (Case 2)

key		0101100010011111010111110101111001010101001111111111111110111001
IV		1110000000000000000000
loaded state	A	00000000**0**0000000000
	B	0101010101**1**10110001000
	C	0000000000**0**000000000000
initial state	A	00000000**0**000000000$\underline{0}$
	B	0101010101**1**1011000100$\underline{0}$
	C	0000000000**0**00000000000$\underline{0}$
Keystream		0x000000000000000000000000000000000000000000000000000000000
key		0111111010110100000001110001110110001110101111111111111110111001
IV		1100000000000000000000
loaded state	A	00000000**0**0000000000
	B	0001101100**1**00110101011
	C	0000000000**0**000000000000
initial state	A	00000000**0**000000000$\underline{0}$
	B	0001101100**1**0011010101$\underline{1}$
	C	0000000000**0**00000000000$\underline{0}$
Keystream		0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

pairs that produce fixed keystream either zeros or ones, where the keystream is presented in hex. The two registers B and C became zeros after performing the loading phase due to the nonautonomous operation. The keystream bits are formed from the output bit of the register A. Note that the bold bits are the clocking tap and the underlined bits are the output bits.

Table 3.13: Two examples of weak key-IV (Case 3)

key		1011100011000000100000011000111111010011111111111111110111111001
IV		1100000000000000000000
loaded state	A	01000001**0**0010000001
	B	0000000000**0**00000000000
	C	0000000000**0**000000000000
initial state	A	10100000**1**000100000$\underline{0}$
	B	0000000000**0**0000000000$\underline{0}$
	C	0000000000**0**00000000000$\underline{0}$
Keystream		0x000000000000000000000000000000000000000000000000000000000
key		1010011010110011010110100000010110000010010011111111110111111001
IV		1111000000000000000000
loaded state	A	10111000**0**1111110100
	B	0000000000**0**00000000000
	C	0000000000**0**000000000000
initial state	A	11001011**1**000011111$\underline{1}$
	B	0000000000**0**0000000000$\underline{0}$
	C	0000000000**0**00000000000$\underline{0}$
Keystream		0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

For Case 3, a 64-bit secret key is formed from 19 key free bits (k_{45}, k_{46} to k_{63}) and 22 IV bits (v_0, v_1 to v_{21}). These 41 bits specify the remaining 45 key bits using the system of equations in Appendix A.2. Therefore, by choosing all possible values for the 19 key free bits and the 22 IV bits, the total number of weak key-IV for Case 3 is 2^{41} and the probability that a randomly chosen key satisfies these equations (for a given IV) is 2^{-45}. (For 2^{22} IVs, the probability is 2^{-23}).

As mentioned above, the number of weak key-IV pairs for each of cases 1, 2 and 3 are 2^{45}, 2^{44} and 2^{41} respectively. Therefore, the total number of weak key-IV pairs, when two registers contain all-zeros, is $2^{45.64}$. Likewise, the probability that a randomly chosen key satisfies the equations for any of these cases (for a given IV) is found to be $2^{-40.36}$. (For a set of 2^{22} IVs with fixed key, the probability is $2^{-18.36}$).

Attack Procedure

As described in Section 3.4.1, if a specific operation of A5/1 can be identified, this reduces the required effort of attacking to obtain the secret key. This section analyses when a keystream sequence is either all zeros or all ones. When a keystream sequence is all zeros, it may be generated by either all three registers are zeros or only two registers are zeros. Section 3.4.1 discussed the situation of three registers are zeros. To remind the reader, the attacking algorithm assumes the ciphertext-only attack to recover the secret key.

Attack Algorithm

The following algorithm can be applied when a keystream sequence is either all zeros or all ones. When it is all zeros, firstly apply the algorithm presented in Section 3.4.1, if it fails, apply the following algorithm.

Step 1: Divide the encrypted speech (ciphertext) into separate frames. Each ciphertext frame corresponds to a single IV.

Step 2: If any encrypted frame is intelligible (seems to be plaintext) and fails to the previous algorithm in Section 3.4.1, or if the complement of any encrypted frame is intelligible and seems to be plaintext, this indicates that the keystream is all ones, then:

Step 3: Guess and check for each possible key of cases 1, 2 and 3 as follows:

- Guess the free key bits in the relevant set of equations in Appendix A.2 for two registers contain all-zeros.
- Use these guessed free key bits and the known IV to calculate the remaining key bits using these equations in Appendix A.2.
- Use the generated potential secret key with other IV's to generate keystreams using A5/1 algorithm.
- Try to decrypt the encrypted frames using the generated keystream.
  - If the encrypted frames are decrypted successfully, then the secret key has been identified.
  - If not, repeat the process for another guess.

Use the secret key with known IV's to decrypt the entire intercepted ciphertext.

Attack Complexity: For the two registers contain all-zeros where the keystream is either all zeros or all ones, it requires calculation up to $2^{23.64}$ to obtain the secret key. The probability of success for this type of keystreams is $2^{-40.36}$.

3.4.3 One Register all Zeros

It is possible to find key and IV pairs such that exactly one register contains all-zero values after the loading phase. Whether this register is clocked or not during the keystream generation, the contribution to the keystream bit form this register is zero. In this situation at least one of the other two registers will be clocked. The value of the keystream bits actually depends only on the content of the last bit of these other two registers.

For the scenario where one register contains all-zeros, considering the operation of the LFSR during the loading phase is linear and non-autonomous where the new bit depends on both the feedback and key or IV bit, it is easy to relate the contents of the three registers after the loading phase to the key and IV bits that were loaded into these registers. From Equation 3.5, the behaviour of these three registers is expressed in three terms $T_A^{22}N_A||M_A$, $T_B^{22}N_B||M_B$ and $T_C^{22}N_C||M_C$. Our analysis is conducted under the assumption that we are looking for a register which has zero contents in its stages after performing the loading phase as shown in the three cases:

Case 4 Register A contains all-zero values

Case 5 Register B contains all-zero values

Case 6 Register C contains all-zero values

Equation 3.5 is analysed using Gaussian Elimination (GE) to find the conditions mentioned above. We start with the term $(T_A^{22}N_A||M_A)$ which is equal to zero to find the relationship between key and IV bits. This process is applied for the other two terms $(T_B^{22}N_B||M_B)$ and $(T_C^{22}N_C||M_C)$. This process generates the relationship between key and IV bits that result in freezing a register after performing the loading phase. For the three cases (4, 5 and 6), we have 19, 22, 23 equations respectively. For each case, there are 86 variables, and hence 67, 64 and 63 variables can be chosen freely.

If a register contains all zeros at the end of the loading phase, it will remain all-zero for the entire frame. The conditions and the probabilities of the weak key-IVs depend on the number of free bits. Table 3.14 shows the number of key and IV free bits that must be chosen to form a 64-bit secret key that results in a weak key-IV that makes one register have all-zero values for the entire frame.

Table 3.14: Number of free bits for weak key-IV of each case of one register containing all-zeros

cases	4	5	6
key free bits	45	42	41
Involved IV bits	22	22	22

Case 4: Based on the analysis of Equation 3.5 described above to freeze register A, Table 3.15 shows an example of weak key-IV that makes register A contains all-zero values until the next rekeying. The keystream is presented in hex and depends only on two registers B and C. Register A contains all-zero values after performing the loading phase due to the nonautonomous operation.

Table 3.15: Example of weak key-IV (Case 4)

key		1001111011001000001111111111111111111111001111111111111111111001
IV		1110000000000000000000
loaded state	A	0000000000000000000
	B	0100000111000111010100
	C	11010011010111101110110
initial state	A	0000000000000000000
	B	1011101101100100100111
	C	11011011011011100110011
Keystream		0x507802ACC6711F53C436082322478AE8CB842631EA9CB9CC6869D6FCA

For Case 4, a 64-bit secret key is formed from 45 key free bits (k_{19}, k_{20} to k_{63}) and 22 IV bits (v_0, v_1 to v_{21}). These 67 bits specify the remaining 19 key bits using the system of equations in Appendix A.2. Therefore, by choosing all possible values for the 45 and 22 key and IV free bits respectively, the total number of weak key-IV pairs for Case 4 is 2^{67} and the probability that a randomly chosen key satisfies these equations (for a given IV) is 2^{-19}.

Case 5: Similarly to Case 4, it is possible to freeze register B. Based on the analysis of Equation 3.5, Table 3.16 shows an example of weak key-IV that produces a keystream (in hex) from only two effective registers A and C. Register B contains all-zero values after performing the loading phase due to the nonautonomous operation.

Table 3.16: Example of weak key-IV (Case 5)

key		1000010110000110010111111110111101111110011111111111111111111001
IV		1110100000000000000000
loaded state	A	0001110101010110110
	B	0000000000000000000000
	C	01111000000000100101011
initial state	A	0111101111111010110
	B	0000000000000000000000
	C	11011010101110111010111
Keystream		0x22832052674272A5FE39A39A530A861AD7672C4976B4B4EBE5A3C7413

For Case 5, a 64-bit secret key is formed from 42 key free bits (k_{22}, k_{23} to k_{63}) and 22 IV bits (v_0, v_1 to v_{21}). These 64 key-IV bits specify the remaining 22 key bits using the system of equations in Appendix A.2. Therefore, by choosing all possible values for the 42 key free bits and the 22 IV free bits, the total number of weak key-IV for Case 5 is 2^{64} and the probability that a randomly chosen key satisfies these equations for a given IV is 2^{-22}.

Case 6: Similarly to Case 4 and 5, it is possible to freeze the third register C. Based on the analysis of Equation 3.5 as described above, Table 3.17 shows an example of weak key-IV that produces a keystream (in hex) using two effective registers A and B. Register C contains all-zero values after performing the loading phase until the next rekeying due to the nonautonomous operation during the loading phase.

Table 3.17: Examples of weak key-IV (Case 6)

key		0011110000100011100000111111111111111110011111111111111111111001
IV		1110000000000000000000
loaded state	A	1001100111001111111
	B	1011111011111010101110
	C	00000000000000000000000
initial state	A	0000101101110000000
	B	1000011100111100110000
	C	00000000000000000000000
Keystream		0x0F162408D5ACED21E25C9CD1B78E0E0A687BCE90F2643E52E99013206

For Case 6, a 64-bit secret key is formed from 41 key free bits (k_{23}, k_{24} to k_{63}) and 22 IV bits (v_0, v_1 to v_{21}). These 63 bits specify the remaining 23 key bits using the equation in Appendix A.2. Therefore, by choosing all possible values for the 41 key free bits and the 22 IV free bits, the total number of weak key-IV pairs for Case 6 is 2^{63} and the probability that a randomly chosen key satisfies these equations for a given IV is 2^{-23}.

As mentioned above, the number of weak key-IV pairs for each of cases 4, 5 and 6 are 2^{67}, 2^{64} and 2^{63} respectively. Therefore, the total number of weak key-IV's that result in freezing one register of A5/1 is $2^{67.25}$ that means there are $2^{45.25}$ weak keys. Likewise, the probability that a randomly chosen key satisfies the equations for any of these cases (for a given IV) is $2^{-18.75}$.

Statistical analysis

From the above analysis, the complexity of recovering the secret key using a keystream sequence which is generated by a weak key-IV such that a single register contains all-zeros is close to exhaustive search and the probability of success is $2^{-18.75}$. The rest of the section focuses on statistical analyses to distinguish a keystream generated by two non-zero registers only (while the third register contains all-zero values) from another keystream generated by non-zero three registers.

A number of statistical tests available for testing randomness, see for example Gustafson [57] and the National Institute of Standards and Technology (NIST) [90]. We use three of these tests to check the randomness of sequences generated by the above cases. These three tests are Balance, Runs tests and Linear Complexity to assess the randomness. Balance test measures the proportion of zeros and ones for an entire sequence. The number of zeros and ones for a given sequence should meet the conditions of the randomness test. So, the probability is $P(0) = P(1) = \frac{1}{2}$. Runs test focuses on the total number of runs in a sequence. Runs is an uninterrupted sequence for specific patterns of bits whether zeros or ones. Linear Complexity determines the smallest LFSR that can generate the whole keystream over the finite field $\mathbb{F}_2^n$ using Berlekamp-Massey algorithm [83].

Frames are generated for different scenarios of the targeted operation of the A5/1 cipher as follows

- None of the registers A, B or C contain all-zero values (A, B and C all participate to form the keystream bits)
- Register A contains all zeros (B and C participate to form the keystream bits)
- Register B contains all zeros (A and C participate to form the keystream bits)
- Register C contains all zeros (A and B participate to form the keystream bits)

For each scenario, 100,000 228-bit frames are generated from 100,000 random loaded states. The Balance and Runs test are applied for these generated frames. Figure 3.8 demonstrates the Balance test for each scenario. Table 3.18 shows the result of the Runs test. Figures 3.9 and 3.10 show the result of Linear Complexity of A5/1 for frame length 228 bits and 2000 bits respectively.

From the above tests, the bits' distributions and bit balancing for all scenarios are alike. So we have not been able to distinguish a keystream generated by

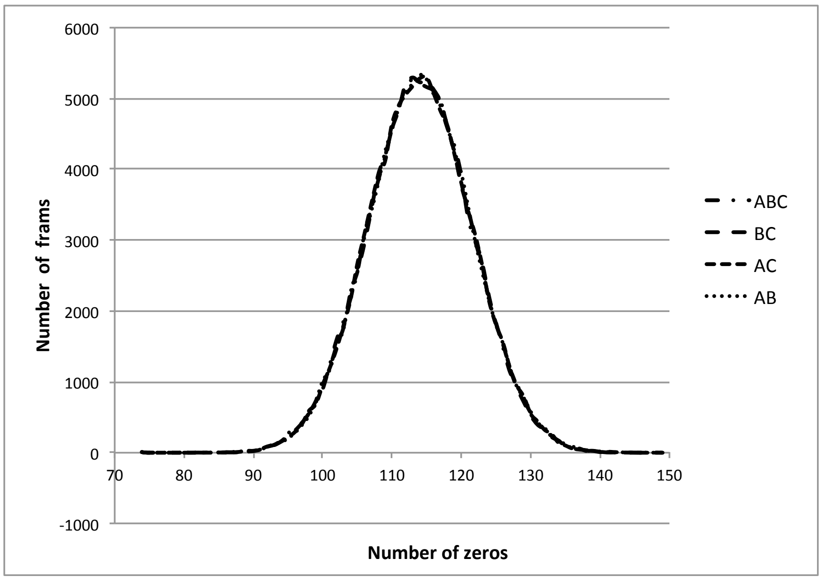


Figure 3.8: Result of the balance test for zeros bits

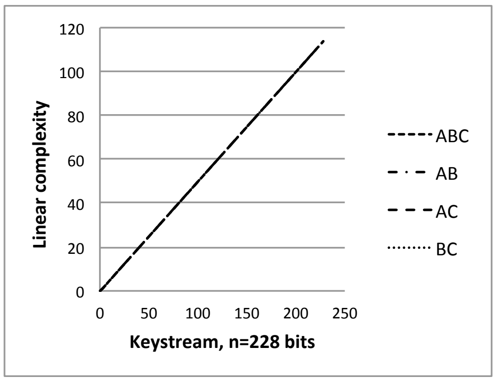


Figure 3.9: Linear complexity for keystream sequences, for $n = 228$ bits

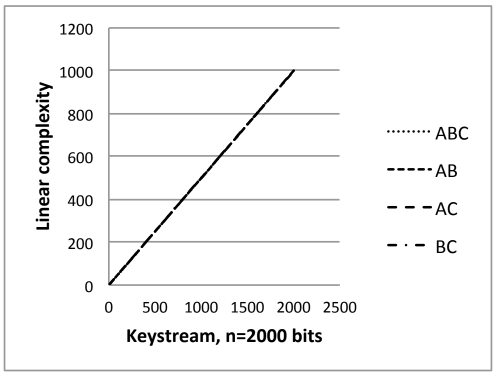


Figure 3.10: Linear complexity for keystream sequences, for $n = 2000$ bits

2 registers (while the third register contains all-zeros) using Balance, Runs or Linear Complexity tests. Since we have not determined a way to distinguish such keystreams, we will not discuss further any method of attacking A5/1 in this scenario.

3.5 Summary and Security Impact

This chapter examined the initialisation process of the A5/1 stream cipher. A5/1 has an internal state of 64 bits, that is shorter than the combined key-IV length

Table 3.18: The average counts from the runs test for 4 scenarios based on 10^5 randomly simulated states

Runs	A, B, C	B, C	A, C	A, B
1	57.471	57.504	57.509	57.497
2	28.592	28.621	28.634	28.621
3	14.255	14.261	14.237	14.255
4	7.091	7.087	7.103	7.115
5	3.532	3.523	3.532	3.529
6	1.767	1.759	1.752	1.756
7	0.874	0.875	0.872	0.869
8	0.437	0.437	0.438	0.437
9	0.219	0.216	0.217	0.214
10	0.109	0.110	0.109	0.108
11	0.053	0.055	0.054	0.053
12	0.028	0.027	0.026	0.027
13	0.013	0.014	0.013	0.013
14	0.007	0.007	0.007	0.007
15	0.003	0.003	0.003	0.003
16	0.001	0.001	0.002	0.001
17	0.0	0.001	0.001	0.001
18	0.0	0.0	0.0	0.001
19	0.0	0.0	0.0	0.0
20	0.0	0.0	0.0	0.0
21	0.0	0.0	0.0	0.0
22	0.0	0.0	0.0	0.0
23	0.0	0.0	0.0	0.0
24	0.0	0.0	0.0	0.0
25	0.0	0.0	0.0	0.0

(64+22) of 86 bits. This leads to an effective reduction in the key-IV space during the initialisation process. The initialisation process of A5/1 loads the secret key and IV using a linear function, while the diffusion phase and keystream generation both use the same nonlinear state update function. In this chapter investigations into three flaws in the A5/1 initialisation process were described: state convergence, slid pairs and weak key-IV pairs. These problems have been discussed in detail in this chapter.

A5/1 uses a majority clocking operation during both the diffusion phase and keystream generation process to introduce non-linearity to the state update function. However, this results in an update function which is not one-to-one. As previously shown by Golić in [54, 55], this leads to state convergence during the *diffusion* phase of the A5/1 initialisation process (and similarly during keystream generation, although that is beyond the scope of this research). This work extends Golić's research [54, 55] from one step to six clocking steps. These results show that the number of inaccessible patterns for each clocking step increases non-uniformly, while the proportion of inaccessible states for each clocking step decreases non-uniformly. The non-uniform rate makes it hard to analyse for large value of α. During the first six iterations of the diffusion phase, the total number of distinct internal states of A5/1 is reduced to approximately half of the number of loaded states. As the number of iterations, α, increases then the number of distinct states decreases. After 100 iterations of the initialisation process, the total number of distinct internal states is extrapolated, based on regression analysis, to be 5% of the number of loaded states. So, there are effectively only $2^{59.59}$ distinct initial states, and therefore less than $2^{59.59}$ distinct keystreams can be produced. This may make A5/1 more vulnerable to attacks such as TMTO attack, where the distinct internal states are decreased.

Concurrent investigation as reported in Kiselev and Tokareva [71] also analysed extending the number of clocks. Their results disagree with the results presented in this thesis. However, as shown in Section 3.2.2 these results are not correct due to a logic error in their counting procedure.

As there is no specific format for the loaded state of A5/1 and the internal state is not larger than the key-IV space, every internal state is a legitimate loaded state and hence forms a slid pair with any of its pre-images under repeated iterations of the state update function. Thus, slid pairs occur ubiquitously during the *diffusion* phase of initialisation process of A5/1. The state update functions for both diffusion phase and keystream generation are identical, therefore, slid pairs for A5/1 always lead to shifted keystream sequences. A particular case of slid pairs which arise from the use of the same secret key and multiple IV's was investigated. This is a common scenario in real time applications, and only requires access to frames from a single conversation, so can be used for a practical secret key recovery attack. A ciphertext-only attack on A5/1 based on the redundancy of the plaintext property and known IV's was presented. Table 3.8 gives a summary of the required ciphertext, complexity and the probability of success for this attack. This attack (ciphertext-only) can reveal the secret key using approximately 5 minutes of conversation with comparison complexity of 2^{32} and probability of success 2^{-31}. Consequently, this problem leaves A5/1 vulnerable for ciphertext-only attack to *recover the secret key*.

The nonautonomous loading phase of A5/1 also generates weak key-IV pairs. Weak key-IV pairs result in one or more registers containing all zeroes at the end of the *loading* phase. This chapter described three scenarios of the weak key-IV pairs. These three scenarios are described as follows: Firstly, all three registers containing all-zero values occurs with probability of 2^{-64}. In this scenario, the secret key can be derived directly from a given IV based on the ciphertext-only attack. Secondly, the scenario where exactly two registers contain all-zeros can also be attacked using ciphertext-only attack to retrieve the secret key with an average probability of success $2^{-40.36}$. This requires guessing up to 23 bits to calculate the remaining secret key bits, with complexity of $2^{28.36}$. The last scenario is where only one register contains all-zeros which occurs with probability of $2^{-18.75}$. However, we have not determined a distinguisher for the keystream in this case. Thus, we did not discuss any attack method for this case. Thereby, A5/1 is vulnerable to ciphertext-only *key recover attacks* using both the first and second scenarios. Equations relating the values of key bits and IV bits in case of these scenarios are discussed in Section 3.4 and presented in Appendix A.2.

Chapter 4

Analysis of Sfinks Stream Cipher Initialisation Process

The Sfinks stream cipher was submitted to the eSTREAM project ; the ECRYPT call for stream cipher proposals, in April 2005 [29]. Sfinks is a bit-based stream cipher that takes an 80-bit secret key and 80-bit IV as inputs and has a 256-bit internal state. Sfinks is categorized as PROFILE 2A, suitable for hardware applications and associated authenticated encryption.

The Sfinks stream cipher was attacked by Courtois [36] using basic and fast algebraic attacks. These algebraic attacks exploit the state update function used during keystream generation, but do not make use of the initialisation process. Courtois found that Sfinks can be broken with complexity of 2^{71} computations using 2^{43} keystream bits, which is faster than the claimed security level of 2^{80}.

As noted above, the Sfinks stream cipher is broken as a keystream generator. The purpose of this research is to investigate the strategy used for initialisation process. We note that the state update functions are different during initialisation and keystream generation precesses. Using a modified version of the keystream generation state update function during initialisation may produce some security benefits. The objective of this chapter is to consider specifically the properties of the initialisation process, rather than keystream generation, and to consider interactions between these two processes that may result in security flaws.

As shown in Chapter 3, the A5/1 internal state size is shorter than the sum of the sizes of the key and IV, the state update function is not one-to-one and the state update functions during initialisation and keystream generation processes are identical functions. This leads to some security flaws. In contrast, Sfinks has internal state size larger than twice the size of the secret key, and the individual components of the state update functions are one-to-one. As well, the state update functions during initialisation and keystream generation processes are not similar. In the following sections we show that despite this, state convergence still occurs in the initialisation process and that slid pairs can occur and lead to shifted keystream sequences.

This chapter is organised as follows: Section 4.1 describes the specification of the Sfinks stream cipher. An analysis of state convergence during the initialisation process is presented in Section 4.2. In Section 4.3, slid pairs are presented with some examples and an attacking procedure. Section 4.5 summarises this chapter and presents the security impact of this analysis..

4.1 Specification of Sfinks Stream Cipher

The Sfinks stream cipher [29] has two main components: a shift register, S, and a nonlinear one-to-one inversion function INV, as shown in Figures 4.1 and 4.2. During the initialisation, it also uses a pipeline (memory) to delay the output of the INV function before combining this output with the shift register content. Additionally, another memory is used during keystream generation to delay the output bit of the shift register by 7 clocking steps before combining it with a specific bit of the INV function to generate a new keystream bit.

Let s_t^i denote the contents of register stage i at time t, where $i = 0, 1, \ldots, 255$ and $t \geq 0$. Sfinks uses an 80-bit secret key $k = k_0, \ldots, k_{79}$ and 80-bit initial value $v = v_0, \ldots, v_{79}$. The linear feedback function is described as follows.

$$\begin{aligned} s_{t+1}^{255} = s_t^{212} \oplus s_t^{194} \oplus s_t^{192} \oplus s_t^{187} \oplus s_t^{163} \oplus s_t^{151} \oplus s_t^{125} \oplus s_t^{115} \\ \oplus s_t^{115} \oplus s_t^{107} \oplus s_t^{85} \oplus s_t^{66} \oplus s_t^{64} \oplus s_t^{52} \oplus s_t^{48} \oplus s_t^{14} \oplus s_t^{0} \end{aligned} \tag{4.1}$$

The nonlinear function INV can be considered as a 16×16 bit S-box. The inversion function is used during both initialisation and keystream generation, but in different ways in each case. Let x_t and $y_{6,t}$ denote the 16-bit input and output of INV respectively at time t, where $x_t = (x_t^{16}, \ldots, x_t^1)$ and $y_{6,t} = (y_{6,t}^{15}, \ldots, y_{6,t}^0)$. Let x_t^i denote the i-th bit of the input of the S-box, $y_{j,t}$ denote the j-th word of 16 bits ($j = 0, 1, \ldots, 6$) in the pipeline (memory) and $y^i_{j,t}$ denote the i-th bit of $y_{j,t}$, all at time t. INV is an invertible function, $\mathbb{F}_{2^{16}} \rightarrow \mathbb{F}_{2^{16}}$ that calculates the inverse of the 16-bit input, modulo the primitive polynomial $X^{16}+X^5+X^3+X^2+1$. The 16 input bits are taken from 16 register stages at time clock t as follows.

$$\begin{aligned}(x^{16}_t, \ldots, x^1_t) =&(s^{255}_t, s^{244}_t, s^{227}_t, s^{193}_t, s^{161}_t, s^{134}_t, s^{105}_t, s^{98}_t, s^{74}_t, s^{58}_t,\\ & s^{44}_t, s^{21}_t, s^{19}_t, s^{9}_t, s^{6}_t, s^{1}_t)\end{aligned} \tag{4.2}$$

The output word $y_{6,t}$ is delayed to become $y_{0,t}$. The delayed version of the 16-bit S-box output, $y_{0,t}$, is treated as 16 bit values as $y_{0,t} = (y^{15}_{0,t}, \ldots, y^0_{0,t})$. During the initialisation, these bits are fed back to specified stages of the shift register. During keystream generation, only one bit of the output of the INV is used, and this bit contributes to the formation of the keystream bit.

During initialisation, the output of the S-box is stored in a pipeline (memory), $y_{j,t}$ for $0 \leq j \leq 6$, to manage the delay of 7 clocking steps. The total required memory to store seven 16-bit output bits is 112 memory bits. Consequently, the total effective state size during the initialisation process is the sum of shift register size and the number of memory bits: 368 bits. During keystream generation, a memory of only 7 bits, m_i for $0 \leq i \leq 6$, is used to perform a delay of 7 clocking steps to stage s^0 that is used to generate a new keystream bit, z_t. The memory stages, $y^0_{j,t}$ are also used to delay $y^0_{6,t}$ for this purpose, but the remaining 15 bits of S-box from $y^1_{j,t}$ to $y^{15}_{j,t}$ for $0 \leq j \leq 6$ are not used during the keystream generation. Therefore, the total effective state size during keystream generation is 270.

4.1.1 Initialisation Process

The initialisation process takes as input the 80-bit key and 80-bit IV and performs 128 iterations to produce the 368-bit initial state. Once this initial state is obtained, keystream generation can begin. The initialisation process is performed in two phases, which we refer to as *loading* and *diffusion*.

Note: the reference implementation of Sfinks [29,75] uses $t = -128$ to denote the start of the diffusion process but, for simplicity, we will use $t = 0$ to denote this time point.

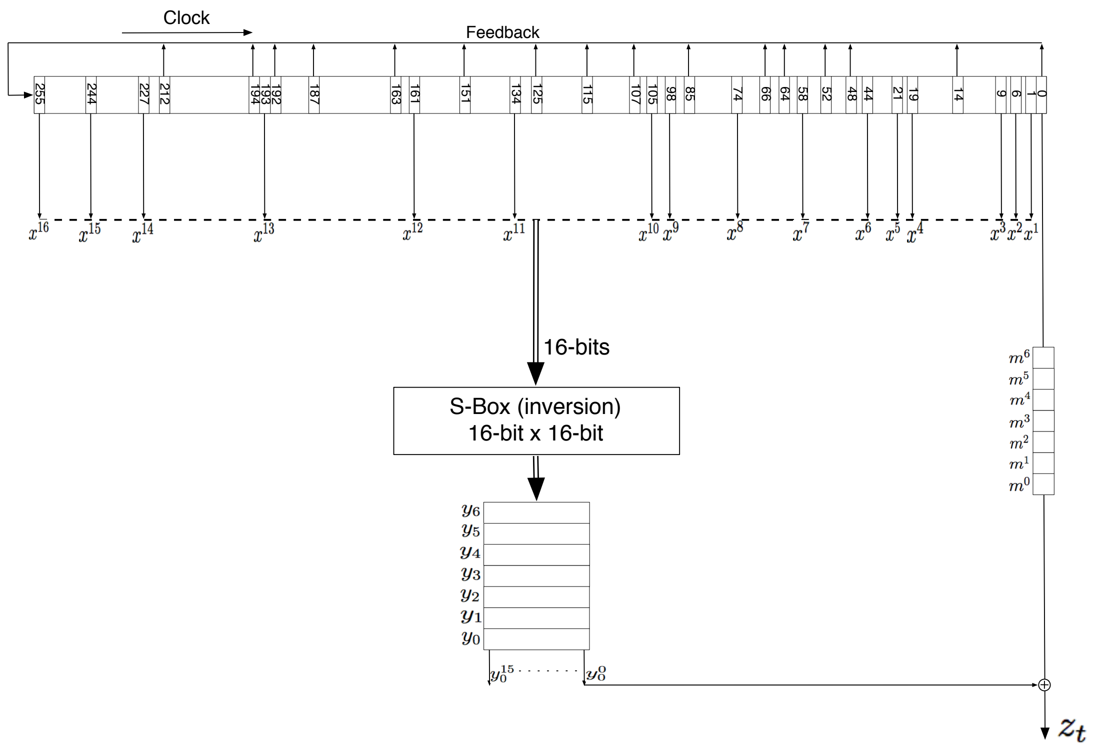


Figure 4.1: Keystream generator of Sfinks stream cipher

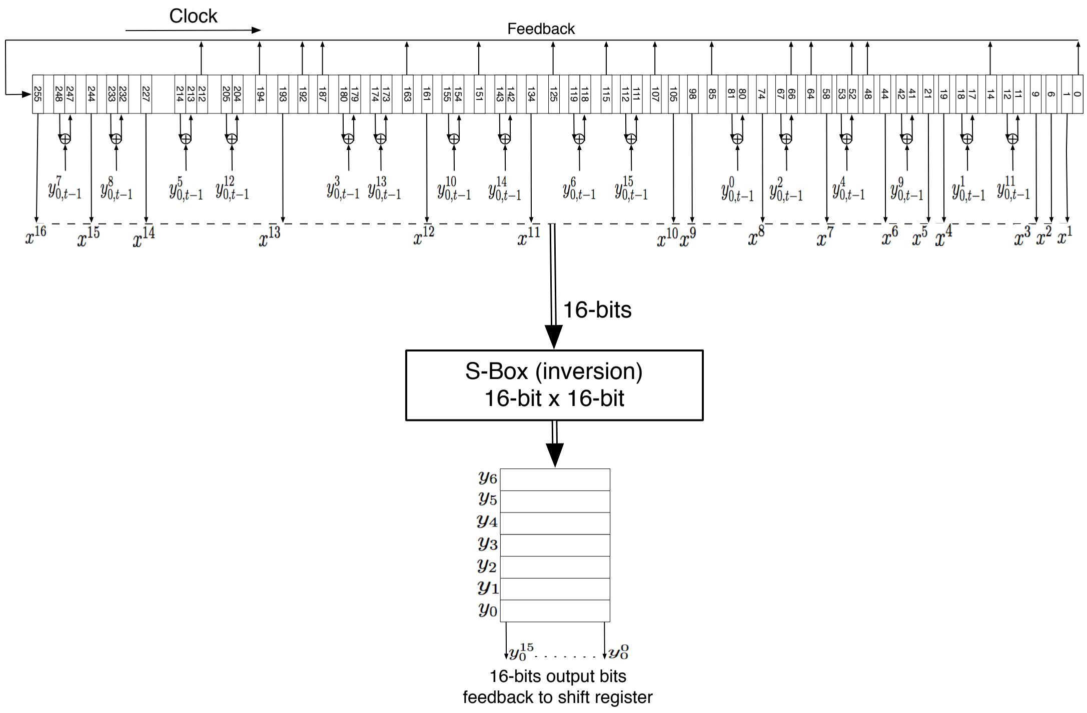
Loading Phase

Firstly, all of the register stages are set to zero. Then the 80-bit key and 80-bit IV are transferred to specified register positions $s_0^{96+i} = k_i$, for $0 \leq i \leq 79$, and $s_0^{176+i} = v_i$, for $0 \leq i \leq 79$. The register stage $s_0^{95} = 1$ and the remaining $s_0^i = 0$, for $0 \leq i \leq 94$. The output of the S-box is set to all-zero for the first seven 16-bit outputs, $y_{j,0}^i = 0$ for $0 \leq j \leq 6$ and $0 \leq i \leq 15$ (the initial value of the pipeline). In [29] this process is described as necessary to clear the pipeline stages in the hardware implementation and to provide the initial values of the output of the S-box to allow for the delay of 7 clocking steps.

When both the secret key and IV have been transferred and the rest of the state bits (the rest of shift register and the memory cells) are fixed to the designated values, the Sfinks stream cipher is in its *loaded state*. Following this, the diffusion phase begins.

Diffusion Phase

The diffusion phase consists of 128 iterations of the initialisation state-update function. Each iteration can be considered as a function which maps the state space to itself. After the diffusion phase is completed, the keystream generator is said to be in its *initial state*.



Clock
Feedback
16-bits
S-Box (inversion) 16-bit x 16-bit
16-bits output bits feedback to shift register

Figure 4.2: Initialisation processes of Sfinks stream cipher

The state update function for the diffusion process uses Equation 4.1 and the output of the nonlinear S-box function. The S-box output feeds back into 16 specified stages of the shift register with a time delay of 7 clocking steps as detailed below. Figure 4.2 gives a general overview of state update function during the diffusion phase of the Sfinks stream cipher.

$$s_t^i = s_{t-1}^{i+1} \oplus y_{0,t-1}^{i \bmod 16} \tag{4.3}$$

for $i = \{11, 17, 41, 52, 66, 80, 111, 118, 142, 154, 173, 179, 204, 213, 232, 247\}$. All other bits are shifted normally, i.e. $s_t^i = s_{t-1}^{i+1}$ for all other i's. At each iteration, the shift register is clocked and then the INV function is called to calculate the inverse of the 16-bit input to the S-box. This is stored as the S-box output in the memory where the first input is the first output. The 16-bit output of the memory is XORed with the contents of the 16 specified stages of the shift register to form the contents of another 16 stages of the shift register.

The S-box function is the only nonlinear component in the initialisation process. A complete description of the state update function during diffusion phase is:

$$s_t^i = \begin{cases} s_{t-1}^{i+1} & \text{for } i = \{0,1,\dots,254\} \text{ except } \{11,17,41,52,66,80, \\ & 111,118,142,154,173,179,204,213,232,247\} \\ s_{t-1}^{i+1} \oplus y_{0,t-1}^{i \bmod 16} & \text{for } i = \{11,17,41,52,66,80,111,118,142,154, \\ & 173,179,204,213,232,247\} \\ \bigoplus_j s_{t-1}^j & \text{for } i = 255 \\ & \text{for } j = \{0,14,48,52,64,66,85,107,115,125,151, \\ & 163,187,192,194,212\} \end{cases}$$

$$y_{j,t} = \begin{cases} y_{j+1,t-1} & \text{for } j = \{0,1,2,3,4,5\} \\ INV(x^{16},\dots,x^1) & \text{for } j = 6 \\ & (x^{16},\dots,x^1) = (s_t^{255}, s_t^{244}, s_t^{227}, s_t^{193}, s_t^{161}, s_t^{134}, \\ & s_t^{105}, s_t^{98}, s_t^{74}, s_t^{58}, s_t^{44}, s_t^{21}, s_t^{19}, s_t^{9}, s_t^{6}, s_t^{1}) \end{cases}$$

4.1.2 Keystream Generation

At $t = 128$, the Sfinks stream cipher has completed the initialisation processes and is ready for keystream generation. A memory component of 7 stages is used to apply a delay of 7 clocking steps to the rightmost bit in the shift register s_t^0. During keystream generation, the output of the INV function is not fed back to shift register, S, and so the register feedback is linear. The rightmost bit in the shift register s_t^0 is shifted to a memory stage m_t^6 at time t. The least significant bit of the 16-bit output value of the S-box, $y_{0,t}^0$ is XORed with the value of stage m_t^0 to produce each keystream bit z_t. The output equation is, $z_t = m_{t-1}^0 \oplus y_{0,t-1}^0$. where

$$m_t^i = \begin{cases} m_{t-1}^{i+1} & \text{for } i = \{0,1,2,3,4,5\} \\ s_{t-1}^0 & \text{for } i = 6 \end{cases}$$

4.2 State Convergence

State convergence occurs when two or more states at time t are mapped to the same state after α iterations (at time $t+\alpha$), for some $\alpha > 0$ and the states do not diverge after this point. That is state convergence occurs when the state update function is not one-to-one. State convergence may occur during the initialisation process and/or keystream generation. This may reduce the effective key-IV size and leave the stream cipher vulnerable to attacks such as distinguishing attacks [87] or time-memory-data trade-off attacks [23].

Analysis of the Sfinks stream cipher initialisation process is complicated by the delay of 7 clocking steps in feeding the S-box output back into the register. However we observe that the correspondence between this delay of seven steps and the difference between certain input and output taps leads to state convergence as shown below. In the remainder of this section, we refer to stages of S which provide inputs to the S-box as input stages and stages of S which receive outputs from the S-box as output stages, respectively.

From Figure 4.2, note that the distance between some input and output stages is equal to the delay time. Specifically, there is one case where s_t^i is an output stage and s_{t-7}^{i+7} is an input stage. In this case, recall from Equation 4.3 that $s_t^i = s_{t-1}^{i+1} \oplus y_{0,t-1}^{i \bmod 16}$, and note that if we complement both s_{t-1}^{i+1} and $y_{0,t-1}^{i \bmod 16}$ then the same value of s_t^i will be obtained. However, $s_{t-1}^{i+1} = s_{t-7}^{i+7}$ under regular clocking, and the S-box output $y_{0,t-1}^{i \bmod 16}$ depends on the contents of the input stage s_{t-7}^{i+7}. That is,

$$s_t^i = s_{t-7}^{i+7} \oplus y_{0,t-1}^{i \bmod 16} = \overline{s_{t-7}^{i+7}} \oplus \overline{y_{0,t-1}^{i \bmod 16}} \tag{4.4}$$

where $\overline{s}$ and $\overline{y}$ represent the complements of s and y respectively.

It is possible that complementing the contents of the input stage s_{t-7}^{i+7} may cause the required change in the S-box output bit $y_{0,t-1}^{i \bmod 16}$. This situation provides the basis of a search for states which converge.

4.2.1 States Which Converge

An examination of the Sfinks register shows there is only one input stage with a distance to the next output stage equal to 7 clocking steps. That input stage is s^{161} and the output stage is s^{154}. The contents of s^{161} correspond to the S-box

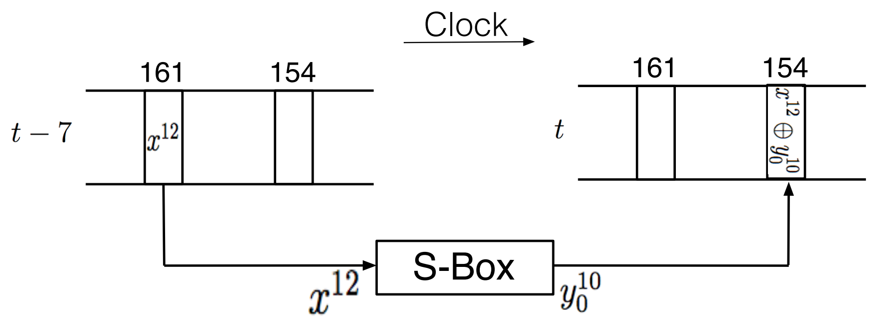


Figure 4.3: Input and output stages have 7 steps delay

input x^{12}, and the S-box output y_0^{10} is fed back to s^{154}. Specifically, $s_{t-7}^{161} = x_{t-7}^{12}$ and $s_t^{154} = s_{t-1}^{155} \oplus y_{0,t-1}^{10}$. According to Equation 4.4, the value of s_t^{154} will not be changed if complementing the input bit $s_{t-7}^{161} = x_{t-7}^{12}$ results in the output bit $y_{0,t-1}^{10}$ being complemented as well. Therefore, we look for pairs of S-box inputs $(x^{16}, \ldots, x^1)$ which differ only in bit x^{12} and for which the corresponding pair of outputs $(y_0^{15}, \ldots, y_0^0)$ differ in bit y_0^{10}.

Consider firstly the input pair for which the output pair differs only in y_0^{10}, as illustrated in Figure 4.3. Such a pair of S-box inputs (and corresponding output) exists, and is presented in Table 4.1. For emphasis, x^{12} is underlined and bold font, as is y_0^{10}. Table 4.2 gives an example of two 256-bit register states S_{t-7}^A and S_{t-7}^B which converge to the same state after 7 iterations. For efficient presentation the hex representation of the 256-bit binary state is given. Note that all register stages except s^{161} are the same. S_{t-7}^A and S_{t-7}^B both converge to S_t.

Since the register S is 256 bits long, and there are 16 stages used as input to the S-box, if we fix the contents of these 16 stages to the pattern given in Table 4.1, we are free to choose any values for the remaining 240 stages. Therefore, there are 2^{240} pairs of states which converge after 7 iterations. One such pair is presented in Table 4.2.

For the S-box pair in Table 4.1 discussed in the illustration above the inputs differed only in position x^{12} and the outputs differed only in position y_0^{10}. In general, however, it is not necessary to apply such a strict condition on the output bits. Referring to Equation 4.3, and considering 6 consecutive steps of regular clocking, we have $s_t^i = s_{t-1}^{i+1} \oplus y_{0,t-1}^{i \bmod 16} = s_{t-7}^{i+7} \oplus y_{0,t-1}^{i \bmod 16}$ for any output bit. If complementing the input bit s_{t-7}^{161} (which is x_{t-7}^{12}) causes $y_{0,t-1}^{i \bmod 16}$ to be changed, it may be possible to complement s_{t-7}^{i+7} to obtain a second state that

Table 4.1: A special S-box pair which differ in x^{12} and y_0^{10} only

	Input	Output
S-box sequence	x^{16} .... x^{1}	y_0^{15} .... y_0^{0}
Stage No.	255 244 227 193 161 134 105 98 74 58 44 21 19 9 6 1	111 142 173 204 11 154 41 232 247 118 213 52 179 66 17 80
1st value	1 1 0 1 $\underline{\mathbf{0}}$ 0 0 0 1 1 1 0 0 0 1 0	0 1 1 0 1 $\underline{\mathbf{1}}$ 0 0 0 1 1 1 0 0 0 1
2nd value	1 1 0 1 $\underline{\mathbf{1}}$ 0 0 0 1 1 1 0 0 0 1 0	0 1 1 0 1 $\underline{\mathbf{0}}$ 0 0 0 1 1 1 0 0 0 1

Table 4.2: Two states (hex) differing only in stage s^{161} which converge

S_{t-7}^A	F19B7E15AF4FF1338DDF080$\underline{\mathbf{0}}$AD8C56A42913E4B90CBEEFD3A4075AFD3351E5C1
S_{t-7}^B	F19B7E15AF4FF1338DDF080$\underline{\mathbf{2}}$AD8C56A42913E4B90CBEEFD3A4075AFD3351E5C1
S_t	BBE336FC2B7E9FE2671B9E10055B58AD481227C972187DDFA7580EB5FA66ABCB

gives the same value for s_t^i. Recall that 16 stages of the shift register S receive the output of the S-box at each iteration of the state update function. Of these 16 output stages, there are only six (s_{t-7}^{11}, s_{t-7}^{17}, s_{t-7}^{41}, s_{t-7}^{52}, s_{t-7}^{80}, s_{t-7}^{111}) which directly or indirectly affect an input stage during the six consecutive clocks. For example, the output y_0^9 of the S-box is fed back to s^{41} and there is an input to the S-box within the delay time at stage s^{44}. Allowing this bit to change, may result in divergence in later steps. Note also that if an input bit of the shift register feedback is complemented at time $t-7$, we also need the stage s_{t-7}^0 to be complemented to assure that the new bit s_{t-6}^{255} will not be changed. Therefore, when considering pairs of inputs and outputs of the S-box for which convergence occurs the values of the six S-box outputs (y_0^0, y_0^1, y_0^4, y_0^9, y_0^{11} and y_0^{15}) must be fixed. Therefore, we look for pairs of the S-box inputs which differ only in x^{12} and for which the output bits differ in y_0^{10} and possibly in any bit of y_0^2, y_0^3, y_0^5, y_0^6, y_0^7, y_0^8, y_0^{10}, y_0^{12}, y_0^{13} and y_0^{14}.

We used an exhaustive computer search to look for pairs of S-box inputs that satisfy the conditions described above, and found 273 pairs of the S-box inputs (and corresponding outputs) with such patterns. Table 4.3 gives three different examples of these S-box input and output pairs. Note that the only difference in each input pair is the underlined bold bit x^{12}. In the output pair, the underlined bold bits are the bits which differ in each pair (which must include y_0^{10}) and the bits y_0^0, y_0^1, y_0^4, y_0^9, y_0^{11} and y_0^{15} (shown in italics) should be the same in each pair. For each pair in Table 4.3, Table 4.4 provides an example of two states based on that pattern that converge to the same state after 7 iterations.

Table 4.3: Examples of complying pairs of S-box inputs and outputs

		Input																Output															
	S-box sequence	x^{16}															x^1	y_0^{15}															y_0^0
	Stage No.	255	244	227	193	161	134	105	98	74	58	44	21	19	9	6	1	111	142	173	204	11	154	41	232	247	118	213	52	179	66	17	80
1st pair	1st value	0	0	0	0	**0**	0	0	0	1	0	1	0	0	1	1	1	*0*	**1**	0	0	*0*	**1**	*0*	**0**	**1**	**0**	**0**	*1*	0	0	*1*	*0*
	2nd value	0	0	0	0	**1**	0	0	0	1	0	1	0	0	1	1	1	*0*	**0**	0	0	*0*	**0**	*0*	**1**	**0**	**1**	**1**	*1*	0	0	*1*	*0*
2nd pair	1st value	0	0	0	0	**0**	0	0	0	1	1	1	0	1	1	1	0	*0*	0	**1**	0	*0*	**1**	*1*	1	1	1	0	*0*	1	1	*1*	*1*
	2nd value	0	0	0	0	**1**	0	0	0	1	1	1	0	1	1	1	0	*0*	0	**0**	0	*0*	**0**	*1*	1	1	1	0	*0*	1	1	*1*	*1*
3rd pair	1st value	0	0	0	0	**0**	0	0	1	1	0	0	1	1	0	1	1	*1*	**1**	**1**	**0**	*1*	**1**	*0*	**0**	**1**	1	1	*1*	0	0	*0*	*1*
	2nd value	0	0	0	0	**1**	0	0	1	1	0	0	1	1	0	1	1	*1*	**0**	**0**	**1**	*1*	**0**	*0*	**1**	**0**	1	1	*1*	0	0	*0*	*1*

Table 4.4: Three examples; each two states which converge to the same state

1st pair	S_{t-7}^{A1}	**3**18B**7**E15**A**F4FF1318DDF080**0**AD**A**C56A4**2**913E4B90CBEEFD3A0075AFD3351E7C**3**
	S_{t-7}^{A2}	**7**18B**F**E15**B**F4FF1318DDF080**2**AD**8**C56A4**0**913E4B90CBEEFD3A0075AFD3351E7C**2**
	S_t	BAE316FC2B5E9FE2631BBE10055B18AD485227C972197DDFA7500EB5FA64A3CF
2nd pair	S_{t-7}^{B1}	718B7E15AF4FF1318D**C**F080**0**AD8C56A42913E4B90CBEEFD3A4075AFD3359E7C1
	S_{t-7}^{B2}	718B7E15AF4FF1318D**D**F080**2**AD8C56A42913E4B90CBEEFD3A4075AFD3359E7C1
	S_t	7E6317FC2B5E9FE26313BE10055B18AD481227C972187DDBA7480CB5FA64B3CF
3rd pair	S_{t-7}^{C1}	**3**18B**7**E15AF4**F**F1318D**C**F080**0**AD**A**C56A42913E4BD0CBEEFD3A0074AFD3379E5C3
	S_{t-7}^{C2}	**7**18B**F**E15AF4**7**F1318D**D**F080**2**AD**8**C56A42913E4BD0CBEEFD3A0074AFD3379E5C3
	S_t	FAE316FC2B7E9FE2631BBE10055B18AD4812A7C97A187DDFA7500E95FA66FBCB

From above, there are 273 pairs of S-box inputs satisfying the convergence conditions out of the 2^{15} possible input pairs. If we assume the possible values for the S-box input bits are distributed randomly and independently, this state convergence has a probability of $\frac{273}{2^{15}} = 2^{-6.9}$.

4.2.2 State Convergence Across the Initialisation Process

Recall from the loading phase Section, that the loaded state of Sfinks has a defined format, with $s_0^{95} = 1$ and $s_0^i = 0$ for $i = 0, \ldots, 94$. This slightly reduces the occurrence of state convergence for the first few iterations of the diffusion process but does not prevent it altogether.

According to the reference implementation of Sfinks [75], the S-box first receives live inputs from the input stages at $t = 1$ (after the first clock of the shift register). Convergence cannot occur until 7 clocking steps after this, at $t = 8$.

Based on the general case discussed above, however, state convergence can occur immediately after these 7 iterations. All that is required to impose the additional condition that y^2 also remain unchanged when x^{12} is complemented. Table 4.5 shows a pair of input and output bits of the S-box that can occur after the first iteration and converge after 7 iterations. This pair satisfies the condition discussed above, and therefore will lead to convergence after 7 clocking steps at $t = 8$. A pair of converging states based on this pattern is presented as an example in Table 4.6.

Table 4.5: A pair of S-box inputs and outputs that can occur at $t = -120$

	Input																Output															
S-box sequence	x^{16}	…	…	…	…	…	…	…	…	…	…	…	…	…	…	x^1	y^{15}	…	…	…	…	…	…	…	…	…	…	…	…	…	…	y^0
Stage No.	255	244	227	193	161	134	105	98	74	58	44	21	19	9	6	1	111	142	173	204	11	154	41	232	247	118	213	52	179	66	17	80
1st value	0	1	0	0	$\underline{\mathbf{0}}$	0	0	0	0	0	0	0	0	0	0	0	*0*	$\underline{\mathbf{1}}$	$\underline{\mathbf{0}}$	0	*0*	$\underline{\mathbf{0}}$	*1*	0	$\underline{\mathbf{1}}$	$\underline{\mathbf{1}}$	1	*1*	$\underline{\mathbf{1}}$	1	*1*	*1*
2nd value	0	1	0	0	$\underline{\mathbf{1}}$	0	0	0	0	0	0	0	0	0	0	0	*0*	$\underline{\mathbf{0}}$	$\underline{\mathbf{1}}$	0	*0*	$\underline{\mathbf{1}}$	*1*	0	$\underline{\mathbf{0}}$	$\underline{\mathbf{0}}$	1	*1*	$\underline{\mathbf{0}}$	1	*1*	*1*

Thus, we see that state convergence can occur throughout the diffusion process of Sfinks cipher. There are 120 iterations that may carry a state convergence during the initialisation. Based on the probability of convergence detemined above, an approximate estimate for the proportion of distinct states remaining after 128 iterations is $(1 - 2^{-6.9})^{120} = 0.9963$. Thus, the number of reachable distinct states is approximately $(1 - 2^{-6.9})^{120} \times 2^{160} = 2^{158.55}$. This can be regarded as an approximate upper bound as there may be other mechanisms of

Table 4.6: Two states (hex) which converge to the same state at $t = -120$

S^A_{-127}	**3**19B7E15AF4FF1318**9D**F080**0**AD**A**C56A4**0**913E4B90CBEEBD3A0074AFD3351E58**0**
S^B_{-127}	**7**19B7E15AF4FF1318**DC**F080**2**AD**8**C56A4**2**913E4B90CBEEBD3A0074AFD3351E58**1**
S_{-120}	3EE336FC2B7E9FE2631BBE10015B18AD485227C972187DD3A7500C95FA64A3CB

convergence in addition to these we have identified.

4.3 Slid Pairs and Synchronisation Attacks

As discussed in Chapter 3, for stream ciphers, it is sometimes possible to find different key-IV pairs that produce phase shifted keystreams [25, 42, 73, 86, 108]. The state update function for the initialisation process defines a cycle of transitions of the internal state. Therefore, each (K, IV) pair represents a point on such a cycle. If it is possible to find a second loaded state generated by a pair (K', IV$'$) as a slid pair of the key-IV pair (K, IV) after a number of iterations, α, then, the (K', IV$'$) pair is in the same cycle as the (K, IV) pair. As discussed previously, the probability of obtaining shifted keystream sequences from such a pair depends on the distance between (K', IV$'$) and (K, IV) and the degree of similarity of the state update functions used during the initialisation process and the keystream generation process.

4.3.1 Slid Pairs Using Sfinks

The main purpose of this analysis is to identify conditions under which a slid key-IV pair of the Sfinks cipher is obtained from a specific loaded key-IV after a number of iterations α. Comparing the properties listed in Section 2.7.3 with the Sfinks cipher, we note that properties (a) and (b) apply, but the state update functions during the initialisation and keystream generation processes are somewhat different. The format of the loaded state is specified. For a given particular key-IV pair, the task is to identify when the next loaded state may occur during the initialisation process. Moreover, it is important to determine a relationship between the original loaded key-IV pair (K, IV) and the next slid key-IV pair (K', IV$'$), which is derived from the original key-IV after a given number of iterations α.

Analysis of the Sfinks stream cipher initialisation process is complicated by the delay of 7 clocking steps in feeding the S-box output back into the shift register, S. However, we observe that the correspondence between the content of stage $s^{95} = 1$ and the next output feedback stage from the S-box s^{80} may lead to slid pairs after a number of iterations α.

Some conditions need to be met for loaded state of Sfinks to produce a slid pair after α iterations. Slid pairs can occur after α iterations if the content of the stage s^{95}_{α} is 1 and the stages from s^{94}_{α} to s^{0}_{α} are all zeros (to follow the Sfinks' loading format). The input bits to the S-box from (x^{10}_t to x^{1}_t) should be zeros except some cases as shown below. The output of the S-box $(y^{11}_{0,t}, y^{9}_{0,t}, y^{4}_{0,t}, y^{2}_{0,t}, y^{1}_{0,t}, y^{0}_{0,t})$ should also be zeros except $y^{0}_{0,15}$ at $t = 15$ and other special cases as shown later.

The content of stage $s^{95-\alpha}_{\alpha}$ can become 0 by flipping the content of the stage s^{95}_{0} to 0 during the α iterations. The content of the stage s^{95}_{α} should be 1. These can be achieved by two steps. Firstly, the content of the stage s^{95}_{0} can be flipped after 15 iterations, due to the next output stage of the S-box s^{80}_{15}. That requires that the S-box output $y^{0}_{0,15} = 1$, so that $s^{80}_{15} \oplus y^{0}_{6,15} = 0$. Secondly, to ensure the content of stage s^{95}_{α} is 1 after α iterations, the content of the stage $s^{95+\alpha}_{0}$ should be 1. The content of stages $s^{95+\alpha-1}_{0}$ to s^{96}_{0} should be zeros and the output stages that lie between s^{95} and s^{0} should be maintained to meet the required contents, zeros, during the α iterations. Based on these requirements, it is impossible to get slid pairs before $t < 15$.

For $\alpha \geq 15$ iterations, it may be possible that slid pairs occur. If $\alpha = 15$, then the content of stage s^{110}_{0} should be 1 as it will be shifted to s^{95}_{15} after 15 iterations. Note that this value of 1 will pass the input stages s^{105} at $t = 5$ and s^{98} at $t = 12$. The input stage s^{105} should be considered where the output stages $(y^{11}_{6,t}, y^{9}_{6,t}, y^{4}_{6,t}, y^{2}_{6,t}, y^{1}_{6,t}, y^{0}_{6,t})$ are zeros. The content of the input stage s^{98} will be 1 at $t = 12$ and its output will feed back at $t = 19$. This is after getting the required slid pair but will affect the content of the pipeline.

We applied exhaustive search over the inputs and outputs of the S-box to find appropriate inputs and outputs to meet the required conditions. For $\alpha = 15$, there is no input value to the S-box with $(x^{9}_{t}, \ldots, x^{1}_{t}) = (0, \ldots, 0)$ and $x^{10}_{t} = 1$ that gives $(y^{11}_{6,t}, y^{9}_{6,t}, y^{4}_{6,t}, y^{2}_{6,t}, y^{1}_{6,t}, y^{0}_{6,t}) = (0, \ldots, 0)$. Therefore more iterations are required to obtain a slid pair. Additionally, the only input to the S-box that satisfies the condition $(x_t^{10}, \ldots, x_t^1) = (0, \ldots, 0)$ is the 16 zeros $(0, \ldots, 0)$ inputs that gives 16 output of zeros $(0, \ldots, 0)$ at any time t. So, this type of input cannot flip the content of s^{80}. Likewise, it is not possible to satisfy the conditions for a slid pair to occur at $t = 16$.

However, for $\alpha \geq 17$ iterations, slid pairs are possible using specific conditions between the outputs of the S-box. In this case we look for two output stages with the shortest distance between them, namely s^{11} and s^{17}. These conditions are applied to the input and output of the S-box during the α iterations. According to the Sfinks' format pattern, the input of the S-box $(x_t^{10}, \ldots, x_t^1)$ must be zeros $(0, \ldots, 0)$ during the affected iterations (from $0 \leq t \leq (\alpha - 7)$ except some iterations). For example, the content of the input stage $s_{\alpha-10}^{105}$ will be 1, which is the input bit of S-box $x_{\alpha-10}^{10} = 1$. These conditions of the input of the S-box apply up to $t = \alpha - 7$. The output of the S-box during the α iterations should maintain the Sfinks' pattern at $t = \alpha$. Therefore, the output bit of the S-box $(y_{0,t}^{11}, y_{0,t}^{9}, y_{0,t}^{4}, y_{0,t}^{2}, y_{0,t}^{1}, y_{0,t}^{0})$ should be zeros. There are some exceptions for these conditions. For example, at time $t = 15$ the $y_{0,15}^{0}$ should have 1 to flip the value of s_{15}^{80} to 0. As well, it may be required to flip some bits and flip them again to 0 during the α iterations. This process is according to the available input-output pairs of the S-box that follow the required conditions as shown later.

The new-bit of the shift register s_t^{255} is an input of the S-box, x_t^{16}. Therefore, it introduces another condition for the linear feedback function of the shift register. If the required new-bit of the shift register is 0 and the real new-bit is 1, there is freedom to flip any tap-bit of the linear feedback function to get the required bit without changing any other condition.

The cases $\alpha = 17$, 18 and 19 are investigated in the following section. A general method of generating slid pairs can be applied until $\alpha = 23$ (by fixing 56 key bits and 45 IV bits). After that, the situation becomes more complicated because the interaction between the linear feedback function and the S-box output stages is more complex. Another method may be required to identify slid pairs for higher values of α.

Limitations of this work. The analysis above considered only the contents of the shift register, S, and neglected the content of the pipeline. However the format for the loaded state of Sfinks includes the pipeline that contains all zeros. It is impossible to obtain a slid pair with all zeros in the pipeline. To clarify this issue, note that if a slid pair occurs at $t = \alpha$ then s^{95} must be 1 at $t = \alpha$ and hence s^{98} will be 1 at $t = \alpha - 3$. However, the stage s^{98} is an input stage for S-box, and therefore, the S-box output at $t = \alpha - 3$ cannot be all zeros. Thus at least one entry in the pipeline at $t = \alpha$ must be non-zero.

4.3.2 Analysis of Slid Pairs

Procedures

This section describes the procedures used to obtain the first occurrence of slid pairs, and the simulation of the Sfinks cipher to obtain slid pairs at time $\alpha = 17$, 18 and 19. This simulation requires an exhaustive search over the inputs and outputs of the S-box. The procedures used to obtain the following result are:

- Use the published C source code of Sfinks [75] to identify the required inputs and outputs of the S-box to satisfy the conditions for $\alpha = 17$, 18 and 19, as shown later.
- Based on the output of the previous point, generate the two internal state with difference of α steps.
- Based on the output of the previous point, determine the specifc values of the the first key-IV pair, (K, IV).
- Generate the relationship between the first key-IV, (K, IV), and second key-IV, (K', IV$'$), pairs.

slid pair at $\alpha = 17$.

As described above the first slid pair can occur at $\alpha = 17$. The following steps and processes are required to obtain the second pair.

- A specific input is applied to the S-box at $t = 1$ to obtain $y^1_{6,1} = 1$. So, the input to the S-box satisfies $(x^{10}_1, \ldots, x^1_1) = (0, \ldots, 0)$ and the output must have $(y^{11}_{6,1}, y^9_{6,1}, y^4_{6,1}, y^2_{6,1}, y^0_{6,1}) = (0, \ldots, 0)$ and $y^1_{6,1} = 1$. This will flip s^{17}_8 to 1.
- For $2 \leq t \leq 6$, the input to the S-box must satisfy $(x^{10}_t, \ldots, x^1_t) = (0, \ldots, 0)$ and give $(y^{11}_{6,t}, y^9_{6,t}, y^4_{6,t}, y^2_{6,t}, y^1_{6,t}, y^0_{6,t}) = (0, \ldots, 0)$.
- The value of $s^{17}_8 = 1$ will be shifted to s^{11}_{14} after 6 iterations.

- At time $t = 7$, a specific input to the S-box is required so that its output will flip the value of s_{14}^{11} to 0 at $t = 14$ with the input to the S-box $(x_7^9, \dots, x_7^1) = (0, \dots, 0)$ and $x_7^{10} = 1$ and the output $(y_{6,7}^9, y_{6,7}^4, y_{6,7}^2, y_{6,7}^1, y_{6,7}^0) = (0, \dots, 0)$ and $y_{6,7}^{11} = 1$. This output will result in flipping the s_{14}^{11} to 0.
- At $t = 8$, the input to the S-box should be specified to be $(x_8^1, \dots, x_8^{10}) = (0, \dots, 0)$. This input must result in the output $(y_{6,8}^{11}, y_{6,8}^9, y_{6,8}^4, y_{6,8}^2, y_{6,8}^1) = (0, \dots, 0)$ and $y_{6,8}^0 = 1$. At $t = 15$, it will flip the value of s_{15}^{80} to 0.
- For $9 \leq t \leq 10$, the following input to the S-box is applied to insure $(x_t^{10}, \dots, x_t^1) = (0, \dots, 0)$ that should result in $(y_{6,t}^{11}, y_{6,t}^9, y_{6,t}^4, y_{6,t}^2, y_{6,t}^1, y_{6,t}^0) = (0, \dots, 0)$.
- After $t = 10$, it is not important to restrict the input to the S-box.
- To maintain the new-bit, s_t^{255} for $t \geq 1$, it requires 10 tap-bits of the linear feedback function to be manipulated as necessary. The tap-bits are chosen to be s_0^{125} to s_0^{134} that carry k_{29} to k_{38}, respectively.

Table 4.7 shows an exhaustive search of the required input/output pairs of the S-box that satisfy the above procedure. The c_1, c_2, c_3, b_1 and b_2 are for later reference in the next section.

Table 4.7: Input/output of the S-box to get the slid pairs at $\alpha = 17$

	Input		Output	
time	Value		time	Value
1	0011110000000000	$\Longrightarrow$	8	0110000100000010
7	$c_1 = 0001101000000000$	$\Longrightarrow$	14	0000110001101000
	$c_2 = 1010111000000000$			1000110011000000
	$c_3 = 1100111000000000$			0011110100100000
8	$b_1 = 1000000000000000$	$\Longrightarrow$	15	1010000101101001
	$b_2 = 0111100000000000$			0011000010000001

slid pair at $\alpha = 18$.

As presented above for the slid pairs at $\alpha = 17$, it is possible to obtain other slid pairs at $\alpha = 18$. The following steps and processes are applied to get the slid pair.

- From $t = 1$, the following input to the S-box is applied to insure the $(x_1^{10}, \dots, x_1^1) = (0, \dots, 0)$ that will insure $(y_{6,1}^{11}, y_{6,1}^{9}, y_{6,1}^{4}, y_{6,1}^{2}, y_{6,1}^{1}, y_{6,1}^{0}) = (0, \dots, 0)$.
- Specific inputs are applied to the S-box at $t = 2$ to obtain $y_{6,2}^1 = 1$, at $t = 9$ the input to the S-box is $(x_t^{10}, \dots, x_t^1) = (0, \dots, 0)$ and the output is $(y_{6,t}^{11}, y_{6,t}^{9}, y_{6,t}^{4}, y_{6,t}^{2}, y_{6,t}^{0}) = (0, \dots, 0)$ and $y_{6,t}^1 = 1$. This will flip s_9^{17} to 1.
- The value of $s_9^{17} = 1$ will be shifted to s_{15}^{11} after 6 iterations.
- For $3 \le t \le 7$, the following inputs are applied to the S-box to insure $(x_t^{10}, \dots, x_t^1) = (0, \dots, 0)$ to obtain $(y_{6,t}^{11}, y_{6,t}^{9}, y_{6,t}^{4}, y_{6,t}^{2}, y_{6,t}^{1}, y_{6,t}^{0}) = (0, \dots, 0)$.
- At time $t = 8$, a specific input to the S-box is required so that its output will flip the value of s_{15}^{11} and s_{15}^{80} to 0 at $t = 15$ with the input to the S-box $(x_8^{9}, \dots, x_8^1) = (0, \dots, 0)$ and $x_8^{10} = 1$ and the output $(y_{6,8}^{9}, y_{6,8}^{4}, y_{6,8}^{2}, y_{6,8}^{1}) = (0, \dots, 0)$ $y_{6,8}^0 = 1$ and $y_{6,8}^{11} = 1$. This output will result in flipping s_{15}^{11} and s_{15}^{80} to 0.
- For $9 \le t \le 11$, the following input is applied to the S-box to insure $(x_t^{10}, \dots, x_t^1) = (0, \dots, 0)$ that will result in $(y_{6,t}^{11}, y_{6,t}^{9}, y_{6,t}^{4}, y_{6,t}^{2}, y_{6,t}^{1}, y_{6,t}^{0}) = (0, \dots, 0)$.
- After $t = 11$, it is not important to restrict the input to the S-box.
- To maintain the new-bit, s_t^{255} for $t \ge 1$, it requires 11 tap-bits of the linear feedback function to be manipulated as necessary. The tap-bits are chosen to be s_0^{212} to s_0^{222} that carry v_{36} to v_{46}, respectively.

Table 4.8 gives a result of an exhaustive search of the required input/output pairs of the S-box that satisfy the above procedure.

Table 4.8: Input/output of the S-box to get the slid pairs at $\alpha = 18$

Input			Output	
time	Value		time	Value
2	0011110000000000	$\Longrightarrow$	9	0110000100000010
8	$d_1 =$ 1011011000000000 $d_2 =$ 0000111000000000	$\Longrightarrow$	15	1000100000001001 1010100101001001

slid pair at $\alpha = 19$.

This case is similar to the slid pair at $\alpha = 17$ with some difference in the tap positions. It is possible to obtain slid pairs at time $\alpha = 19$ as follow.

- For $1 \leq t \leq 2$, the zeros inputs are applied to the S-box to insure $(x_t^{10}, \ldots, x_t^1) = (0, \ldots, 0)$ to obtain $(y_{6,t}^{11}, y_{6,t}^9, y_{6,t}^4, y_{6,t}^2, y_{6,t}^1, y_{6,t}^0) = (0, \ldots, 0)$.
- A specific input is applied to the S-box at $t = 3$ to obtain $y_{6,3}^1 = 1$. So , the input to the S-box is $(x_3^{10}, \ldots, x_3^1) = (0, \ldots, 0)$ and the output is $(y_{6,3}^{11}, y_{6,3}^9, y_{6,3}^4, y_{6,3}^2, y_{6,3}^0) = (0, \ldots, 0)$ and $y_{6,3}^1 = 1$. This will flip s_{10}^{17} to 1 at time $t = 10$.
- For $4 \leq t \leq 7$, the zeros inputs are applied to the S-box to insure $(x_t^{10}, \ldots, x_t^1) = (0, \ldots, 0)$ to obtain $(y_{6,t}^{11}, y_{6,t}^9, y_{6,t}^4, y_{6,t}^2, y_{6,t}^1, y_{6,t}^0) = (0, \ldots, 0)$.
- At $t = 8$, the input to the S-box should be specified to be $(x_8^1, \ldots, x_8^{10}) = (0, \ldots, 0)$. This input will result in the output $(y_{6,8}^{11}, y_{6,8}^9, y_{6,8}^4, y_{6,8}^2, y_{6,8}^1) = (0, \ldots, 0)$ and $y_{6,8}^0 = 1$. At $t = 15$, it will flip the value of s_{15}^{80} to 0.
- The value of $s_{10}^{17} = 1$ will be shifted to s_{16}^{11} after 6 iterations.
- At time $t = 9$, a specific input to the S-box is required so that its output will flip the value of s_{16}^{11} to 0. The input to the S-box are $(x_9^9, \ldots, x_9^1) = (0, \ldots, 0)$ and $x_9^{10} = 1$ and the output $(y_{6,9}^9, y_{6,9}^4, y_{6,9}^2, y_{6,9}^1, y_{6,9}^0) = (0, \ldots, 0)$ and $y_{6,9}^{11} = 1$. This output will result in flipping s_{16}^{11} to 0 at time $t = 16$.
- For $10 \leq t \leq 12$, the zeros input is applied to the S-box to insure that $(x_t^{10}, \ldots, x_t^1) = (0, \ldots, 0)$ to get $(y_{6,t}^{11}, y_{6,t}^9, y_{6,t}^4, y_{6,t}^2, y_{6,t}^1, y_{6,t}^0) = (0, \ldots, 0)$.

- After $t = 12$, it is not important to restrict the input to the S-box.
- To maintain the new-bit, s_t^{255} for $t \geq 1$, it requires 12 tap-bits of the linear feedback function to be manipulated as necessary. The tap-bits are chosen to be s_0^{212} to s_0^{223} that carry v_{36} to v_{47}, respectively.

As shown above, Table 4.9 shows all the possible input/output values to the S-box using an exhaustive search of the required of the S-box that satisfy the above procedure.

Table 4.9: Input/output of the S-box to get the slid pairs at $\alpha = 19$

Input			Output	
time	Value		time	Value
3	0011110000000000	$\Longrightarrow$	10	0110000100000010
8	$b_1 =$ 1000000000000000	$\Longrightarrow$	15	1010000101101001
	$b_2 =$ 0111100000000000			0011000010000001
9	$e_1 =$ 0001101000000000	$\Longrightarrow$	16	0000110001101000
	$e_2 =$ 1010111000000000			1000110011000000
	$e_3 =$ 1100111000000000			0011110100100000

4.3.3 Findings and Results

This section focuses on the result of the analysis of slid pairs according to the previous procedures that are applied to the Sfinks cipher. The findings of slid pairs are presented after different number of iterations at $t = 17$, 18 and 19. At each specific number of iterations, the slid pairs have specific conditions and relationship between these slid key-IV pairs. To remind the reader, $y_{j,t}^i$ denotes the i-th bit of j-th word of the memory at time t, for $i \in \{0, 1, \ldots, 15\}$.

slid pair at $t = 17$.

The slid pair is found for a key-IV with conditions. These conditions to get another key-IV pair at $\alpha = 17$ are to have specific values for some key-IV bits. There are 40 key bits and 30 IV bits respectively which should have specific values as shown below.

$$\begin{aligned}
k_i = 0,\ i \in \{&75, 74, 71, 70, 69, 68, 67, 48, 47, 46, 44, 43,\\
&42, 41, 40, 19, 18, 17, 15, 14, 13, 12, 11,\\
&10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0\}\\
k_i = 1,\ i \in \{&72, 66, 39, 16\}
\end{aligned}$$

$$(k_{73}, k_{45}) = \begin{cases}
(0,0) & \text{for } x_8 = b_1 \text{ and } x_7 = c_1\\
(0,1) & \text{for } x_8 = b_1 \text{ and } x_7 = c_2\\
(0,1) & \text{for } x_8 = b_1 \text{ and } x_7 = c_3\\
(1,0) & \text{for } x_8 = b_2 \text{ and } x_7 = c_1\\
(1,1) & \text{for } x_8 = b_2 \text{ and } x_7 = c_2\\
(1,1) & \text{for } x_8 = b_2 \text{ and } x_7 = c_3
\end{cases}$$

$$\begin{aligned}
v_i = 0,\ i \in \{&78, 77, 74, 73, 72, 71, 70, 69, 61, 60, 57, 56,\\
&55, 54, 53, 27, 26, 23, 22, 21, 20, 19\}\\
v_i = 1,\ i \in \{&52, 18\}
\end{aligned}$$

$$(v_{76}, v_{75}, v_{59}, v_{58}, v_{25}, v_{24}) = \begin{cases}
(0,0,0,0,0,1) & \text{for } x_8 = b_1 \text{ and } x_7 = c_1\\
(0,0,0,1,0,0) & \text{for } x_8 = b_1 \text{ and } x_7 = c_2\\
(0,1,0,0,0,0) & \text{for } x_8 = b_1 \text{ and } x_7 = c_3\\
(1,0,1,0,1,1) & \text{for } x_8 = b_2 \text{ and } x_7 = c_1\\
(1,0,1,1,1,0) & \text{for } x_8 = b_2 \text{ and } x_7 = c_2\\
(1,1,1,0,1,0) & \text{for } x_8 = b_2 \text{ and } x_7 = c_3
\end{cases}$$

There are 40 and 50 free key and IV bits respectively as shown below.

$$\begin{aligned}
k_i,\ i \in \{&79, 78, 77, 76, 65, 64, 63, 62, 61, 60, 59, 58, 57, 56,\\
&55, 54, 53, 52, 51, 50, 49, 38, 37, 36, 35, 34, 33,\\
&32, 31, 30, 29, 28, 27, 26, 25, 24, 23, 22, 21, 20\}
\end{aligned}$$

$$\begin{aligned} v_i,\ i \in \{&79, 68, 67, 66, 65, 64, 63, 62, 51, 50, 49, 48, 47, 46,\\ &45, 44, 43, 42, 41, 40, 39, 38, 37, 36, 35, 34, 33, 32,\\ &31, 30, 29, 28, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7,\\ &6, 5, 4, 3, 2, 1, 0\} \end{aligned}$$

There are 6 possibilities of the input to the S-box to satisfy the required condition to get slid pairs at $t = 17$. The second key-IV, (K', IV'), pair can be expressed in the form of the first key-IV, (K, IV), pair as follow. Note: Terms printed in boldface are known bits, the values of which depend on the input of the S-box as shown in Table 4.10.

$$\begin{aligned}
v'_{79} &= v_{52} \oplus \mathbf{y^{5}_{6,8}} \oplus v_{34} \oplus v_{32} \oplus v_{27} \oplus v_3 \oplus k_{71} \oplus k_{45} \oplus k_{35} \oplus k_{27} \oplus k_5\\
v'_{78} &= v_{51} \oplus \mathbf{y^{5}_{6,7}} \oplus v_{33} \oplus v_{31} \oplus v_{26} \oplus v_2 \oplus k_{70} \oplus k_{44} \oplus k_{34} \oplus k_{26} \oplus k_4\\
v'_{77} &= v_{50} \oplus v_{32} \oplus v_{30} \oplus v_{25} \oplus v_1 \oplus k_{69} \oplus k_{43} \oplus k_{33} \oplus k_{25} \oplus k_3\\
v'_{76} &= v_{49} \oplus v_{31} \oplus v_{29} \oplus v_{24} \oplus v_0 \oplus k_{68} \oplus k_{42} \oplus k_{32} \oplus k_{24} \oplus k_2\\
v'_{75} &= v_{48} \oplus v_{30} \oplus v_{28} \oplus v_{23} \oplus k_{79} \oplus k_{67} \oplus k_{41} \oplus k_{31} \oplus k_{23} \oplus k_1\\
v'_{74} &= v_{47} \oplus v_{29} \oplus v_{27} \oplus v_{22} \oplus k_{78} \oplus k_{66} \oplus k_{40} \oplus k_{30} \oplus k_{22} \oplus k_0 \oplus 1\\
v'_{73} &= v_{46} \oplus v_{28} \oplus v_{26} \oplus v_{21} \oplus k_{77} \oplus k_{65} \oplus k_{39} \oplus k_{29} \oplus k_{21} \oplus 1\\
v'_{72} &= v_{45} \oplus v_{27} \oplus v_{25} \oplus v_{20} \oplus k_{76} \oplus k_{64} \oplus k_{38} \oplus k_{28} \oplus k_{20} = 0\\
v'_{71} &= v_{44} \oplus v_{26} \oplus v_{24} \oplus v_{19} \oplus k_{75} \oplus k_{63} \oplus k_{37} \oplus k_{27} \oplus k_{19} \oplus 1 = 0\\
v'_{70} &= v_{43} \oplus v_{25} \oplus v_{23} \oplus v_{18} \oplus k_{74} \oplus k_{62} \oplus k_{36} \oplus k_{26} \oplus k_{18} = \mathbf{a_1}\\
v'_{69} &= v_{42} \oplus v_{24} \oplus v_{22} \oplus v_{17} \oplus k_{73} \oplus k_{61} \oplus k_{35} \oplus k_{25} \oplus k_{17} = \mathbf{a_2}\\
v'_{68} &= v_{41} \oplus v_{23} \oplus v_{21} \oplus v_{16} \oplus k_{72} \oplus k_{60} \oplus k_{34} \oplus k_{24} \oplus 1 = 0\\
v'_{67} &= v_{40} \oplus v_{22} \oplus v_{20} \oplus v_{15} \oplus k_{71} \oplus k_{59} \oplus k_{33} \oplus k_{23} \oplus k_{15} = 0\\
v'_{66} &= v_{39} \oplus v_{21} \oplus v_{19} \oplus v_{14} \oplus k_{70} \oplus k_{58} \oplus k_{32} \oplus k_{22} \oplus k_{14} = 0\\
v'_{65} &= v_{38} \oplus v_{20} \oplus v_{18} \oplus v_{13} \oplus k_{69} \oplus k_{57} \oplus k_{31} \oplus k_{21} \oplus k_{13} = 0\\
v'_{64} &= v_{37} \oplus v_{19} \oplus v_{17} \oplus v_{12} \oplus k_{68} \oplus k_{56} \oplus k_{30} \oplus k_{20} \oplus k_{12} = 0\\
v'_{63} &= v_{36} \oplus v_{18} \oplus v_{16} \oplus v_{11} \oplus k_{67} \oplus k_{55} \oplus k_{29} \oplus k_{19} \oplus k_{11} = 0
\end{aligned}$$

$v'_{62} = v_{79}$	$v'_{53} = v_{70} \oplus \mathbf{y^{8}_{6,7}}$	$v'_{44} = v_{61} = 0$	$v'_{35} = 1 \oplus \mathbf{y^{5}_{6,8}}$
$v'_{61} = v_{78} = 0$	$v'_{52} = v_{69} = 0$	$v'_{43} = v_{60} = 0$	$v'_{34} = v_{51} \oplus \mathbf{y^{5}_{6,7}}$
$v'_{60} = v_{77} = 0$	$v'_{51} = v_{68}$	$v'_{42} = \mathbf{v_{59}}$	$v'_{33} = v_{50}$
$v'_{59} = \mathbf{v_{76}}$	$v'_{50} = v_{67}$	$v'_{41} = \mathbf{v_{58}}$	$v'_{32} = v_{49}$
$v'_{58} = \mathbf{v_{75}}$	$v'_{49} = v_{66}$	$v'_{40} = v_{57} = 0$	$v'_{31} = v_{48}$
$v'_{57} = v_{74} = 0$	$v'_{48} = v_{65}$	$v'_{39} = v_{56} = 0$	$v'_{30} = v_{47}$
$v'_{56} = v_{73} = 0$	$v'_{47} = v_{64} \oplus 1$	$v'_{38} = v_{55} = 0$	$v'_{29} = v_{46}$
$v'_{55} = v_{72} = 0$	$v'_{46} = v_{63}$	$v'_{37} = v_{54} = 0$	$v'_{28} = v_{45}$
$v'_{54} = v_{71} \oplus \mathbf{y^{8}_{6,8}}$	$v'_{45} = v_{62}$	$v'_{36} = v_{53} = 0$	$v'_{27} = v_{44}$

$v'_{26} = v_{43} \oplus \mathbf{y^{12}_{6,8}}$	$k'_{79} = v_{16}$	$k'_{52} = k_{69} = 0$	$k'_{25} = k_{42} = 0$
$v'_{25} = v_{42} \oplus \mathbf{y^{12}_{6,7}}$	$k'_{78} = v_{15}$	$k'_{51} = k_{68} = 0$	$k'_{24} = k_{41} = 0$
$v'_{24} = v_{41}$	$k'_{77} = v_{14}$	$k'_{50} = k_{67} = 0$	$k'_{23} = k_{40} = 0$
$v'_{23} = v_{40}$	$k'_{76} = v_{13}$	$k'_{49} = k_{66} = 1$	$k'_{22} = k_{39} = 1$
$v'_{22} = v_{39}$	$k'_{75} = v_{12} \oplus 1$	$k'_{48} = k_{65}$	$k'_{21} = k_{38}$
$v'_{21} = v_{38}$	$k'_{74} = v_{11} \oplus \mathbf{y^{13}_{6,7}}$	$k'_{47} = k_{64}$	$k'_{20} = k_{37} \oplus \mathbf{y^{6}_{6,8}}$
$v'_{20} = v_{37}$	$k'_{73} = v_{10}$	$k'_{46} = k_{63}$	$k'_{19} = k_{36} \oplus \mathbf{y^{6}_{6,7}}$
$v'_{19} = v_{36}$	$k'_{72} = v_{9}$	$k'_{45} = k_{62}$	$k'_{18} = k_{35}$
$v'_{18} = v_{35}$	$k'_{71} = v_{8}$	$k'_{44} = k_{61}$	$k'_{17} = k_{34}$
$v'_{17} = v_{34}$	$k'_{70} = v_{7}$	$k'_{43} = k_{60}$	$k'_{16} = k_{33}$
$v'_{16} = v_{33}$	$k'_{69} = v_{6}$	$k'_{42} = k_{59}$	$k'_{15} = k_{32}$
$v'_{15} = v_{32}$	$k'_{68} = v_{5} \oplus 1$	$k'_{41} = k_{58}$	$k'_{14} = k_{31}$
$v'_{14} = v_{31}$	$k'_{67} = v_{4}$	$k'_{40} = k_{57}$	$k'_{13} = k_{30} \oplus \mathbf{y^{15}_{6,8}}$
$v'_{13} = v_{30}$	$k'_{66} = v_{3}$	$k'_{39} = k_{56}$	$k'_{12} = k_{29} \oplus \mathbf{y^{15}_{6,7}}$
$v'_{12} = v_{29}$	$k'_{65} = v_{2}$	$k'_{38} = k_{55}$	$k'_{11} = k_{28}$
$v'_{11} = v_{28}$	$k'_{64} = v_{1}$	$k'_{37} = k_{54} \oplus 1$	$k'_{10} = k_{27}$
$v'_{10} = v_{27} = 0$	$k'_{63} = v_{0}$	$k'_{36} = k_{53}$	$k'_{9} = k_{26}$
$v'_{9} = v_{26} = 0$	$k'_{62} = k_{79}$	$k'_{35} = k_{52}$	$k'_{8} = k_{25}$
$v'_{8} = \mathbf{v_{25}}$	$k'_{61} = k_{78}$	$k'_{34} = k_{51}$	$k'_{7} = k_{24}$
$v'_{7} = \mathbf{v_{24}}$	$k'_{60} = k_{77}$	$k'_{33} = k_{50}$	$k'_{6} = k_{23}$
$v'_{6} = v_{23} = 0$	$k'_{59} = k_{76}$	$k'_{32} = k_{49}$	$k'_{5} = k_{22}$
$v'_{5} = v_{22} = 0$	$k'_{58} = k_{75} = 0$	$k'_{31} = k_{48} = 0$	$k'_{4} = k_{21}$
$v'_{4} = v_{21} = 0$	$k'_{57} = k_{74} = 0$	$k'_{30} = k_{47} = 0$	$k'_{3} = k_{20}$
$v'_{3} = v_{20} = 0$	$k'_{56} = \mathbf{k_{73}}$	$k'_{29} = k_{46} = 0$	$k'_{2} = k_{19} = 0$
$v'_{2} = v_{19} = 0$	$k'_{55} = k_{72} \oplus 1 = 0$	$k'_{28} = \mathbf{k_{45}}$	$k'_{1} = k_{18} = 0$
$v'_{1} = 1 \oplus \mathbf{y^{3}_{6,8}}$	$k'_{54} = k_{71} = 0$	$k'_{27} = k_{44} = 0$	$k'_{0} = k_{17} = 0$
$v'_{0} = v_{17} \oplus \mathbf{y^{3}_{6,7}}$	$k'_{53} = k_{70} = 0$	$k'_{26} = k_{43} = 0$	

Note that 32 bits of v' and 23 bits of k' are fixed by the above relations.

slid pair at $\alpha = 18$.

For slid pair at $\alpha = 18$, the conditions to get another key-IV pair are to have specific values for some key-IV bits at $t = 0$. There are 43 and 33 bits of key and IV respectively which must have fixed values as shown below.

Table 4.10: Six alternative inputs and outputs to the S-box at $\alpha = 17$

Conditions		Input										Output												
x_8	x_7	$\mathbf{a_1}$	$\mathbf{v_{76}}$	$\mathbf{v_{59}}$	$\mathbf{v_{25}}$	$\mathbf{k_{73}}$	$\mathbf{a_2}$	$\mathbf{v_{75}}$	$\mathbf{v_{58}}$	$\mathbf{v_{24}}$	$\mathbf{k_{45}}$	$\mathbf{y^{5}_{6,8}}$	$\mathbf{y^{8}_{6,8}}$	$\mathbf{y^{12}_{6,8}}$	$\mathbf{y^{3}_{6,8}}$	$\mathbf{y^{6}_{6,8}}$	$\mathbf{y^{15}_{6,8}}$	$\mathbf{y^{5}_{6,7}}$	$\mathbf{y^{8}_{6,7}}$	$\mathbf{y^{12}_{6,7}}$	$\mathbf{y^{3}_{6,7}}$	$\mathbf{y^{13}_{6,7}}$	$\mathbf{y^{6}_{6,7}}$	$\mathbf{y^{15}_{6,7}}$
b_1	c_1	1	0	0	0	0	0	0	0	1	0	1	1	0	1	1	1	1	0	0	1	0	1	0
b_1	c_2	1	0	0	0	0	1	0	1	0	1	1	1	0	1	1	1	0	0	0	0	0	1	1
b_1	c_3	1	0	0	0	0	1	1	0	0	1	1	1	0	1	1	1	1	1	1	0	1	0	0
b_2	c_1	0	1	1	1	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	1	0	1	0
b_2	c_2	0	1	1	1	1	1	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	1	1
b_2	c_3	0	1	1	1	1	1	1	0	0	1	0	0	1	0	0	0	1	1	1	0	1	0	0

$$
\begin{aligned}
k_i = 0,\ i \in &\{76, 75, 74, 72, 71, 70, 69, 68, 66, 49, 48, 47, 45, 44,\\
&43, 42, 41, 39, 20, 19, 18, 16, 15, 14, 13, 12, 11, 10,\\
&9, 8, 7, 6, 5, 4, 3, 2, 1, 0\}\\
k_i = 1,\ i \in &\{67, 46, 40, 17\}
\end{aligned}
$$

$$
k_{73} = \begin{cases} 0 & \text{for } x_8 = d_1 \\ 1 & \text{for } x_8 = d_2 \end{cases}
$$

$$
\begin{aligned}
v_i = 0,\ i \in &\{79, 78, 77, 76, 75, 74, 73, 72, 71, 70, 69, 62, 61, 60\\
&58, 57, 56, 55, 54, 52, 28, 27, 26, 24, 23, 22,\\
&21, 20, 18\}\\
v_i = 1,\ i \in &\{53, 19\}
\end{aligned}
$$

$$(v_{59}, v_{25}) = \begin{cases} (1,1) & \text{for } x_8 = d_1 \\ (0,0) & \text{for } x_8 = d_2 \end{cases}$$

There are 37 and 47 free key and IV bits respectively as shown.

$$k_i,\, i \in \{79, 78, 77, 65, 64, 63, 62, 61, 60, 59, 58, 57, 56, 55, 54,\\ 53, 52, 51, 50, 38, 37, 36, 35, 34, 33, 32, 31, 30, 29, 28,\\ 27, 26, 25, 24, 23, 22, 21\}$$

$$v_i,\, i \in \{68, 67, 66, 65, 64, 63, 51, 50, 49, 48, 47, 46, 45, 44, 43,\\ 42, 41, 40, 39, 38, 37, 36, 35, 34, 33, 32, 31, 30, 29, 17,\\ 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0\}$$

There are two possibilities of the input to the S-box to satisfy the required conditions to get a slid pair at $\alpha = 18$. The second key-IV pair, $(K',\ \text{IV}')$, can be expressed in term of the first key-IV, $(K,\ \text{IV})$, as follow. Note: terms printed in boldface are known bits, the values of which depend on the input of the S-box as shown in Table 4.11.

$v'_{79} = v_{53} \oplus v_{35} \oplus v_{33} \oplus v_{28} \oplus v_4 \oplus k_{72} \oplus k_{46} \oplus k_{36} \oplus k_{28} \oplus k_6$
$v'_{78} = v_{52} \oplus v_{34} \oplus v_{32} \oplus v_{27} \oplus v_3 \oplus k_{71} \oplus k_{45} \oplus k_{35} \oplus k_{27} \oplus k_5$
$v'_{77} = v_{51} \oplus v_{33} \oplus v_{31} \oplus v_{26} \oplus v_2 \oplus k_{70} \oplus k_{44} \oplus k_{34} \oplus k_{26} \oplus k_4$
$v'_{76} = v_{50} \oplus v_{32} \oplus v_{30} \oplus \mathbf{v_{25}} \oplus v_1 \oplus k_{69} \oplus k_{43} \oplus k_{33} \oplus k_{25} \oplus k_3$
$v'_{75} = v_{49} \oplus v_{31} \oplus v_{29} \oplus v_{24} \oplus v_0 \oplus k_{68} \oplus k_{42} \oplus k_{32} \oplus k_{24} \oplus k_2$
$v'_{74} = v_{48} \oplus v_{30} \oplus v_{28} \oplus v_{23} \oplus k_{79} \oplus k_{67} \oplus k_{41} \oplus k_{31} \oplus k_{23} \oplus k_1 \oplus 1$
$v'_{73} = v_{47} \oplus v_{29} \oplus v_{27} \oplus v_{22} \oplus k_{78} \oplus k_{66} \oplus k_{40} \oplus k_{30} \oplus k_{22} \oplus k_0$
$v'_{72} = v_{46} \oplus v_{28} \oplus v_{26} \oplus v_{21} \oplus k_{77} \oplus k_{65} \oplus k_{39} \oplus k_{29} \oplus k_{21} \oplus 1 = 0$
$v'_{71} = v_{45} \oplus v_{27} \oplus \mathbf{v_{25}} \oplus v_{20} \oplus k_{76} \oplus k_{64} \oplus k_{38} \oplus k_{28} \oplus k_{20} = 0$
$v'_{70} = v_{44} \oplus v_{26} \oplus v_{24} \oplus v_{19} \oplus k_{75} \oplus k_{63} \oplus k_{37} \oplus k_{27} \oplus k_{19} = 0$
$v'_{69} = v_{43} \oplus \mathbf{v_{25}} \oplus v_{23} \oplus v_{18} \oplus k_{74} \oplus k_{62} \oplus k_{36} \oplus k_{26} \oplus k_{18} = \mathbf{a_3}$
$v'_{68} = v_{42} \oplus v_{24} \oplus v_{22} \oplus v_{17} \oplus \mathbf{k_{73}} \oplus k_{61} \oplus k_{35} \oplus k_{25} \oplus k_{17} \oplus 1 = 0$
$v'_{67} = v_{41} \oplus v_{23} \oplus v_{21} \oplus v_{16} \oplus k_{72} \oplus k_{60} \oplus k_{34} \oplus k_{24} \oplus k_{16} = 0$
$v'_{66} = v_{40} \oplus v_{22} \oplus v_{20} \oplus v_{15} \oplus k_{71} \oplus k_{59} \oplus k_{33} \oplus k_{23} \oplus k_{15} = 0$
$v'_{65} = v_{39} \oplus v_{21} \oplus v_{19} \oplus v_{14} \oplus k_{70} \oplus k_{58} \oplus k_{32} \oplus k_{22} \oplus k_{14} = 0$
$v'_{64} = v_{38} \oplus v_{20} \oplus v_{18} \oplus v_{13} \oplus k_{69} \oplus k_{57} \oplus k_{31} \oplus k_{21} \oplus k_{13} = 0$
$v'_{63} = v_{37} \oplus v_{19} \oplus v_{17} \oplus v_{12} \oplus k_{68} \oplus k_{56} \oplus k_{30} \oplus k_{20} \oplus k_{12} = 0$
$v'_{62} = v_{36} \oplus v_{18} \oplus v_{16} \oplus v_{11} \oplus k_{67} \oplus k_{55} \oplus k_{29} \oplus k_{19} \oplus k_{11} = 0$

$v'_{61} = v_{79} = 0$

$v'_{60} = v_{78} = 0$

$v'_{59} = v_{77} = 0$

$v'_{58} = v_{76} = 0$

$v'_{57} = v_{75} = 0$

$v'_{56} = v_{74} = 0$

$v'_{55} = v_{73} = 0$

$v'_{54} = v_{72} = 0$

$v'_{53} = v_{71} \oplus \mathbf{y^{8}_{6,8}}$

$v'_{52} = v_{70} = 0$

$v'_{51} = v_{69} = 0$

$v'_{50} = v_{68}$

$v'_{49} = v_{67}$

$v'_{48} = v_{66}$

$v'_{47} = v_{65} \oplus 1$

$v'_{46} = v_{64}$

$v'_{45} = v_{63}$

$v'_{44} = v_{62} = 0$

$v'_{43} = v_{61} = 0$

$v'_{42} = v_{60} = 0$

$v'_{41} = \mathbf{v_{59}}$

$v'_{40} = v_{58} = 0$

$v'_{39} = v_{57} = 0$

$v'_{38} = v_{56} = 0$

$v'_{37} = v_{55} = 0$

$v'_{36} = v_{54} = 0$

$v'_{35} = v_{53} = 1$

$v'_{34} = v_{52} = 0$

$v'_{33} = v_{51}$

$v'_{32} = v_{50}$

$v'_{31} = v_{49}$

$v'_{30} = v_{48}$

$v'_{29} = v_{47}$

$v'_{28} = v_{46}$

$v'_{27} = v_{45}$

$v'_{26} = v_{44}$

$v'_{25} = v_{43}$

$v'_{24} = v_{42}$

$v'_{23} = v_{41}$

$v'_{22} = v_{40}$

$v'_{21} = v_{39}$

$v'_{20} = v_{38}$

$v'_{19} = v_{37}$

$v'_{18} = v_{36}$

$v'_{17} = v_{35}$

$v'_{16} = v_{34}$

$v'_{15} = v_{33}$

$v'_{14} = v_{32}$

$v'_{13} = v_{31}$

$v'_{12} = v_{30}$

$v'_{11} = v_{29}$

$v'_{10} = v_{28} = 0$

$v'_{9} = v_{27} = 0$

$v'_{8} = v_{26} = 0$

$v'_{7} = \mathbf{v_{25}}$

$v'_{6} = v_{24} = 0$

$v'_{5} = v_{23} = 0$

$v'_{4} = v_{22} = 0$

$v'_{3} = v_{21} = 0$

$v'_{2} = v_{20} = 0$

$v'_{1} = v_{19} = 1$

$v'_{0} = v_{18} \oplus 1 = 1$

$k'_{79} = v_{17}$

$k'_{78} = v_{16}$

$k'_{77} = v_{15}$

$k'_{76} = v_{14}$

$k'_{75} = v_{13}$

$k'_{74} = v_{12} \oplus \mathbf{y^{13}_{6,8}}$

$k'_{73} = v_{11}$

$k'_{72} = v_{10}$

$k'_{71} = v_{9}$

$k'_{70} = v_{8}$

$k'_{69} = v_{7}$

$k'_{68} = v_{6} \oplus 1$

$k'_{67} = v_{5}$

$k'_{66} = v_{4}$

$k'_{65} = v_{3}$

$k'_{64} = v_{2}$

$k'_{63} = v_{1}$

$k'_{62} = v_{0}$

$k'_{61} = k_{79}$

$k'_{60} = k_{78}$

$k'_{59} = k_{77}$

$k'_{58} = k_{76} = 0$

$k'_{57} = k_{75} = 0$

$k'_{56} = k_{74} = 0$

$k'_{55} = \mathbf{k_{73}}$

$k'_{54} = k_{72} = 0$

$k'_{53} = k_{71} = 0$

$k'_{52} = k_{70} = 0$

$k'_{51} = k_{69} = 0$

$k'_{50} = k_{68} = 0$

$k'_{49} = k_{67} = 1$

$k'_{48} = k_{66} = 0$

$k'_{47} = k_{65}$

$k'_{46} = k_{64}$

$k'_{45} = k_{63}$

$k'_{44} = k_{62}$

$k'_{43} = k_{61}$

$k'_{42} = k_{60}$

$k'_{41} = k_{59}$

$k'_{40} = k_{58}$

$k'_{39} = k_{57}$

$k'_{38} = k_{56}$

$k'_{37} = k_{55} \oplus 1$

$k'_{36} = k_{54}$

$k'_{35} = k_{53}$

$k'_{34} = k_{52}$

$k'_{33} = k_{51}$

$k'_{32} = k_{50}$

$k'_{31} = k_{49} = 0$

$k'_{30} = k_{48} = 0$

$k'_{29} = k_{47} = 0$

$k'_{28} = k_{46} = 1$

$k'_{27} = k_{45} = 0$

$k'_{26} = k_{44} = 0$

$k'_{25} = k_{43} = 0$

$k'_{24} = k_{42} = 0$

$k'_{23} = k_{41} = 0$

$k'_{22} = k_{40} = 1$

$k'_{21} = k_{39} = 0$

$k'_{20} = k_{38}$

$k'_{19} = k_{37} \oplus \mathbf{y^{6}_{6,8}}$

$k'_{18} = k_{36}$

$k'_{17} = k_{35}$

$k'_{16} = k_{34}$

$k'_{15} = k_{33}$

$k'_{14} = k_{32}$

$k'_{13} = k_{31}$

$k'_{12} = k_{30} \oplus 1$

$k'_{11} = k_{29}$

$k'_{10} = k_{28}$

$k'_{9} = k_{27}$

$k'_{8} = k_{26}$

$k'_{7} = k_{25}$

$k'_{6} = k_{24}$

$k'_{5} = k_{23}$

$k'_{4} = k_{22}$

$k'_{3} = k_{21}$

$k'_{2} = k_{20} = 0$

$k'_{1} = k_{19} = 0$

$k'_{0} = k_{18} = 0$

Note that 43 bits of v' and 25 bits of k' are fixed by the above relations.

slid pair at $\alpha = 19$.

For the slid pair at $\alpha = 19$, there are 47 and 36 bits of key and IV respectively which must take fixed value as follow.

Table 4.11: Two alternative inputs and outputs to the S-box at $\alpha = 18$

Condition	Input				Output		
x_8	$\mathbf{a_3}$	$\mathbf{v_{59}}$	$\mathbf{v_{25}}$	$\mathbf{k_{73}}$	$\mathbf{y_8^{13}}$	$\mathbf{y_8^{8}}$	$\mathbf{y_8^{6}}$
b_1	1	1	1	0	0	0	0
b_2	0	0	0	1	1	1	1

$$\begin{aligned}
k_i = 0, i \in &\{77, 76, 75, 74, 72, 71, 70, 69, 67, 66, 50, 49, 48,\\
&46, 45, 44, 43, 42, 40, 39, 21, 20, 19, 17, 16, \ldots, 0\}\\
k_i = 1, i \in &\{68, 41, 18, 17\}
\end{aligned}$$

$$(k_{73}, k_{47}) = \begin{cases} (0,0) & \text{for } x_8 = b_1 \text{ and } x_9 = e_1 \\ (0,1) & \text{for } x_8 = b_1 \text{ and } x_9 = e_2 \\ (0,1) & \text{for } x_8 = b_1 \text{ and } x_9 = e_3 \\ (1,0) & \text{for } x_8 = b_2 \text{ and } x_9 = e_1 \\ (1,1) & \text{for } x_8 = b_2 \text{ and } x_9 = e_2 \\ (1,1) & \text{for } x_8 = b_2 \text{ and } x_9 = e_3 \end{cases}$$

$$\begin{aligned}
v_i = 0, i \in &\{79, 78, 76, 75, 74, 73, 72, 71, 70, 69, 63, 62, 61, 58,\\
&57, 56, 55, 53, 52, 29, 28, 27, 24, 23, 22, 21, 19, 18\}\\
v_i = 1, i \in &\{54, 20\}
\end{aligned}$$

$$(v_{77}, v_{76}, v_{60}, v_{59}, v_{26}, v_{25}) = \begin{cases} (0,0,0,0,1,0) & \text{for } x_8 = b_1 \text{ and } x_9 = e_1 \\ (0,0,1,0,0,0) & \text{for } x_8 = b_1 \text{ and } x_9 = e_2 \\ (1,0,0,0,0,0) & \text{for } x_8 = b_1 \text{ and } x_9 = e_3 \\ (0,1,0,1,1,1) & \text{for } x_8 = b_2 \text{ and } x_9 = e_1 \\ (0,1,1,1,0,1) & \text{for } x_8 = b_2 \text{ and } x_9 = e_2 \\ (1,1,0,1,0,1) & \text{for } x_8 = b_2 \text{ and } x_9 = e_3 \end{cases}$$

As for the cases $\alpha = 17$ and 18, expressions can again be determined for (K', IV$'$) in terms of (K, IV). These are not reported in this thesis.

All the slid pairs are obtained using computer simulation for all possible inputs and outputs of the S-box. As shown previously, slid pairs occur according to specific conditions at time $t \geq 17$. Slid pairs occur with a limitation, where the pipeline cannot be reset to all zeros.

Shifted Keystream

We now consider the additional constraints which must be satisfied in order for slid pair (K, IV) and (K', IV$'$) to generate shifted keystream sequences. These constraints depend on the degree of the similarity between the state update functions during the initialisation and keystream generation processes.

To obtain a shifted keystream from the slid pairs at $\alpha = 17$, all the output of the S-box for the last 17 iterations of the initialisation process (from $t = 111$ to $t = 128$) should be all zeros to get the linear feedback during these iterations of the initialisation process. Therefore, the input of the S-box from $t = 104$ to $t = 121$ should be zeros. To ensure all the relevent input bits are zeros, there are 187 bits which should be fixed and another 17 bits to be fixed for the linear feedback function. The total fixed bits are 204 bits. The probability of satisfying 204 independent constraints is 2^{-204}. In this case, the free bits at the loaded state are 90 bits. Hence, there are 2^{90} possible slid pairs and the probability of slid pair gives an out of phase keystream is 2^{-204}. Therefore, the expected number of shifted keystream sequences from these slid pairs $= 2^{-204} \times 2^{90} = 2^{-114}$. So, it seems extremely unlikely that any shifted keystream will result from any of the slid pairs.

Similarly, for the slid pairs at $\alpha = 18$ and 19, the last 18 and 19 outputs of the S-box should be zeros respectively. The total fixed bits in these cases are 211 and 218 bits. The probability of satisfying 211 and 218 independent constraints are 2^{-211} and 2^{-218} respectively. In this case, the free bits at the loaded state are 84 and 79 bits. Therefore expected number of keystream out of phase shifted by 18 and 19 bits from any of the relevant slid pairs are $2^{(-211+84)} = 2^{-127}$ and $2^{(-218+79)} = 2^{-139}$.

4.3.4 Attack Procedure

Since Sfinks has an 80-bit key and a 256-bit shift register, it is not feasible to simply guess the whole secret key and then generate the keystream to check whether the guess is correct. However, if it is possible to identify the occurrence of a slid pair (extremely unlikely as discussed previously), this enables us to use the resulting relationship to reduce the number of key bits that need to be guessed, forming the basis for an attack on the cipher. The attack presented in this research uses the properties of the slide attacks. We assume a known-plaintext attack scenario. The following algorithm outlines the attacking procedure.

This algorithm is described for 17-bit shifted versions of keystream. However, it can be extended to other shifts as well. The algorithm depends on two keystreams with unknown secret keys and known IV's such that the resulting keystreams are shifted versions of one another. Note that we are interested here to find the secret keys.

Recalling the specification of the loading phase of Sfinks in Section 4.1, it is impossible to find two shifted keystreams generated by a secret key with different IV's. Therefore, in this case, we are focusing on two keystreams generated by diferent key-IV pairs, (K, IV) and (K', IV$'$). (K, IV)

Attacking Algorithm

Inputs: Pairs of plaintext and ciphertext.

Step 1: From the known-plaintext-ciphertext attack, XOR the plaintext with the corresponding ciphertext to get the keystream sequence.

Step 2: Divide each keystream sequence into separate keystreams. Each keystream corresponds to a different key-IV combination based on the rekying process.

Step 3: Check the similarity of $keystream_i$ and $keystream_j \ll 17$ for every available key-IV pair, using $keystream_i \oplus keystream_j \ll 17$.

- If identical, note the key-IV pairs (K, IV) and (K', IV$'$).
- If not, the algorithm fails for this 17-bit shift.

Step 4: If shifted keystream has been identified, guess and check each possible key, as follows

- Using the relevant equations for $t = 17$ as shown above, where the IV's (IV and IV$'$) are known.

- Due to the slid pairs, there are 40 and 23 bits of the K and K' are known and fixed,
- Guess the 40 free key bits of K in the relevant set of equations as shown above.
- Use these free key bits and the known IV's to calculate the remaining key bits of K' using the related equations as shown above.
- Use the generated secret key K' with the corresponding IV$'$ to generate its keystream using the Sfinks algorithm.
- Compare between the generated keystream and the known keystream.
  - If they are identical, the secret keys K and K' have been found.
  - If not, repeat the process for another guess of K and calculate the K'.

Step 5: Use the secret key K and K' with known IV's to decrypt the entire related ciphertexts.

Outputs: The secret keys K and K'.

Comments on algorithm:

In Step 3, if two keystreams have been identified as shifted keystream by 17 bits, then the guessing process should cover the 40 free key bits instead of the 80 key bits. For each guess, it requires to substitute the guessed key bits into the relevant equations as mentioned above to find the secret key K'. At this time, it is not clear whether the guessed key is correct or not. The proposed key must be verified by generating a keystream that should be same as the known keystream.

If the generated keystream is identical to the known keystream then the secret key is correct, if not use another guess. Note that the above algorithm is for 17-bit shifts. If it fails, then it is possible to try for 18-bit shifts or more.

Attack Complexity: In Step 3, the comparison depends on the available length of the plaintext and its corresponding ciphertext. If an attacker has enough plaintext-ciphertext pairs, it is possible to check the shifted keystreams. As mentioned previously, the probability of obtaining shifted keystreams from a given slid pair is 2^{-204}, and the total number of possible slid pairs is 2^{90}. Then the expected number of shifted keystreams from slid pairs is 2^{-114}. Thus this cipher appears to be secure against slid attacks, but the presence of slid pairs is still a concern.

4.3.5 Analysis of Sfinks with Slight Modification

This section analyses a slightly modified version of Sfinks, where the modification is applied to the padding format of Sfinks during the loading phase. This analysis shows that the effect of a minor change in the padding increases the probability of occurrence of slid pairs and the probability of obtaining shifted keystreams from given slid pairs.

In this modification, we change the pattern of the padding to be zeros for all stages (s^{95} to s^{90}). In other words, we set the content of stage s^{95} to 0 instead of 1. The slid pairs may now occur after only one clock under two conditions. Firstly, the 16-bit input to the S-box must be zero at time $t = 1$. This requires us to fix the values of 8 stages (s_1^{255},s_1^{244}, s_1^{227}, s_1^{193}, s_1^{161}, s_1^{134}, s_1^{105}, s_1^{98}) to be zeros to ensure that the pipeline is still in the initial condition to having a zero pipeline (this avoid the limitation in the analysis of actual design of Sfinks). The last seven of these stages can be expressed in terms of key and IV bits as (k_{66}, k_{39}, k_{10}, k_3, v_{69}, v_{52}, v_{18}). In addition, since $s^{255} = x^{16}$ must be zero, the linear feedback function imposes a condition that the XOR of the following bits must be zero: (k_{68}, k_{56}, k_{30}, k_{20}, k_{12}, v_{34}, v_{19}, v_{17}, v_{12}), so that $s_1^{255} = 0$. Secondly, the content of s_0^{96} must be similar to the padding, that is $s_0^{96} = s_0^{95} = 0$.

Thus, a slid pair with a zeroed pipeline can be obtained by fixing 9 bits out of the 160 key-IV bits. Therefore, the proportion of valid key-IV pairs which leads to these slid pairs is 2^{-9}. Hence there are $2^{160-9} = 2^{151}$ slid pairs for $\alpha = 1$.

A slid pair can lead to an out-of-phase keystream by fixing another 16 bits to ensure that the last iteration of the initialisation process at $t = 128$ is linear. Specifically, we require that all 16 input bits of the S-box at time $t = 121$ should be zeros. The probability of satisfying these 16 independent constraints is 2^{-16}. Therefore, the expected proportion of Key-IV pairs that leads to phase shifted keystream with 1 step is $2^{-9} \times 2^{-16} = 2^{-25}$ and the expected number of out of phase keystream sequences is $2^{151} \times 2^{-16} = 2^{135}$.

We see that a small change in the padding format during the loading phase of the initialisation has hugely increased the probability of obtaining both slid pairs and corresponding shifted keystream. Moreover, the condition of maintaining a zero pipeline is also satisfied in the modified version. As expressed in this section and the analysis of slid pairs of Sfinks stream cipher, there are 2^{151} and 2^{90} out of 2^{160} possible slid key-IV pairs for the modified version and current cipher respectively. The probabilities of a slid pair that gives a shifted keystream are 2^{-16} and 2^{-204} for these versions respectively.

4.4 Weak Key-IV Combinations

Sfinks stream cipher transfers the secret key and IV into the internal state simultaneously and applies a specific padding format. The padding pattern sets stages not filled by key and IV bits to zeros except one specific stage (which is set to one). Therefore, the loaded state will start with a non-zero state. Because of this format, we argue that the probability of obtaining an all-zero internal state (including the memory pipeline) will be very small. In particular, the one in stage s_0^{95} cannot be cancelled until it reaches s_{15}^{80}, by which time it will have passed s_{10}^{85} and contributed to the feedback to s_{11}^{255}. In order to cancel s_{15}^{80}, y_{14}^{80} must equal 1, as there must be at least one non-zero input x_7^i for some $i = 1, \ldots, 16$. But the input stage which contains a one at that time will also need to be canceled by the S-box output at some late time. Although this argument is not exact, it clearly shows that any conditions which lead to an all-zero state in Sfinks must be complex and the likelihood of this occurring must therefore be very low. An exact analysis of this problem is left for future work.

4.5 Summary and Security Impact

This chapter examined the initialisation process of the Sfinks stream cipher. Sfinks uses a 256-bit internal state which is larger than the key-IV size ($80+80 = 160$ bits). The initialisation process of Sfinks loads the secret key and IV into specific stages and fills the remaining stages with pre-determined values. The key-IV is diffused across the entire internal state through a 128 iterations (clocks) using the nonlinear state update function. During this process, each iteration updates the contents of 17 stages of the internal state. During the keystream generation process, the state update function is the linear feedback shift register only, and the output function is nonlinear using one bit from the shift register and another bit from the S-box to form a keystream bit. This chapter investigated two flaws in the Sfinks initialisation process: state convergence and slid pairs. These problems have been discussed in detail in this chapter and are summarized below.

The individual components of the state update function of Sfinks are one-to-one but their combination is not, which results in state convergence during the initialisation process. State convergence occurs during the *diffusion* phase of Sfinks as a result of the combination of the delay in feeding the S-box output back to the shift register and the shift register tap spacing. Specifically, this is caused by the correspondence between the 7-clock feedback delay and the distance of 7 clocking steps between certain input and output stages of the S-box. The combination of these components of the state update function results in an initialisation state update function that is not one-to-one and leads to state convergence. These results show that the state convergence can occur after the first seven iterations of initialisation and the probability of obtaining state convergence is estimated to be $2^{-6.9}$.

Sfinks is designed for use with an 80-bit key and 80-bit IV with specific pattern of loaded state. As the state size (256 bits) is greater than the sum of the key and IV size, it seems reasonable to assume that 2^{160} different keystream could be produced. However, the state converges problem demonstrated in this chapter reduces the number of distinct keystream. We estimate the number of distinct keystream is less than $2^{158.55}$. Although the impact of this convergence on the security of Sfinks is minor (an approximate reduction of 1.45 bit after 128 iterations), it can be avoided entirely by more careful design.

The similarity of iterations of state update function during initialisation process and the format of the loaded state results in slid pairs in the internal state. Although the state update functions of the initialisation and keystream generation processes are different, slid pair which do occur, although with low probability. Thus, slid pairs occur during the *diffusion* phase of the initialisation process of Sfinks. The padding pattern and the format of the state update function defer the first slid pair to 17 iterations ($\alpha = 17$). A known-plaintext attack is applied when $\alpha = 17$ by fixing 70 bits out of 160 key-IV bits with proportion of valid key-IV pairs is 2^{90} out of 2^{160}. The expected probability of shifted keystream sequences from 2^{90} slid pairs (for $\alpha = 17$) is 2^{-114}. Similar result also apply for $\alpha = 18$ and $\alpha = 19$. For the current design of Sfinks, the security impact of slid pairs and shifted keystream is negligible. However, we have shown that the design of padding is essential to achieve this result.

We have also shown that the format specified for the loaded state of cipher can have a strong impact on the probability of obtaining slid pairs and shifted keystreams. Specifically, we considered a minor variation of Sfinks in which the padding used in the loaded state consists entirely of zeros. Note that this format only differs from the actual specification for Sfinks in one bit position. Using this modified loading format, we showed that genuine slid pairs (with a zero pipeline) can occur after one clock, the probability of obtaining a slid pair increases to 2^{-9} and the corresponding probability of obtaining shifted keystream increases to 2^{-16} for a given slid pair. Thus, a small change in the loading format leads to a huge increase in susceptibility to slid pairs compared to the actual version. Therefore, the padding format may result in a dramatic security impact on the cipher.

Chapter 5

Analysis of CSA-SC Initialisation Process

The Digital Video Broadcasting Common Scrambling Algorithm (DVB-CSA) is specified by the European Telecommunications Standards Institute (ETSI) [51]. It has been used for scrambling and encrypting the MPEG-2 transport stream (digital television) since 1994. The algorithm specification was initially protected by a non-disclosure agreement from the ETSI. However, in 2002 the CSA was reverse-engineered and published on the Internet.

The Common Scrambling Algorithm consists of a cascade of block and stream ciphers. To encrypt, the block cipher is applied first followed by the stream cipher. To decrypt, the stream cipher is applied first followed by the block cipher. Both block and stream ciphers use the same 64-bit key. The scope of this research is limited to the initialisation process of the stream cipher. The stream cipher component of the Digital Video Broadcasting Common Scrambling Algorithm is referred to as CSA-SC [16].

The CSA-SC has been described in several different ways, with equivalent representations having different internal state sizes. In Simon's patents [15–17] the internal state consists of 107 bits. Weinmann and Wirt [104] reduced the CSA-SC to a 103-bit internal state by cutting down a memory by 4 bits. Simpson et al [96] presented a model with an 89-bit internal state, by shifting the positions of the inputs to the S-boxes by 1 stage, to remove additional redundant storage memories. All of these models are functionally equivalent. In this research, we use the 89-bit model described in [96].

Analysis of the Common Scrambling Algorithm (CSA) has been presented in a number of papers; some analysed the block cipher [74, 101, 105] and others analysed the stream cipher [96, 104]. Weinmann and Wirt [104] presented an attack based on their reduced model and claimed the complexity of 2^{44} to find the value of the register A. Simpson et al [96] analysed the keystream generation process and showed that the state update function during this phase is invertible and that there exist distinct state cycles for the register A. Their results contradict the claim in [104] and they state directly "the complexity of that state recovery attack is not around 2^{45}, as claimed, but in fact worse than exhaustive key search" with a revised estimate for Weinmann and Wirt approach of $2^{66.7}$. Simpson et al [96] then applied the generic Time-Memory trade-off attacks to the 89-bit state representation and demonstrated state recovery attacks with complexity of $O(2^{50})$ and key recovery attacks with complexity of $O(2^{53})$.

The previous analyses are conducted on the keystream generation process alone and do not analyse the initialisation process. By contrast, this analysis considers the initialisation process of the CSA-SC. In this chapter, we investigate two aspects of the initialisation process of the CSA-SC: state convergence and slid pairs.

This chapter is outlined as follows. Section 5.1 describes CSA-SC and its initialisation process, based on the 89-bit state representation given in [96]. Section 5.2 presents the analysis of the initialisation process of CSA-SC and also investigates state convergence during both the initialisation and keystream generation processes. Analysis of slid pairs is discussed in Section 5.3. Section 5.5 summarises this chapter and presents the security impact of this analysis.

5.1 Specification of CSA-SC

The CSA-SC [44,84,96,104] keystream generator uses word based registers (with a 4-bit word size) and bit based state update functions. The structure consists of two feedback shift registers (FSRs) denoted A and B, a combiner with memory formed from memory stages denoted E, F and c and 7 S-boxes. Registers A and B have ten stages each, and each stage stores a nibble (4-bit word). The combiner memories E and F each store a 4-bit word, and c stores only 1 bit. For analysis at the bit level, the 7 S-boxes and a modular addition operation which updates memories F and c are the only nonlinear functions used during the initialisation and keystream generation processes. Figure 5.1 illustrates the CSA-SC during the keystream generation and the initialisation process (dashed lines are used only during initialisation).

Let A_i^t and B_i^t denote the i^{th} stage of registers A and B respectively, for $i \in \{0, \ldots, 9\}$, at time t. Let $a_{i,j}^t$ and $b_{i,j}^t$ represent the content of the j^{th} bit of the i^{th} stage at time t for registers A and B, respectively, for $i \in \{0, \ldots, 9\}$ and $j \in \{0, \ldots, 3\}$. Let E^t, F^t and c^t represent the contents of the memory stages E F and c at time t respectively. Let e_i^t and f_i^t represent the i^{th} bit of the memories E and F at time t.

CSA-SC uses a 64-bit secret key $K = k_0, \ldots, k_{63}$, where k_i is the i^{th} key bit, and a 64-bit initial value IV $= v_0, \ldots, v_{63}$, where v_i is the i^{th} IV bit. During the initialisation process these key-IV values are spread across the entire internal state. Let I_A^t and I_B^t denote 4-bit words taken from the IV, and used at time t as input to registers A and B respectively (details are given in the following section). Let $i_{A,j}^t$ represent the j^{th} bit of I_A^t for $j \in \{0, 1, 2, 3\}$ and similarly $i_{B,j}^t$ represent the j^{th} bit of the I_B^t for $j \in \{0, 1, 2, 3\}$.

In the previous representations of the CSA-SC, seven nonlinear S-boxes are used during both the initialisation and keystream processes. Each S-box takes 5 bits as input from register A and produces 2 output bits. The total number of distinct input and output bits of the S-boxes are 35 and 14, respectively. In this analysis, we treat these seven S-boxes as 14 Boolean functions, so the analysis of 14 Boolean functions is easier than the previous representation of S-boxes, where each S-box takes 5 bits as input to generate 2 output bits. Each Boolean function takes 5 bits as input and produces 1 output bit. Let $S_j(i_0, i_1, i_2, i_3, i_4)$ represent the j^{th} Boolean function with respect to 5 inputs, which are i_0, i_1, i_2, i_3, i_4, For more detail, see Appendix B, where Table B.1 presents the algebraic form of the 14 Boolean functions and Table B.2 shows the truth table of these 14 Boolean functions.

Twelve of the Boolean function outputs are used directly to update the contents of A_0, B_0 and (conditionally) F and c, while the other two Boolean function outputs determine the manner in which B_0, F and c are updated. The output bits of the Boolean functions at time t can be grouped in terms of the components they are used to update. We use notation corresponding to the previous references [96, 104] as follows: four bits $x_0^t, x_1^t, x_2^t, x_3^t$ are labeled as X, four bits

$y_0^t, y_1^t, y_2^t, y_3^t$ are labeled as Y, four bits $z_0^t, z_1^t, z_2^t, z_3^t$ are labeled as Z, one bit is labeled as p^t and last bit is labeled as q^t; all of these output bits are represented at time t as shown in Table 5.1. Note that the least significant bit or stage of each nibble is indexed by 0, $\oplus$ denotes bitwise XORs, $\boxplus$ denotes addition modulo 2^4 and ROL_i represents a left rotation by i bits.

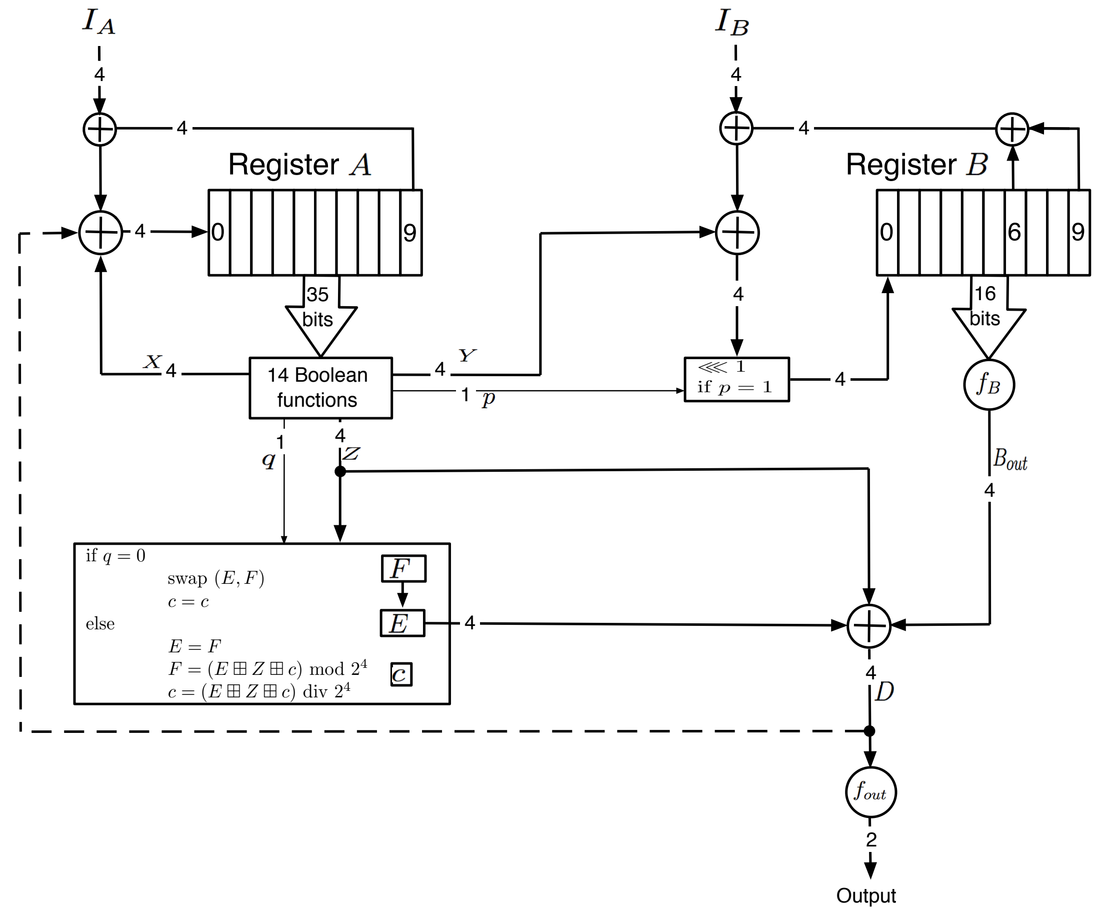


Figure 5.1: Common Scrambling Algorithm Stream Cipher (CSA-SC)

5.1.1 Initialisation process

The initialisation process uses a 64-bit secret key and a 64-bit IV and performs 32 iterations (starting at $t = -31$) [104] to produce the 89-bit initial state. Once the initial state is obtained, keystream generation can begin. The initialisation process occurs in two phases: *key loading phase* and *diffusion phase*. Note that loading of the IV is performed during the diffusion phase.

Table 5.1: Input and output bits of the 14 Boolean functions

		X				Y				Z					
Functions		S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}
Output bits		x_0^t	x_1^t	x_2^t	x_3^t	y_0^t	y_1^t	y_2^t	y_3^t	z_0^t	z_1^t	z_2^t	z_3^t	p^t	q^t
	i_0^t	$a_{3,3}$	$a_{1,3}$	$a_{2,1}$	$a_{4,0}$	$a_{3,1}$	$a_{5,2}$	$a_{3,3}$	$a_{1,3}$	$a_{2,1}$	$a_{4,0}$	$a_{3,1}$	$a_{5,2}$	$a_{2,2}$	$a_{2,2}$
Input	i_1^t	$a_{1,1}$	$a_{2,0}$	$a_{3,2}$	$a_{1,2}$	$a_{4,1}$	$a_{4,3}$	$a_{1,1}$	$a_{2,0}$	$a_{3,2}$	$a_{1,2}$	$a_{4,1}$	$a_{4,3}$	$a_{3,0}$	$a_{3,0}$
bits	i_2^t	$a_{2,3}$	$a_{5,1}$	$a_{6,3}$	$a_{6,1}$	$a_{5,0}$	$a_{6,0}$	$a_{2,3}$	$a_{5,1}$	$a_{6,3}$	$a_{6,1}$	$a_{5,0}$	$a_{6,0}$	$a_{7,1}$	$a_{7,1}$
	i_3^t	$a_{4,2}$	$a_{5,3}$	$a_{7,0}$	$a_{7,3}$	$a_{7,2}$	$a_{8,1}$	$a_{4,2}$	$a_{5,3}$	$a_{7,0}$	$a_{7,3}$	$a_{7,2}$	$a_{8,1}$	$a_{8,2}$	$a_{8,2}$
	${}^ti_4$	$a_{8,0}$	$a_{6,2}$	$a_{9,1}$	$a_{9,0}$	$a_{9,3}$	$a_{9,2}$	$a_{8,0}$	$a_{6,2}$	$a_{9,1}$	$a_{9,0}$	$a_{9,3}$	$a_{9,2}$	$a_{8,3}$	$a_{8,3}$

Key loading phase

In the loading phase, firstly, all of the registers A and B and memories E, F and c are set to zeros. Then the 64-bit secret key is transferred to specified positions in the shift registers as follows:

$$a_{i,j} = \begin{cases} k_{4\cdot i+j} & \text{for } i \in \{0,\dots,7\} \\ 0 & \text{for } i \in \{8,9\} \end{cases}$$

$$b_{i,j} = \begin{cases} k_{32+4\cdot i+j} & \text{for } i \in \{0,\dots,7\} \\ 0 & \text{for } i \in \{8,9\} \end{cases}$$

Diffusion phase

The diffusion phase consists of 32 iterations of the initialisation state-update function. During each iteration two different nibbles (4 bits) of the IVs I_A^t and I_B^t are loaded into stages A_0^t and B_0^t at time t. These I_A^t and I_B^t can be described as follows:

$$I_A^t = \begin{cases} v_x, v_{x+1}, v_{x+2}, v_{x+3} & \text{for } t \in \{-31, -29, \ldots\} \\ v_{x+4}, v_{x+5}, v_{x+6}, v_{x+7} & \text{for } t \in \{-30, -28, \ldots\} \end{cases}$$
$$I_B^t = \begin{cases} v_{x+4}, v_{x+5}, v_{x+6}, v_{x+7} & \text{for } t \in \{-31, -29, \ldots\} \\ v_x, v_{x+1}, v_{x+2}, v_{x+3} & \text{for } t \in \{-30, -28, \ldots\} \end{cases}$$
$$\text{for } x = 8 \cdot [7 + \lceil t/4 \rceil]$$

Every byte of the IV is used in four consecutive iterations of the state update function, as described above. Each byte consists of two nibbles: high and low, (H, L). For odd t, I_A^t and I_B^t take the H and L values respectively. For even t the nibbles are used in the other order, so I_A^t and I_B^t take the L and H values respectively.

The state update functions for registers A and B during the initialisation process (in Figure 5.1) are described as follows:

$$A_i^t = A_{i-1}^{t-1} \qquad \text{for } i \in \{1, \ldots, 9\} \tag{5.1a}$$
$$A_0^t = A_9^{t-1} \oplus X^{t-1} \oplus D^{t-1} \oplus I_A^t \tag{5.1b}$$

$$B_i^t = B_{i-1}^{t-1} \qquad \text{for } i \in \{1, \ldots, 9\} \tag{5.2a}$$
$$B_0^t = (B_6^{t-1} \oplus B_9^{t-1} \oplus Y^{t-1} \oplus I_B^t)_{ROL_{p^{t-1}}} \tag{5.2b}$$

The state update functions for memories E, F and c during the initialisation process and keystream generation are described as follows:

$$E^t = F^{t-1} \tag{5.3a}$$
$$F^t = \begin{cases} E^{t-1} & \text{if } q^{t-1} = 0 \\ E^{t-1} \boxplus Z^{t-1} \boxplus c^{t-1} \bmod 2^4 & \text{if } q^{t-1} = 1 \end{cases} \tag{5.3b}$$
$$c^t = \begin{cases} c^{t-1} & \text{if } q^{t-1} = 0 \\ (E^{t-1} \boxplus Z^{t-1} \boxplus c^{t-1}) \text{ div } 2^4 & \text{if } q^{t-1} = 1 \end{cases} \tag{5.3c}$$

During the initialisation, the output of the combiner memories is also XORed with the output of register B and 4 bits of the Boolean function outputs (Z) to give a 4-bit output at time t denoted D^t as shown in Figure 5.1.

$$D^t = E^t \oplus Z^t \oplus B^t_{out} \tag{5.4}$$

where

$$\begin{aligned}B^t_{out} =&(b^t_{0,out} \parallel b^t_{1,out} \parallel b^t_{2,out} \parallel b^t_{3,out})\\ =&(b^t_{8,2} \oplus b^t_{5,3} \oplus b^t_{2,1} \oplus b^t_{7,0} \parallel b^t_{4,3} \oplus b^t_{7,2} \oplus b^t_{3,0} \oplus b^t_{4,1} \parallel\\ &b^t_{5,0} \oplus b^t_{7,1} \oplus b^t_{2,3} \oplus b^t_{3,2} \parallel b^t_{2,0} \oplus b^t_{5,1} \oplus b^t_{6,2} \oplus b^t_{8,3})\end{aligned}$$

After the diffusion phase is completed, the keystream generator is said to be in its *initial state*. Following this, the keystream generation process begins.

5.1.2 Keystream generation

At $t = 0$, the CSA-SC has completed the initialisation processes and is ready for keystream generation. During keystream generation, there is no feedback from pre-output word D to the register A, and there is no input IV to the registers A and B. During this phase, the state update function for the registers A and B is as follows:

$$A^t_0 = A^{t-1}_9 \oplus X^{t-1} \tag{5.5a}$$

$$B^t_0 = (B^{t-1}_6 \oplus B^{t-1}_9 \oplus Y^{t-1})_{ROL_{p^{t-1}}} \tag{5.5b}$$

The keystream generator generates two output bits z^t at each clock time t from the 4 bits D, by combining pairs of bits as follows: $z^t = (d^t_2 \oplus d^t_3 || d^t_0 \oplus d^t_1)$.

5.2 State Convergence in CSA-SC

CSA-SC uses a 64-bit secret key and a 64-bit IV to form an 89-bit internal state. Thus, there are 2^{128} possible key-IV combinations but only 2^{89} possible internal states. When the 2^{128} possible key-IV combinations are mapped to at most 2^{89} possible internal states, there is clearly a compression of multiple key-IV pairs to the same internal state. As the state-update function is nonlinear during the diffusion phase, the degree of compression may not be uniform. However, on average, there are 2^{39} key-IV pairs corresponding to each internal state (even before we consider the effect of state convergence).

In addition to this compression, state convergence occurs during initialisation specifically during the IV loading-diffusion phase. Our analysis of the state convergence that occurs during the initialisation process of CSA-SC is based on assuming that the entire state contents are known at time t and considering the possible contents of A_9^{t-1}. A_9^{t-1} is used as input to the state update function to form the new values of A_0^t, B_0^t, and for the case when $q = 1$ the values of F^t and c^t, and then A_9^{t-1} is shifted out of the internal state. We show below that multiple distinct values for the contents of A_9^{t-1} may result in the same value for each of A_0^t, B_0^t, F^t and c^t. Figure 5.2 shows the interaction between the contents of stages A_0^t, B_0^t and A_9^{t-1}.

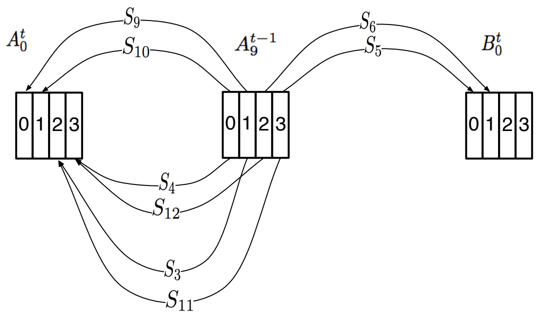


Figure 5.2: State convergence and stages' interaction during initialisation process

5.2.1 Initialisation State Update Equations

Equations 5.1a to 5.4 describe the state update function of the initialisation process in terms of 4-bit words. However, in this section, the state update equations of the initialisation process are described at the bit level. This representation helps to analyse and identify aspects of the initialisation process such as the existence of converging state and slid pairs.

The bit values for p^t and q^t obtained from S_{13} and S_{14} determine the state update processes for B_0^t, F^t and c^t via Equations 5.2b, 5.3b and 5.3c. The contents of register B and memories F and c can thus be written as follows:

Register B

It is possible to rewrite the first nibble of the register B as a function of the Boolean function $S_{13}(p^t)$. To simplify the notation, we write $B'_0 = (b'_0, b'_1, b'_2, b'_3)$ for the transient value of B_0 before the left rotation is applied; we have:

$$B_0'^t = B_6^{t-1} \oplus B_9^{t-1} \oplus Y^{t-1} \oplus I_B^t$$

so

$$b_0'^t = b_{6,0}^{t-1} \oplus b_{9,0}^{t-1} \oplus y_0^t \oplus i_{B,0}^t \tag{5.6a}$$
$$b_1'^t = b_{6,1}^{t-1} \oplus b_{9,1}^{t-1} \oplus y_1^t \oplus i_{B,1}^t \tag{5.6b}$$
$$b_2'^t = b_{6,2}^{t-1} \oplus b_{9,2}^{t-1} \oplus y_2^t \oplus i_{B,2}^t \tag{5.6c}$$
$$b_3'^t = b_{6,3}^{t-1} \oplus b_{9,3}^{t-1} \oplus y_3^t \oplus i_{B,3}^t \tag{5.6d}$$

and

$$B_0^t = (B_0'^t)_{ROL_{p^{t-1}}}$$

The individual new bits for register B are described as follows:

$$b_{0,0}^t = b_0'^t(1 \oplus p^{t-1}) \oplus b_1'^t(p^{t-1}) \tag{5.7a}$$
$$b_{0,1}^t = b_1'^t(1 \oplus p^{t-1}) \oplus b_2'^t(p^{t-1}) \tag{5.7b}$$
$$b_{0,2}^t = b_2'^t(1 \oplus p^{t-1}) \oplus b_3'^t(p^{t-1}) \tag{5.7c}$$
$$b_{0,3}^t = b_3'^t(1 \oplus p^{t-1}) \oplus b_0'^t(p^{t-1}) \tag{5.7d}$$

Memories F and c

From Equations 5.3b and 5.3c, we have:

$$F^t = E^{t-1}(1 \oplus q^{t-1}) \oplus (E^{t-1} \boxplus Z^{t-1} \boxplus c^{t-1} \bmod 2^4)(q^{t-1}) \tag{5.8a}$$
$$c^t = c^{t-1}(1 \oplus q^{t-1}) \oplus ((E^{t-1} \boxplus Z^{t-1} \boxplus c^{t-1}) \operatorname{div} 2^4)(q^{t-1}) \tag{5.8b}$$

To express the integer addition in Equations 5.8a and 5.8b as binary addition, we introduce new intermediate terms (h_0, h_1, h_2, h_3) as follows:

$$h_0^t = e_0^{t-1} z_0^{t-1} \oplus e_0^{t-1} c^{t-1} \oplus z_0^{t-1} c^{t-1} \tag{5.9a}$$

$$h_1^t = e_1^{t-1} z_1^{t-1} \oplus e_1^{t-1} h_0^t \oplus z_1^{t-1} h_0^t \tag{5.9b}$$

$$h_2^t = e_2^{t-1} z_2^{t-1} \oplus e_2^{t-1} h_1^t \oplus z_2^{t-1} h_1^t \tag{5.9c}$$

$$h_3^t = e_3^{t-1} z_3^{t-1} \oplus e_3^{t-1} h_2^t \oplus z_3^{t-1} h_2^t \tag{5.9d}$$

The system of equations for the memory stages F can then be written using Equations 5.9 as follows:

$$f_0^t = e_0^{t-1}(1 \oplus q^{t-1}) \oplus (e_0^{t-1} \oplus z_0^{t-1} \oplus c^{t-1})(q^{t-1}) \tag{5.10a}$$

$$f_1^t = e_1^{t-1}(1 \oplus q^{t-1}) \oplus (e_1^{t-1} \oplus z_1^{t-1} \oplus h_0^t)(q^{t-1}) \tag{5.10b}$$

$$f_2^t = e_2^{t-1}(1 \oplus q^{t-1}) \oplus (e_2^{t-1} \oplus z_2^{t-1} \oplus h_1^t)(q^{t-1}) \tag{5.10c}$$

$$f_3^t = e_3^{t-1}(1 \oplus q^{t-1}) \oplus (e_3^{t-1} \oplus z_3^{t-1} \oplus h_2^t)(q^{t-1}) \tag{5.10d}$$

and similarly, we have

$$c^t = c^{t-1}(1 \oplus q^{t-1}) \oplus h_3^t(q^{t-1}) \tag{5.10e}$$

The new 4-bit words A_0^t and B_0^t of both registers A and B, respectively, can also be expressed in terms of individual bits as follows:

For register A:

$$a_{0,0}^t = a_{9,0}^{t-1} \oplus S_1(a_{3,3}^{t-1} \ldots a_{8,0}^{t-1}) \oplus S_9(a_{2,1}^{t-1} \ldots a_{9,1}^{t-1}) \oplus e_0^{t-1} \oplus b_{0out}^{t-1} \oplus i_{A,0}^t \tag{5.11a}$$

$$a_{0,1}^t = a_{9,1}^{t-1} \oplus S_2(a_{1,3}^{t-1} \ldots a_{6,2}^{t-1}) \oplus S_{10}(a_{4,0}^{t-1} \ldots a_{9,0}^{t-1}) \oplus e_1^{t-1} \oplus b_{1out}^{t-1} \oplus i_{A,1}^t \tag{5.11b}$$

$$a_{0,2}^t = a_{9,2}^{t-1} \oplus S_3(a_{2,1}^{t-1} \ldots a_{9,1}^{t-1}) \oplus S_{11}(a_{3,1}^{t-1} \ldots a_{9,3}^{t-1}) \oplus e_2^{t-1} \oplus b_{2out}^{t-1} \oplus i_{A,2}^t \tag{5.11c}$$

$$a_{0,3}^t = a_{9,3}^{t-1} \oplus S_4(a_{4,0}^{t-1} \ldots a_{9,0}^{t-1}) \oplus S_{12}(a_{5,2}^{t-1} \ldots a_{9,2}^{t-1}) \oplus e_3^{t-1} \oplus b_{3out}^{t-1} \oplus i_{A,3}^t \tag{5.11d}$$

For register B:

$$b_0'^t = b_{9,0}^{t-1} \oplus b_{6,0}^{t-1} \oplus S_5(a_{3,1}^{t-1} \dots a_{9,3}^{t-1}) \oplus i_{B,0}^t \tag{5.12a}$$

$$b_1'^t = b_{9,1}^{t-1} \oplus b_{6,1}^{t-1} \oplus S_6(a_{5,2}^{t-1} \dots a_{9,2}^{t-1}) \oplus i_{B,1}^t \tag{5.12b}$$

$$b_2'^t = b_{9,2}^{t-1} \oplus b_{6,2}^{t-1} \oplus S_7(a_{3,3}^{t-1} \dots a_{8,0}^{t-1}) \oplus i_{B,2}^t \tag{5.12c}$$

$$b_3'^t = b_{9,3}^{t-1} \oplus b_{6,3}^{t-1} \oplus S_8(a_{1,3}^{t-1} \dots a_{6,2}^{t-1}) \oplus i_{B,3}^t \tag{5.12d}$$

5.2.2 State Convergence During Initialisation Process

We examine these individual update equations for each bit position in A_0^t and B_0^t to determine if it is possible for any state to have multiple pre-images. Our focus is on the update of A and B through the values of A_9^{t-1} affecting new values of A_0^t and B_0^t. From Equations 5.11a to 5.11d, we see that A_9^{t-1} affects A_0^t in two different ways: both directly and through the Boolean functions S_3, S_4, S_9, S_{10}, S_{11} and S_{12} (x_2^t, x_3^t, z_0^t, z_1^t, z_2^t and z_3^t). Similarly, the values of A_9^{t-1} affect the new values of B_0^t through the Boolean functions S_5, S_6, S_7 and S_8 (y_0^t, y_1^t, y_2^t and y_3^t) and S_{13} (p) as presented by Equations 5.12a to 5.12d. We note that the contents of A_0^{t-1} to A_8^{t-1} and B_0^{t-1} to B_8^{t-1} are present in the subsequent state as A_1^t to A_9^t and B_1^t to B_9^t and are therefore assumed to be known. Likewise, we can treat any term in the initialisation state update functions which depends only on these values as fixed: these terms are S_1, S_2, S_7, S_8, b_{0out}^{t-1} to b_{3out}^{t-1} and $b_{6,0}^{t-1}$ to $b_{6,3}^{t-1}$. We also assume that $i_{A,0}^t$ to $i_{A,3}^t$ and $i_{B,0}^t$ to $i_{B,3}^t$ are given and therefore fixed as well. By doing this, we can introduce constants $C_0, C_1, C_2, C_3, C_4, C_5, C_6, C_7$ to simplify Equations 5.11a to 5.11d and 5.12a to 5.12d as follows:

Defining

$$
\begin{aligned}
C_0 &= S_1(a_{3,3}^{t-1}\dots a_{8,0}^{t-1}) \oplus b_{0out}^{t-1} \oplus i_{A,0}^{t} \\
C_1 &= S_2(a_{1,3}^{t-1}\dots a_{6,2}^{t-1}) \oplus b_{1out}^{t-1} \oplus i_{A,1}^{t} \\
C_2 &= b_{2out}^{t-1} \oplus i_{A,2}^{t} \\
C_3 &= b_{3out}^{t-1} \oplus i_{A,3}^{t} \\
C_4 &= b_{6,0}^{t-1} \oplus i_{B,0}^{t} \\
C_5 &= b_{6,1}^{t-1} \oplus i_{B,1}^{t} \\
C_6 &= b_{6,2}^{t-1} \oplus S_7(a_{3,3}^{t-1}\dots a_{8,0}^{t-1}) \oplus i_{B,2}^{t} \\
C_7 &= b_{6,3}^{t-1} \oplus S_8(a_{1,3}^{t-1}\dots a_{6,2}^{t-1}) \oplus i_{B,3}^{t}
\end{aligned}
$$

we have

$$a_{0,0}^{t} = C_0 \oplus a_{9,0}^{t-1} \oplus S_9(a_{2,1}^{t-1}\dots a_{9,1}^{t-1}) \oplus e_0^{t-1} \tag{5.14a}$$
$$a_{0,1}^{t} = C_1 \oplus a_{9,1}^{t-1} \oplus S_{10}(a_{4,0}^{t-1}\dots a_{9,0}^{t-1}) \oplus e_1^{t-1} \tag{5.14b}$$
$$a_{0,2}^{t} = C_2 \oplus a_{9,2}^{t-1} \oplus S_3(a_{2,1}^{t-1}\dots a_{9,1}^{t-1}) \oplus S_{11}(a_{3,1}^{t-1}\dots a_{9,3}^{t-1}) \oplus e_2^{t-1} \tag{5.14c}$$
$$a_{0,3}^{t} = C_3 \oplus a_{9,3}^{t-1} \oplus S_4(a_{4,0}^{t-1}\dots a_{9,0}^{t-1}) \oplus S_{12}(a_{5,2}^{t-1}\dots a_{9,2}^{t-1}) \oplus e_3^{t-1} \tag{5.14d}$$

and

$$b_0'^{t} = (C_4 \oplus b_{9,0}^{t-1} \oplus S_5(a_{3,1}^{t-1}\dots a_{9,3}^{t-1})) \tag{5.15a}$$
$$b_1'^{t} = (C_5 \oplus b_{9,1}^{t-1} \oplus S_6(a_{5,2}^{t-1}\dots a_{9,2}^{t-1})) \tag{5.15b}$$
$$b_2'^{t} = (C_6 \oplus b_{9,2}^{t-1}) \tag{5.15c}$$
$$b_3'^{t} = (C_7 \oplus b_{9,3}^{t-1}) \tag{5.15d}$$

During the first two clocks of the initialisation process, the values of A_9^t are zero, according to the loading format. Therefore, there are no multiple pre-images for the internal state at $t = -31$ or $t = -30$. State convergence cannot occur before the 3rd clock step.

Equations 5.14a to 5.14d and 5.15a to 5.15d can be rearranged to evaluate the previous state at time $t-1$ in terms of the state at time t, as follows:

$$a_{9,0}^{t-1} = C_0 \oplus a_{0,0}^{t} \oplus S_9(a_{2,1}^{t-1} \dots a_{9,1}^{t-1}) \oplus e_0^{t-1} \tag{5.16a}$$
$$a_{9,1}^{t-1} = C_1 \oplus a_{0,1}^{t} \oplus S_{10}(a_{4,0}^{t-1} \dots a_{9,0}^{t-1}) \oplus e_1^{t-1} \tag{5.16b}$$
$$a_{9,2}^{t-1} = C_2 \oplus a_{0,2}^{t} \oplus S_3(a_{2,1}^{t-1} \dots a_{9,1}^{t-1}) \oplus S_{11}(a_{3,1}^{t-1} \dots a_{9,3}^{t-1}) \oplus e_2^{t-1} \tag{5.16c}$$
$$a_{9,3}^{t-1} = C_3 \oplus a_{0,3}^{t} \oplus S_4(a_{4,0}^{t-1} \dots a_{9,0}^{t-1}) \oplus S_{12}(a_{5,2}^{t-1} \dots a_{9,2}^{t-1}) \oplus e_3^{t-1} \tag{5.16d}$$

$$b_{9,0}^{t-1} = C_4 \oplus b_0'^{t} \oplus S_5(a_{3,1}^{t-1} \dots a_{9,3}^{t-1}) \tag{5.17a}$$
$$b_{9,1}^{t-1} = C_5 \oplus b_1'^{t} \oplus S_6(a_{5,2}^{t-1} \dots a_{9,2}^{t-1}) \tag{5.17b}$$
$$b_{9,2}^{t-1} = C_6 \oplus b_2'^{t} \tag{5.17c}$$
$$b_{9,3}^{t-1} = C_7 \oplus b_3'^{t} \tag{5.17d}$$

Note that the bits of A_9^{t-1} still appear implicitly on the right hand sides of these equations and also that the values of $b_{9,0}^{t-1}$ to $b_{9,3}^{t-1}$ are determined uniquely once $a_{9,2}^{t-1}$ and $a_{9,3}^{t-1}$ are known. (That is, every distinct solution for A_9^{t-1} yields a unique corresponding solution for B_9^{t-1}.)

Now, we examine the set of Equations 5.16a to 5.16d to identify state convergence in the CSA-SC. Let $\overline{x}$ denote the complement of x. In this section, for any of the Boolean functions $S_3, S_4, S_9, S_{10}, S_{11}$ and S_{12}, the function is said to have *even parity* with respect to i_4 if $S(i_0, i_1, i_2, i_3, \overline{i_4}) = S(i_0, i_1, i_2, i_3, i_4)$, and to have *odd parity* with respect to i_4 if $S(i_0, i_1, i_2, i_3, \overline{i_4}) = \overline{S(i_0, i_1, i_2, i_3, i_4)}$ (For the remainder of this section, we will refer to these parity conditions by saying that S is even or odd respectively). From Equations 5.16a to 5.16d, the values of $a_{0,0}^{t}$ to $a_{0,3}^{t}$ are assumed to be fixed to investigate whether solutions for $a_{9,0}^{t-1}$ to $a_{9,3}^{t-1}$ exist which satisfy Equation 5.16a to 5.16d. We consider the analysis separately for the two possible values of q^{t-1}. For a randomly chosen state, the probability that q is equal to 0 or 1 is $\frac{1}{2}$, since the Boolean function S_{14} is a balanced function.

The procedures used to obtain the following result are:

- Use MAGMA to generate the system of equations after 1 clock as shown in Appendix B.
- Use the real CSA-SC cipher to generate the truth table of the 14 Boolean

functions to determine the even parity and odd parity functions with respect to the fifth input, i_4.

- Based on the theoretical analysis (as shown later), use specific number of states that are applied to the CSA-SC cipher to generate the next states that are identical states (to demonstrate state convergence practically).

State Convergence when $q^{t-1} = 0$

When $q^{t-1} = 0$, $E^{t-1} = F^t$ is known, so the bits e_0^{t-1} to e_3^{t-1} can be absorbed into the constants C_0 to C_3. Consider this modified version of the Equations 5.16a and 5.16b for $a_{9,0}^{t-1}$ and $a_{9,1}^{t-1}$: If S_9 is even, then the value of $a_{9,0}^{t-1}$ is determined uniquely by Equation 5.16a, and Equation 5.16b then determines a unique value for $a_{9,1}^{t-1}$ (whether S_{10} is even or odd, since the inputs $(i_0, i_1, i_2, i_3, i_4)$ of S_{10} are then fully specified). Similarly, if S_{10} is even, the value of $a_{9,1}^{t-1}$ is determined uniquely and leads to a unique solution for $a_{9,0}^{t-1}$ as well. Thus $a_{9,0}^{t-1}$ and $a_{9,1}^{t-1}$ are both uniquely determined if either or both S_9 and S_{10} is even.

For an example of a system of equations that has unique solutions, suppose the following set of conditions holds:

1. $a_{0,0}^t = a_{0,1}^t = C_0 = C_1 = 0$.

2. The inputs $(i_0, i_1, i_2, i_3) = (0, 1, 1, 0)$ for both S_9 and S_{10}.

From Table B.2 in Appendix B, it is clear that S_9 is even since $S_9(0, 1, 1, 0, a_{9,1}^{t-1})$ $= 0$ whether $a_{9,1}^{t-1} = 0$ or 1; likewise, it can be seen that S_{10} is odd. Now Equation 5.16a and Condition 1 (above) imply that $a_{9,0}^{t-1} = S_9(0, 1, 1, 0, a_{9,1}^{t-1}) = 0$, and from Equation 5.16b, $a_{9,1}^{t-1} = S_{10}(0, 1, 1, 0, 0) = 1$. So, the value of $a_{9,0}^{t-1}$ is 0 which leads to a unique value of $a_{9,1}^{t-1}$ that is 1. Therefore, both $a_{9,0}^{t-1} = 0$ and $a_{9,1}^{t-1} = 1$ have unique values.

Conversely, if S_9 and S_{10} are both odd, then the pair of Equations 5.16a and 5.16b may be either inconsistent or consistent. If the equations are inconsistent, there is no solution for $a_{9,0}^{t-1}$ and $a_{9,1}^{t-1}$, but if they are consistent, then there are two valid solutions (a, b) and $(\overline{a}, \overline{b})$ for $(a_{9,0}^{t-1}, a_{9,1}^{t-1})$.

For example, if

1. $a_{0,0}^t = C_0 = C_1 = 0$ and $a_{0,1}^t = 1$, $S_9(\ldots, a_{9,1}^{t-1}) = a_{9,1}^{t-1}$.

2. $S_{10}(\ldots, a_{9,0}^{t-1}) = a_{9,0}^{t-1}$.

Then Equation 5.16a and 5.16b imply respectively that $a_{9,0}^{t-1} \oplus a_{9,1}^{t-1} = 0$ and $a_{9,0}^{t-1} \oplus a_{9,1}^{t-1} = 1$, so there is no solution for $a_{9,0}^{t-1}$ and $a_{9,1}^{t-1}$.

The solutions of $a_{9,0}^{t-1}$ and $a_{9,1}^{t-1}$ have been determined. Now, consider Equations 5.16c and 5.16d for $a_{9,2}^{t-1}$ and $a_{9,3}^{t-1}$. For any particular solution of Equations 5.16a and 5.16b, the given value of $a_{9,0}^{t-1}$ and $a_{9,1}^{t-1}$ will determine the values of $S_3(\ldots, a_{9,1}^{t-1})$ and $S_4(\ldots, a_{9,0}^{t-1})$. These values can be included in the constants C_2 and C_3 then the situation of the third and fourth equations are identical to that discussed above for $a_{9,0}^{t-1}$ and $a_{9,1}^{t-1}$. A similar argument and analysis can be used to draw the following conclusion for $a_{9,2}^{t-1}$ and $a_{9,3}^{t-1}$:

- If either S_{11} or S_{12} is even, then the values of $a_{9,2}^{t-1}$ and $a_{9,3}^{t-1}$ are determined uniquely from the related equations.
- If S_{11} and S_{12} are both odd, then there is either no solution for $a_{9,2}^{t-1}$ and $a_{9,3}^{t-1}$ or there are two complementary solution pairs for these variables.

Combining the above arguments, we see that there can be between zero and four solutions to the full equation set 5.16a to 5.16d. We now discuss these outcomes in more detail.

If there is no solution for $a_{9,0}^{t-1}$ and $a_{9,1}^{t-1}$, then it is not possible to obtain a valid solution set for the full set of variables $a_{9,0}^{t-1}$ to $a_{9,3}^{t-1}$. However, if there is a unique solution for $a_{9,0}^{t-1}$ and $a_{9,1}^{t-1}$, then there will be either no, one or two solutions for the full set of variables exactly when $a_{9,2}^{t-1}$ and $a_{9,3}^{t-1}$ have no, one or two solutions.

Now, we consider the case when there are two valid solution pairs for $a_{9,0}^{t-1}$ and $a_{9,1}^{t-1}$. If either S_{11} or S_{12} is even, then each of these will have a unique solution for $a_{9,2}^{t-1}$ and $a_{9,3}^{t-1}$. Then, there will be exactly two solutions for the full set of variables $a_{9,0}^{t-1}$ to $a_{9,3}^{t-1}$ (these sets will differ in $a_{9,0}^{t-1}$ and $a_{9,1}^{t-1}$ and may or may not differ in the other two variables $a_{9,2}^{t-1}$ and $a_{9,3}^{t-1}$). If S_{11} and S_{12} are both odd, then the equations for $a_{9,2}^{t-1}$ and $a_{9,3}^{t-1}$ may have zero or two solutions. Therefore, each of $a_{9,0}^{t-1}$ and $a_{9,1}^{t-1}$ solutions will have either no or two solutions for $a_{9,2}^{t-1}$ and $a_{9,3}^{t-1}$. By considering S_3 or S_4 further as follows, we find:

- If S_3 and S_4 are both even, then both solutions for $a_{9,0}^{t-1}$ and $a_{9,1}^{t-1}$ lead to the same pair of equations for $a_{9,2}^{t-1}$ and $a_{9,3}^{t-1}$ and therefore, one of the following points is valid:
  - There is no solution for $a_{9,2}^{t-1}$ and $a_{9,3}^{t-1}$ in either case.

  - There are two solutions for $a_{9,2}^{t-1}$ and $a_{9,3}^{t-1}$ in both cases and the same solution applies in both cases.

- If S_3 and S_4 are both odd, the two solutions for $a_{9,0}^{t-1}$ and $a_{9,1}^{t-1}$ lead to different pairs of equations for $a_{9,2}^{t-1}$ and $a_{9,3}^{t-1}$, but either both pairs are consistent or both pairs are inconsistent. Therefore, either of the following descriptions is valid:
  - There is no solution for $a_{9,2}^{t-1}$ and $a_{9,3}^{t-1}$ in either case.
  - There are two solutions for $a_{9,2}^{t-1}$ and $a_{9,3}^{t-1}$ in each case, but the pair of solutions for one case is different from the pair of solutions in the other case.

- If one of S_3 or S_4 is odd and the other is even, then the two solutions of $a_{9,0}^{t-1}$ and $a_{9,1}^{t-1}$ lead to pairs of equations with one common equation in each pair. In this case, one pair of equations will be consistent and the other must be inconsistent. So, one case will have no solution for $a_{9,2}^{t-1}$ and $a_{9,3}^{t-1}$ and the other will have two solutions for $a_{9,2}^{t-1}$ and $a_{9,3}^{t-1}$.

Putting these results together, then:

- If S_3 and S_4 are both even, there are either zero or four solution sets for $a_{9,0}^{t-1}$ to $a_{9,3}^{t-1}$, which differ in $a_{9,0}^{t-1}$ and $a_{9,1}^{t-1}$ but have the same two solutions for $a_{9,2}^{t-1}$ and $a_{9,3}^{t-1}$.

- If S_3 and S_4 are both odd, there are either zero or four solution sets for $a_{9,0}^{t-1}$ to $a_{9,3}^{t-1}$, which differ in $a_{9,0}^{t-1}$ and $a_{9,1}^{t-1}$ and also in $a_{9,2}^{t-1}$ and $a_{9,3}^{t-1}$.

- If one of S_3 and S_4 is even and the other is odd, there are exactly two solution sets for $a_{9,0}^{t-1}$ to $a_{9,3}^{t-1}$.

By examining the truth tables for $S_3, S_4, S_9, S_{10}, S_{11}$ and S_{12} (presented in Appendix B, Table B.2), we can determine the probabilities that each of the above Boolean function is even or odd, assuming a uniform distribution over all possible contents for A_0 to A_8 in register A. Also, assuming a uniform distribution of the content of A_9, the probability of consistent and inconsistent equations when both of the relevant functions are odd will be $\frac{1}{2}$. From Figure 5.3, the total probability for each branch can be calculated.

The number of solutions presented above for the variable set $a_{9,0}^{t-1}$ to $a_{9,3}^{t-1}$ also represents the number of pre-images for each case. Table 5.2 collates the

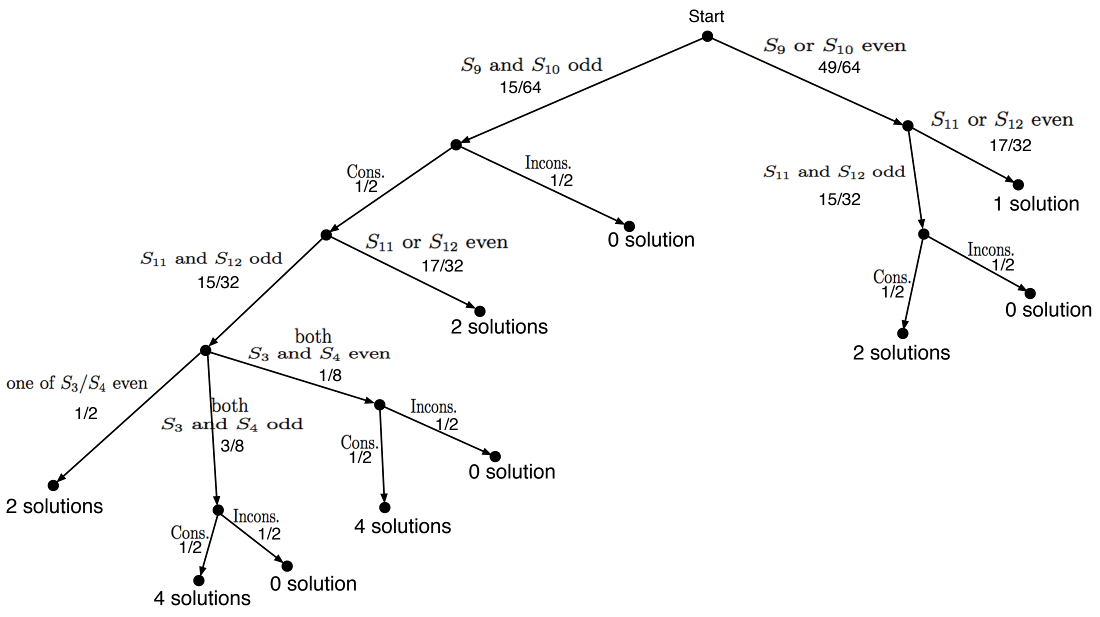


Figure 5.3: Branching of number of solutions (pre-images)

proportion and number of pre-images for each case. Tables 5.3 and 5.4 show examples of states with two and four pre-images respectively. Note that the only difference between the pre-image states is the bold bits $a_{9,0}$ to $a_{9,3}$ and $b_{9,0}$ to $b_{9,3}$.

Table 5.2: Proportions and number of pre-images for cases when $q^{t-1} = 0$

Case	(i)	(ii)	(iii)	(iv)
Number of pre-images	0	1	2	4
Proportion	$\frac{5085}{16384}$	$\frac{833}{2048}$	$\frac{2205}{8192}$	$\frac{225}{16384}$
Proportion (decimal)	0.31	0.407	0.269	0.014

As mentioned above, state convergence can occur from the third clock in the diffusion phase of initialisation. Now, if we assume that the state convergence occurs after the 64 key bits and 8 IV bits are loaded, and we have a 72 bits of information in the internal state. Then the number of distinct internal states after the first clock with state convergence is $(1 - \frac{5085}{16384}) \cdot 2^{72} = 2^{71.46}$ states (when $q = 0$), where the proportion of 0 pre-image in Table 5.2 represents the proportion of inaccessible patterns at this clock, which is $\frac{5085}{16384} \approx 0.31$.

As the secret key is loaded to the registers A and B, we assume the content of register A is approximately independent identically distributed random bits during the clocks from the 3rd to 10th. From this assumption, the probabilities of state convergence apply approximately for the 8 clocks during diffusion phase (considering the output bit of the Boolean functions), that start from the 3rd clock. Therefore, we argue that the inaccessible patterns for each clocking step is quite similar across these 8 clocks. We can estimate the upper bound of the accessible states (for only $q^{t-1} = 0$) is $(2^{-0.54})^8 \approx 2^{-4.29}$, based on this assumption, the accessible states decreases approximately by 4 bits after the 10th clock of the initialisation process. After that, the contents of registers A and B are generated from the feedback and the output of Boolean functions, so we cannot assume the contents of registers A and B are independent identically distributed random form.

Table 5.3: Two pre-images for a given state when $q^{t-1} = 0$

	A	
State	1110010101010101000100001101010101010001	
1st pre-image	01010101010100010000110101010101000101**00**	. . .
2nd pre-image	01010101010100010000110101010101000101**11**	

	B	E	F	c
	0010111001011110010101101110110101010011	0010	1100	1
. . .	1110010111100101011011101101010100110111	1100	0010	1
	1110010111100101011011101101010100110111	1100	0010	1

State Convergence when $q^{t-1} = 1$

In the case of $q = 1$, from Equations 5.9a to 5.9d and 5.10a to 5.10e, the contents of the memories F^t and c^t are computed using addition modulo 2^4 of the values of E^{t-1}, Z^{t-1} and c^{t-1}. The investigation of the case $q = 1$ requires to solve the relevant equations, Equations 5.10a to 5.10e and 5.11a to 5.11d in term of the set of variables $a_{9,0}^{t-1}$ to $a_{9,3}^{t-1}$, e_0^{t-1} to e_3^{t-1}, c^{t-1} and the intermediate variables h_0^t to h_3^t. They will result in the number of valid solutions, which represent the number of pre-images.

Equations 5.10a to 5.10e are quintic equations. We expect multiple solutions

Table 5.4: Four pre-images for a given state when $q^{t-1} = 0$

	A	
State	0100010101010001000101010101010001110101	
1st pre-image	010101010001000101010101010001110101**0000**	
2nd pre-image	010101010001000101010101010001110101**0011**	...
3rd pre-image	010101010001000101010101010001110101**1110**	
4th pre-image	010101010001000101010101010001110101**1101**	

	B	*E*	*F*	*c*
	1001111001011110010101101110110101010011	0010	0001	1
	111001011110010101101110110101010011**0100**	1100	0010	1
...	111001011110010101101110110101010011**1000**	1100	0010	1
	111001011110010101101110110101010011**0000**	1100	0010	1
	111001011110010101101110110101010011**1100**	1100	0010	1

for these set of variables $a_{9,0}^{t-1}$ to $a_{9,3}^{t-1}$. Table 5.5 shows an example of two pre-images of one state when $q^{t-1} = 1$. The complexity of solving this set of equations is the degree of these equations (fifth) and considering the behaviour of the Boolean functions S_1 to S_4 and S_9 to S_{12} during the update function. The analysis of $q^{t-1} = 1$ will be left for future work.

Table 5.5: Two pre-images for a given state when $q^{t-1} = 1$

	A	
State	0010010101010111000100001101110101010001	
1st pre-image	01010101011100010000110111010101000101**11**	...
2nd pre-image	01010101011100010000110111010101000101**00**	

	B	*E*	*F*	*c*
	1001111001011110010101101110110101110011	0010	0011	1
...	1110010111100101011011101101011100111**0**00	0101	0010	0
	1110010111100101011011101101011100111**1**00	0101	0010	1

5.2.3 State Convergence During Keystream Generation

This section describes the investigation to determine whether state convergence can occur during keystream generation. Figure 5.4 illustrates the interaction between A_9^{t-1} with both of A_0^t and B_0^t at the same time. The state update function during the keystream generation of CSA-SC can be described as follows:

$$
\begin{aligned}
A_0^t =& A_9^{t-1} \oplus X^{t-1} \\
B_0^t =& (B_6^{t-1} \oplus B_9^{t-1} \oplus Y^{t-1})_{ROL_{p^{t-1}}}
\end{aligned}
$$

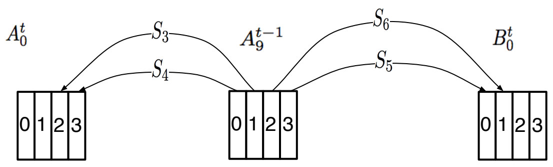


Figure 5.4: State convergence and stages' interaction during keystream generation

The new values of stages A_0^t and B_0^t can be considered in terms of their individual bits as follows:

$$a_{0,0}^t = a_{9,0}^{t-1} \oplus S_1(a_{3,3}^{t-1} \dots a_{8,0}^{t-1}) \tag{5.18a}$$
$$a_{0,1}^t = a_{9,1}^{t-1} \oplus S_2(a_{1,3}^{t-1} \dots a_{6,2}^{t-1}) \tag{5.18b}$$
$$a_{0,2}^t = a_{9,2}^{t-1} \oplus S_3(a_{2,1}^{t-1} \dots a_{9,1}^{t-1}) \tag{5.18c}$$
$$a_{0,3}^t = a_{9,3}^{t-1} \oplus S_4(a_{4,0}^{t-1} \dots a_{9,0}^{t-1}) \tag{5.18d}$$

$$b_0'^t = (b_{9,0}^{t-1} \oplus b_{6,0}^{t-1} \oplus S_5(a_{3,1}^{t-1} \dots a_{9,3}^{t-1})) \tag{5.18e}$$
$$b_1'^t = (b_{9,1}^{t-1} \oplus b_{6,1}^{t-1} \oplus S_6(a_{5,2}^{t-1} \dots a_{9,2}^{t-1})) \tag{5.18f}$$
$$b_2'^t = (b_{9,2}^{t-1} \oplus b_{6,2}^{t-1} \oplus S_7(a_{3,3}^{t-1} \dots a_{8,0}^{t-1})) \tag{5.18g}$$
$$b_3'^t = (b_{9,3}^{t-1} \oplus b_{6,3}^{t-1} \oplus S_8(a_{1,3}^{t-1} \dots a_{6,2}^{t-1})) \tag{5.18h}$$

Equations 5.18a to 5.18h can be treated in a similar way to Equations 5.11a to 5.11d and 5.12a to 5.12d. By considering anything which depends only on the values of stages A_0^{t-1} to A_8^{t-1} and B_0^{t-1} to B_8^{t-1} to be fixed, we can introduce constants C'_0, C'_1, C'_2, C'_3, C'_4 and C'_5 to simplify these equations as follows:

$$\begin{aligned}
C'_0 &= S_1(a_{3,3}^{t-1} \dots a_{8,0}^{t-1}) \\
C'_1 &= S_2(a_{1,3}^{t-1} \dots a_{6,2}^{t-1}) \\
C'_2 &= b_{6,0}^{t-1} \\
C'_3 &= b_{6,1}^{t-1} \\
C'_4 &= b_{6,2}^{t-1} \oplus S_7(a_{3,3}^{t-1} \dots a_{8,0}^{t-1}) \\
C'_5 &= b_{6,3}^{t-1} \oplus S_8(a_{1,3}^{t-1} \dots a_{6,2}^{t-1})
\end{aligned}$$

$$a_{0,0}^t = C'_0 \oplus a_{9,0}^{t-1} \tag{5.20a}$$

$$a_{0,1}^t = C'_1 \oplus a_{9,1}^{t-1} \tag{5.20b}$$

$$a_{0,2}^t = a_{9,2}^{t-1} \oplus S_3(a_{2,1}^{t-1} \dots a_{9,1}^{t-1}) \tag{5.20c}$$

$$a_{0,3}^t = a_{9,3}^{t-1} \oplus S_4(a_{4,0}^{t-1} \dots a_{9,0}^{t-1}) \tag{5.20d}$$

$$b'^t_0 = C'_2 \oplus b_{9,0}^{t-1} \oplus S_5(a_{3,1}^{t-1} \dots a_{9,3}^{t-1}) \tag{5.20e}$$

$$b'^t_1 = C'_3 \oplus b_{9,1}^{t-1} \oplus S_6(a_{5,2}^{t-1} \dots a_{9,2}^{t-1}) \tag{5.20f}$$

$$b'^t_2 = C'_4 \oplus b_{9,2}^{t-1} \tag{5.20g}$$

$$b'^t_3 = C'_5 \oplus b_{9,3}^{t-1} \tag{5.20h}$$

Equations 5.20a and 5.20b have unique solutions for the variables $a_{9,0}^{t-1}$ and $a_{9,1}^{t-1}$. Using Equations 5.20c and 5.20d, these solutions lead to unique solutions for $a_{9,2}^{t-1}$ and $a_{9,3}^{t-1}$, and hence from Equations 5.20e to 5.20h there are unique solutions for $b_{9,0}^{t-1}$ to $b_{9,3}^{t-1}$ also. As a result, there is only one pre-image for each state of register A. Hence, there is no state convergence in CSA-SC during the keystream generation process. Thus, the state convergence occurs only during the initialisation process of CSA-SC.

5.3 Analysis of Slid Pairs

In this section we investigate the possibility of finding multiple pairs (K, IV) and (K', IV') which produce a phase shifted version of the same keystream. For the CSA-SC, the loaded state corresponds to loaded keys only since the IV is loaded during the diffusion phase. Because of this, the second state of a slid pair depends only on the associated key, K', and we only need to consider the corresponding IV$'$, when determining whether the slid pair gives shifted keystream. Therefore, we can classify the steps that are required to obtain slid pairs and shifted keystream as follows:

i) The constraints that are required to obtain slid pairs.

ii) The constraints that are required to obtain slid pairs.

iii) The constraints that are required to obtain shifted keystream from slid pairs.

Note that Condition (*ii*) applies because the diffusion phase in CSA-SC is non-autonomous (there is an external input to the internal state); this condition does not apply to most other stream ciphers.

The purpose of this analysis is to identify the constraints that are required to obtain slid pairs and shifted keystream for the CSA-SC in the same cycle after α iterations. Comparing the properties listed in Section 2.7.3 with the operation of the CSA-SC, we note that properties (a) and (b) apply, but the state update functions during the initialisation and keystream generation processes have a degree of difference from one another.

Now, for a given key-IV pair, the first task is to identify the next key loaded state that may occur during the diffusion phase and to be in the same cycle (to satisfy the key loaded state of CSA-SC). Moreover, it is important to determine the relationship between the first key-IV pair, (K, IV), and the second key, K', which is obtained from the first key-IV after α iterations.

As presented in the above argument, there are some general conditions that are required to obtain slid pairs to meet the key loaded state format of the CSA-SC after α iterations. Slid pairs can occur after α iterations if the contents of stages A_8^α, A_9^α, B_8^α and B_9^α become zeros, and the memories E^α, F^α and c^α are still zeros to follow the CSA-SC's key loading format (Note from Equations 5.3a to 5.3c that E^t, F^t and c^t cannot become zero simultaneously once any of them has a non-zero content). The contents of stages A_8^α, A_9^α, B_8^α and B_9^α can be zeros as follows:

1) If $\alpha = 1$, it requires 8 key bits to be zeros. These key bits are $k_{28} - k_{31}$ and $k_{60} - k_{63}$.

2) For $2 \leq \alpha \leq 8$, two key bytes, $k_{32-4\alpha} - k_{39-4\alpha}$ and $k_{64-4\alpha} - k_{71-4\alpha}$, are zeros and separated by 32 bits.

3) For $\alpha = 9$, the key bits $k_0 - k_3$ and $k_{32} - k_{35}$ should be zeros, and Equations 5.21a and 5.21b (below) should be satisfied at $t = -31$.

4) For $\alpha \geq 10$, Equations 5.21a and 5.21b (below) should be satisfied for two successive iterations to generate two consecutive stages of zeros for registers A and B at the same time.

$$A_9^{t-1} \oplus X^{t-1} \oplus D^{t-1} \oplus I_A^t = 0 \tag{5.21a}$$

$$(B_6^{t-1} \oplus B_9^{t-1} \oplus Y^{t-1} \oplus I_B^t)_{ROL_{p^{t-1}}} = 0 \tag{5.21b}$$

In addition, the output bits of either the Boolean functions S_9 to S_{12} (Z) or the function S_{14} (q) should be zeros for $0 \leq t \leq \alpha$ to ensure that the contents of the memories E^α, F^α and c^α are zeros. In all the above cases, note that the second key, k', is uniquely determined by the first key, k, and the first $\lceil \frac{\alpha}{4} \rceil$ bytes of the first IV. The second IV$'$, is uniquely determined based on the first IV, and the number of shifted steps.

Because the registers are not autonomous during the diffusion phase, we must also consider the conditions that IV and IV$'$ must satisfy in order for the states resulting from (K, IV) and (K', IV$'$) to remain in step for the rest of the diffusion phase. The required specific conditions depend on the value of α and are discussed below in further detail.

To obtain shifted keystream from a slid pair that has remained in step, the following equations (Equations 5.22a and 5.22b) should be satisfied for the last α iterations of the second key-IV pair, (K', IV$'$).

$$D^{t-1} \oplus I_A^t = 0 \tag{5.22a}$$
$$I_B^t = 0 \tag{5.22b}$$

If all of the above equations are satisfied, the second keystream will be shifted from the first keystream by 2α bits.

Recall the IV loading format of the CSA-SC. The IV is loaded during the diffusion phase and each IV byte is loaded into registers A and B for four successive iterations. If α is not a multiple of 4, this operation imposes significant restrictions on the form of the IVs for the states of a slid pair to remain in step with one another. For this reason, we will describe slid pairs for $\alpha = 1, 2$ and 3 iterations briefly. Then, we will examine and focus on slid pairs for $\alpha = 4$ iterations (which have reasonable result for attacks).

The following process was used to obtain the result presented below:

- Use the above theoretical analysis to determine which key and IV pairs that result in slid pairs
- Use the real CSA-SC cipher to generate two keystream sequences using the two key-IV pairs to demonstrate practically the shifted keystream .

For $\alpha = 1$ iteration

Recalling the three steps that are mentioned in the beginning of this section, slid pairs and shifted keystream may occur with the following conditions:

i) 8 key bits (k_{28} to k_{31} and k_{60} to k_{63}) must be zeros (with probability 2^{-8}). Either the value of q^{-32} or the output of Boolean functions S_9 to S_{12} (z_0^{-32} to z_3^{-32}) are zeros (with probability $\frac{17}{32} \approx 2^{-0.913}$).

ii) The bytes of the first IV, are identical and the low nibbles of IV$'$ equal to the high nibbles of IV, $v'_L = v_H$ and vice versa $v_L = v'_H$. This condition restricts IV to 2^8 values of the potential 2^{64} and determines v' uniquely in terms of v.

iii) Shifted keystream may occur if the v'_H is zeros (with probability 2^{-4}). Moreover, Equation 5.22a must be satisfied for the last iteration of the diffusion phase (with probability 2^{-4}).

In summary, the probability of a randomly chosen key leading to a slid pair is $2^{-8.913}$ and the probability of a valid IV which will allow the slid pair to propagate and lead to shifted keystream, is $\frac{2^8}{2^{64}} \cdot 2^{-4} = 2^{-60}$. There is also a further probability of 2^{-4} that (K', IV') will satisfy Equation 5.22a at time $t = 0$. The second key K' is determined uniquely from K bits and some IV bits using Equations 5.11a to 5.11d to find $k'_0 - k'_3$, Equations 5.12a to 5.12d to find $k'_{32} - k'_{35}$ and the remaining key bits of K' are shifted by 4, $K' = K \gg 4$. The IV$'$ is determined directly from IV. Table 5.6 demonstrates two key-IV pairs which generate two keystreams shifted by 2 bits. These two keystreams are identical except the first two bits from the first keystream, that is keystream$_2$ = keystream$_1 \ll 2$.

Table 5.6: For $\alpha = 1$, two key-IV pairs generate two keystreams shifted by 2 bits

key$_1$	FFD7FFF0FFFFFFF0
IV$_1$	7070707070707070
keystream$_1$	{10} A914DA57A7CF11B65B00EEFDA9F7ABB8635E2DD117 04348B7FBFF30076BA92AC1893E9DDB2332F0949FCB24 648C61E34C7581A5624D2507AD6BFACEB6168C32
key$_2$	6FFD7FFFCFFFFFFF
IV$_2$	0707070707070707
keystream$_2$	A914DA57A7CF11B65B00EEFDA9F7ABB8635E2DD117 04348B7FBFF30076BA92AC1893E9DDB2332F0949FCB24 648C61E34C7581A5624D2507AD6BFACEB6168C32

For $\alpha = 2$ iterations

Recalling the three steps that are presented in the beginning of this section, slid pairs and shifted keystream may occur with the following conditions:

i) 16 key bits from K must be zeros (k_{24} to k_{31} and k_{56} to k_{63}) with probability 2^{-16}. The memories E, F and c should be zeros for $t = -31$ and $t = -30$ (with probability $(\frac{17}{32})^2 \approx 2^{-1.825}$).

ii) The bytes of first IV should be identical and IV = IV$'$. This condition again restricts IV to 2^8 values of the potential 2^{64}.

iii) To obtain shifted keystream, the last byte of second IV′, should be zero (with probability 2^{-8}), so the entire bytes of IVs, IV and IV′ should be zeros. Equation 5.22a must be satisfied for (K', IV') at time $t = -1$ and $t = 0$, which happens with probability 2^{-8}.

The probability of a randomly chosen key leading to a slid pair is $2^{-17.825}$. The probability of a valid IV which will allow slid pairs to propagate and lead to shifted keystream becomes $2^{-56} \cdot 2^{-8} = 2^{-64}$. There is a further probability of 2^{-8} for Equation 5.22a to be satisfied for two clocking steps. The second key-IV pair, (K', IV'), can be determined uniquely from the first key-IV pair, (K, IV), by extension of the method shown for $\alpha = 1$.

For $\alpha = 3$ iterations

Similarly to $\alpha = 1$ or 2, and recalling the three steps for occurrence of slid pairs and shifted keystream, we require

i) 16 key bits must be zeros (k_{20} to k_{27} and k_{52} to k_{59}) with probability 2^{-16}. The memories E, F and c should be zeros for the three clock steps (with probability $(\frac{17}{32})^3 \approx 2^{-2.738}$). Thus the probability of a randomly chosen key leading to a slid pair is $2^{-18.738}$.

ii) The valid IV that is required to keep slid pairs in step for this case is similar to the case of $\alpha = 1$, and occurs with probability of 2^{-56}.

iii) To obtain shifted keystream from slid pairs, the last byte of the second IV′ should be zero (with probability of 2^{-8}) and Equation 5.22a should be satisfied for the last 3 clocks (with probability 2^{-12}).

Thus, the total probability of a valid IV which will allow slid pairs to propagate and lead to shifted keystream becomes 2^{-64} with an additional probability of 2^{-12} that Equation 5.22a is satisfied for time $t = -2$, $t = -1$ and $t = -0$. The second key-IV pair, (K', IV'), can again be determined uniquely and directly from (K, IV).

These cases when $\alpha = 1$, 2 and 3, seem to be impractical due to the very small range of IV values for the which the slid pairs will lead to shifted keystream. Likewise other cases when $\alpha \not\equiv 0 \pmod 4$ e.g. $\alpha = 5$, 6 and 7 have the same probabilities and process of obtaining valid IV as for $\alpha = 2$ and 3. Moreover, the probabilities of possible valid keys, which is required to obtain slid pairs, and the conditions of shifted keystreams are successively lower, as are the probabilities of satisfying Equation 5.22a to guarantee shifted keystream.

For $\alpha = 4$ iterations

In this case, the three steps that are required for slid pairs and shifted keystream are as follows:

i) Two bytes of the first key, K, must be zeros, namely k_{16} to k_{23} and k_{48} to k_{55} (with probability of 2^{-16}). Either the values of q or the output of Boolean functions S_9 to S_{12} are zeros for the first four clocks to ensure that the memories E, F and c are in the initial condition (with probability $(\frac{17}{32})^4 = 2^{-3.65}$).

ii) This case works with any IV (the first IV) and the second IV$'$ is a 1-byte shifted version of the first IV.

iii) Shifted keystream from slid pairs may occur if the last byte of second IV$'$ is zero which happens with probability of 2^{-8}. Equation 5.22a must be satisfied for the last four iterations of the diffusion phase (the 29th, 30th, 31st and 32nd clocks) and has probability of 2^{-16}.

If a specific second IV$'$ is found (2^{56} out of 2^{64} IVs) then a slid pair may occur with probability of $2^{-19.65}$ and this slid pair may lead to shifted keystream with probability of 2^{-16}. Therefore, the total probability to obtain shifted keystream from all the key-IV set is $2^{-43.65}$. In this case, the keystreams are out of phase by 8 bits.

We do not provide an example for this case of $\alpha = 4$, since the previous example of $\alpha = 1$ is enough to demonstrate the work of the slid pairs and shifted keystreams. For this case of $\alpha = 4$, if the key-IV pairs (K, IV) and (K', IV$'$) are predetermined then slid pairs and shifted keystream may occur with probabilities $2^{-3.65}$ and $2^{-19.65}$ respectively.

The second key-IV pair, (K', IV$'$), can be determined directly from the first key-IV pair, (K', IV$'$), as mentioned above in the case of $\alpha = 1$. The second IV$'$ is a shifted version of the first IV by one byte where (IV$'$ =IV$\ll$ 1(byte)), and its last byte should be zero. The second key K' can be determined directly from Equations 5.11a to 5.11d and 5.12a to 5.12d for four clocks with the remainder of K' again being a shifted copy of K. For each clock, 8 bits from K' can be expressed in term of key-IV pair, (K, IV).

The principle of slid pairs for $\alpha = 4$ can be applied for multiples of 4 for $\alpha = 4, 8, \ldots 28$. The probability of slid pairs and shifted keystreams of these cases of multiples of 4 can be calculated as follows:

- Obtaining two bytes of zeros in the first key, K, for probability of 2^{-16}.
- The memories E, F and c to be in the initial condition (zeros), the probability is $(\frac{17}{32})^{\alpha}$.
- The last $(\alpha/4)$ bytes of the second IV$'$ should be zeros (with probability $2^{-2\alpha}$).
- Equation 5.22a should be satisfied for α clocks (it has probability of $2^{-4\alpha}$).

Thus, the combined probability of obtaining shifted keystream when α is a multiple of 4 is $2^{-16-0.913\alpha-2\alpha-4\alpha} = 2^{-16-6.913\alpha}$. This analysis shows that the best probability that can be found to obtain slid pairs and shifted keystream is for $\alpha = 4$.

The probabilities of obtaining slid pairs and shifted keystream are shown Table 5.7, for the previous cases when $\alpha = 1, 2, 3, 4$ and also for $\alpha = 8$. This table sets out the probabilities for the three steps discussed above:

i) The probability that a randomly chosen key, K, will lead to a slid pair.

ii) The proportion of the first IV for which the states of the slid pair will remain in step during the rest of the diffusion phase.

iii) The combined probability of Equation 5.22a and the condition on IV$'$ being satisfied, so that shifted keystream is obtained.

Table 5.7: Probabilities of obtaining slid pairs and shifted keystreams for α clocks

α	1	2	3	**4**	8
(i) To obtain slid pairs	$2^{-8.913}$	$2^{-17.825}$	$2^{-18.738}$	$\mathbf{2^{-19.65}}$	$2^{-23.3}$
(ii) To keep slid pairs in step	2^{-56}	2^{-56}	2^{-56}	2^{0}	2^{0}
(iii) To obtain shifted keystream	2^{-8}	2^{-16}	2^{-20}	$\mathbf{2^{-24}}$	2^{-48}
Total probabilities	$2^{-72.913}$	$2^{-89.825}$	$2^{-94.738}$	$\mathbf{2^{-43.65}}$	$2^{-71.3}$

5.4 Weak Key-IV Combinations

The CSA-SC transfers the secret key directly into the internal state and applies specific padding. The IV is loaded during the diffusion phase using the state update function. So, the IV loading operation makes the state update function during the diffusion phase a non-autonomous operation. Therefore, the weak key-IV combination seems possible to happen at the end of the initialisation process. The registers A and B can reset to contain all-zero values at the end of the initialisation process. But the memories c, E and F cannot be reset to contain all-zero values if any of them gain value other than zero during the diffusion phase. Therefore, the weak key-IV problem (as defined in Section 2.7.5) may occur in CSA-SC but it is hard to keep the memory c, E and F contain all-zero values during all the iterations of the diffusion phase. This problem is left for future work.

5.5 Summary and Security Impact

The initialisation process of the CSA-SC stream cipher was examined in this chapter. We based our analysis on the description of CSA-SC from [96], which uses an internal state of 89 bits. The initialisation process of CSA-SC loads the secret key into the internal state using a linear function to form a specific key loaded state (where some stages are filled by zeros). Following this, the IV is loaded during the diffusion phase. There are 32 iterations (clocks) during the diffusion phase and each IV byte is used for four consecutive iterations. The diffusion phase and keystream generation both use nonlinear state update functions with a degree of difference as shown in Figure 5.1.

This internal state is shorter than the combined key-IV length (64+64) of 128 bits. This leads to an effective reduction in the key-IV space during initialisation process, so that on average 2^{39} key-IV pairs correspond to each internal state.

In this chapter, investigations into two flaws in the CSA-SC initialisation process were described: state convergence and slid pairs. These problems have been discussed in detail in this chapter and are summarized below.

During the initialisation process (specifically during the diffusion phase), the state update function of CSA-SC is not one-to-one due to a combination of the linear feedback and the output of the 14 Boolean functions. This results in state convergence during the *diffusion* phase only of the CSA-SC initialisation process. Precisely, the content of stage A_9^{t-1}, which is shifted out of the internal state at time t, affects the contents of stages A_0^t in two different ways: through the linear feedback function and through the output of seven Boolean functions namely S_3, S_4, S_9, S_{10}, S_{11}, S_{12} and S_{14}. State convergence cannot occur in the first two iterations of the diffusion phase due to the specified format of the key loaded state. State convergence may start to occur from the third iteration of diffusion phase once A_9^t is not constrained to be zero.

We identify two distinct cases, $q^{t-1} = 0$ and $q^{t-1} = 1$, each occurring with probability of $\frac{1}{2}$. In the case of $q^{t-1} = 0$, as a result of the state convergence, after the third iteration of the diffusion phase, the state space is reduced by at least $\frac{5085}{16384} \approx 0.31$. Therefore, the overall proportion of states lost in this way is half of this value, that is approximately 0.155. In the case when $q^{t-1} = 1$, we have demonstrated by an example in Table 5.5 that state convergence occurs, but the exact degree of convergence is yet to be determined; this case is left for future work. We argue that the proportion of inaccessible states in the case $q^{t-1} = 0$ is approximately 0.31 for each clocking step from 3rd to 10th clocks, as the assumption of independent identically distributed random bits in register A is valid for these 8 clocks. After that, the assumption may not be valid. Based on our analysis when $q^{t-1} = 0$, there is a significant reduction in the state space that could be exploited by attackers for example using TMTO attacks.

The similarity of the iterations of the state update function during initialisation process (diffusion phase) and the format of key loaded state results in slid pairs in the internal state. For randomly chosen keys, slid pairs occur with probability that depends on the value of α. Slid pairs may lead to shifted version of the same keystream provided certain conditions on IV and IV$'$ are satisfied, along with additional conditions relating IV$'$ to the output D. However the state update functions of the initialisation and keystream generation processes have a degree of difference. Therefore, slid pairs may lead to shifted keystreams with a specific probability. Thus, slid pairs may occur during the *diffusion* phase of the initialisation process of CSA-SC and may lead to shifted version of the same keystream. From this investigation, the best result to find slid pairs is at $\alpha = 4$, where the key-IV pair (K, IV) generates a key loaded state after 4 iterations. This case has a probability of $2^{-19.65}$ of obtaining slid pairs. These slid pairs may lead to shifted key stream with a probability of 2^{-24}, as presented in Table 5.7

for $\alpha = 4$. Slid pairs and shifted keystream may help attackers to perform key recovery attack.

Chapter 6

Security Issues in the Initialisation Process of Stream Ciphers

Keystream generator of stream ciphers produces pseudorandom sequence aiming to mimic the one-time-pad (OTP). OTP uses a keystream which is randomly chosen and whose length is equal to the length of the messages and never used again for another encryption [98, p.27] [102, 103]. It is a secure cipher and proved by Shannon [92]. Keystream generator takes secret key and IV as input to form an initial state before keystream generation begin. A well-designed initialisation process should ensure that each key-IV pair generates a distinct and unpredictable session key (initial state), and then from this a distinct and unpredictable keystream. The analyses in Chapters 3, 4 and 5 and other results reported in the existing literature demonstrate flaws in both the loading and diffusion phases of the initialisation processes of several ciphers as discussed in Chapter 2, so it is clear that this aim is not always achieved.

Flaws may appear during either the loading phase, the diffusion phase or both. Examination of the flaws, based on the phases in which they occur, gives the ability to identify and understand the causes. From this detailed examination, we provide design guidelines to enable these flaws to be avoided or to at least reduce significantly the probability of occurrence.

Initialisation processes must be secure against at least the known generic attacks. The initialisation process should prevent secret key recovery attacks even when the session key (initial state) has been obtained (for example, through a state recovery attack). Therefore, the initialisation process should play a role like a security guard.

Other aspects to be considered when using stream ciphers are factors that can affect the efficiency of the initialisation process, especially for real-time applications. Although they are not our major focus. It is important to avoid processes which simultaneously decrease both efficiency and security. Some further observations about the security and efficiency of the initialisation process are provided in the remaining sections of this chapter. Within this context, the number of iterations for each specific design should be considered in terms of both the security and efficiency perspectives.

Table 6.1 summarises the results that have been found related flaws which are investigated in Chapters 3, 4 and 5. These flaws are compression, convergence, slid pairs, shifted keystream and weak Key-IV pairs. In this table, we considered our results related to A5/1, Sfinks and CSA-SC, and we also listed similar results from the literature for other ciphers. Concerns of these flaws are that may increase the vulnerability to some attacks such as TMTO attacks, differential attacks or ciphertext-only attack to recover the secret key as discussed in Chapters 3, 4 and 5. For example shifted slid pairs and weak Key-IV lead to key recovery attacks, and convergence can increase the vulnerability to TMTO attacks. Therefore, these flaws can make easy to use these attacks.

Table 6.1 shows the ciphers investigated in this thesis are highlighted in the table in **bold** font. The check-marks (✓) and ✅ indicate that the flaws have been identified: in the existing literature, and in our work in this research, respectively. The (x) mark indicates that the flaw does not exist for the particular cipher. The absence of a mark for a specific flaw means that the particular cipher has not yet been analysed for this flaw.

Chapter 6 is organised as follows: Section 6.1 describes flaws that have been identified in the initialisation processes of shift-register stream ciphers. Security aspects during the loading and diffusion phases are presented in Section 6.2 and Section 6.3, respectively. Section 6.4 presents recommendations for new proposals of initialisation processes. Section 6.5 summarises and concludes this chapter.

Table 6.1: Security flaws in the initialisation processes of specific stream ciphers

	RC4	*Trivium*	*Grain v1*	*Dragon*	*MICKEY v1*	*LILI-II*	**A5/1**	**Sfinks**	**CSA-SC**
Compression	x	x	x	x		✓[8]	✔[10]	x	✓[12]
Convergence		x	x		✓[6]	x[9]	✔[10]	✔[11]	✔[13]
Slid pairs		✓[2]	✓[3]		✓[7]		✔[10]	✔[11]	✔[13]
Shifted keystream		✓[2]	✓[3]		✓[7]		✔[10]	✔[11]	✔[13]
Weak key-IV	✓[1]		✓[4]		⋆		✔[10]	◇	⋆

⋆ theoretically, weak key-IV pairs exist due to the non-autonomous feedback during the initialisation process.

◇ further investigation is required to clarify weak Key-IV in Sfinks as discussed in Section 4.4.

[1] Based on the observation in [79] and [52]. They described a weakness in the second output of RC4 which is based on the secret key.

[2] Priemuth-Schmid and Biryukov [86] demonstrated slid pairs and shifted keystream in the Trivium cipher.

[3] Kuçuk [73] and Banik et al [10] demonstrated slid pairs and shifted keystream in Grain v1.

[4] Zhang and Wang [108] demonstrated some sliding weak key-IV combination.

[6] Hong and Kim [65] reported a state entropy loss in the MICKEY v1.

[7] Helleseth et al [61] examined the existence of slid pairs and shifted keystream.

[8] Biham and Dunkelman [20] reported Key-IV compression in LILI-II during the loading phase.

[9] Teo [100] reported that state convergence does not exist in the LILI family of stream ciphers.

[10] For A5/1, compression and State convergence are investigated in Section 3.2, Slid pairs and shifted keystream are examined in Section 3.3, weak Key-IV Combinations is discussed in Section 3.4, and the key recovery is covered in each of the previous sections.

[11] For Sfinks, State convergence is analyzed in Section 4.2, Slid pairs and shifted keystream are examined in Section 4.3, weak Key-IV Combinations is considered in Section 4.4, and the key recovery is covered in each of the previous sections.

[12] Simpson et al [96] commented on the compression of the Key-IV in the CSA-SC

[13] For the CSA-SC, State convergence is investigated in Section 5.2, and Slid pairs and shifted keystream are analyzed in Section 5.3,

6.1 Security Flaws in Initialisation Process

In this research, we investigated the initialisation processes of three stream ciphers, namely A5/1, Sfinks and CSA-SC. These ciphers have the same structure as most other stream ciphers, in that they use shift registers in combination with non-linear functions. They consist of a variety of structures and non-linear functions. Therefore, the recommendations and outcomes from these analyses are relevant to most stream ciphers that are based on shift registers. Moreover, the recommendations are considered based on our work and the literature.

The investigation focuses on the security of the initialisation process and flaws. There are some security flaws identified and discussed in this chapter. This section focuses in the identified security flaws and their causes in brief as follows:

Compression. Compression occurs when the sum of the key and IV sizes is greater than the internal state size and result in two or more Key-IV pairs generating the same loaded state.

Convergence. This condition occurs when two or more internal states map to the same internal state after applying a state update function for $\alpha \geq 1$ iterations. State convergence may result from using a state update function that is not one-to-one .

Slid pairs. Slid pairs may occur due to the similarity of the state update function during successive iterations of the initialisation process and is also affected by the form of the padding pattern. If any used when the key and IV bits are loaded into the internal state during the initialisation process.

Shifted keystream. Shifted keystream may occur for an identified slid pair due to the similarity or partial similarity between the state update functions during both the initialisation and keystream generation processes.

Weak Key-IV combinations. Non-autonomous operation for either loading or diffusion phases may result in one or more shift registers containing all-zero after the relevant phase.

The following sections (Sections 6.2 and 6.3) discuss more the properties and features of the loading and diffusion phases of the initialisation process thoroughly that could cause the above flaws.

6.2 Loading Phase and Security Flaws

The loading phase is the process of transferring the secret key and IV into the internal state of the keystream generator for a stream cipher. Note however that this does not happen in every case: for some stream ciphers, the IV is loaded during the diffusion phase (e.g. CSA-SC [104]). During the loading phase, the secret key and IV may be directly transferred to pre-determined stages of the internal state (e.g. Sfinks [29] and Trivium [43]), or the transfer may be achieved by using a linear state update function (e.g. A5/1 [30]), or a non-linear state update function (e.g. MICKEY [6–8]).

6.2.1 Padding Pattern Weaknesses

For ciphers in which the secret key, of length l and IV of length j are transferred directly to the internal state S of size s, where $l + j < s$, then some stages of S remain unfilled. Padding is required to fill these remaining stages. A specific padding pattern must be prescribed in the cipher proposal to obtain the loaded state. For example, Sfinks [29] uses a padding pattern which is all-zeros except one specific bit (which is one), and Grain [58–60] uses all-ones as the padding pattern.

The choice of padding pattern may affect the security of the initialisation process as, for some ciphers, this increases the ease with which slid pairs can occur and these in turn may result in shifted keystreams. Appropriate padding may defer the occurrences of slid pairs to further clocking steps and increase constraints that are required to obtain slid pairs and shifted keystream. This can clearly be seen for ciphers such as Trivium, Sfinks and Grain, as discussed below.

In the case of Trivium, as reported by Priemuth-Schmid and Biryukov in [86], slid pairs can not occur for less than 111 clocks. If the padding pattern was changed so that the last 3 stages are filled with zeros rather than ones, (identical pattern), then slid pairs could occur after only one clock, with probability of 1/2.

The Sfinks stream cipher is another example where the padding pattern and the state update function defer the first slid pair; in this case to after the 17th iteration. If the content of the stage s^{40} in Sfinks is changed to one in the padding format, then the occurrence of a slid pair will be deferred until the 24th iteration and the relevant conditions on the state contents will also be more complicated. In contrast, if the padding is all zeros (identical pattern), then slid pairs may occur after one iteration with high probability as shown in Section 4.3.

In another example, for the Grain cipher, the padding pattern is identical (all of them are ones) and the state update functions of the initialisation process and keystream generation have a degree of similarity. As shown by Zhang and Wang [108], a slid pair can be found after one clock and gives shifted keystream by one bit with probability of 2^{-2}. If the first bit or last bit of the 16-bit padding pattern, which is (1111111111111111), is changed to such as (0111111111111111) or (1111111111111110), then the slid pair will not occur until after 16 iterations, which would generate a shifted keystream with a 16 bit shift with probability of 2^{-32}. In summary, careful design of the padding patterns may defer the occurrence of the slid pairs and reduce its probability.

If the padding system is carefully designed (using extensive analysis), then it may make slid pairs that are generated by either the same secret key with different IVs, different secret keys with the same IV, or both, more complicated or need additional constraints to occur. For example, Priemuth-Schmid and Biryukov [86] discussed the search to find these two cases (same secret key and different IVs, or different secret keys and same IV) for the analysis of slid pairs of Trivium. They checked for clock-shifts from 111 to 142 and they could not find a solution for this system. Recall that the occurrence of slid pairs for the case of the same secret key with multiple IVs is the most important flaw for practical attacks, as shown in Section 2.7.3.

6.2.2 Loading Process and Flaws

Loading may be performed in a linear or non-linear manner. If the loading is performed using a linear function, then the loading function is one-to-one and there is no convergence during this phase. For example, A5/1 [30] loads the secret key and IV using a linear state update function (the LFSR feedback functions alone), so there is no state convergence during this phase. If the loading function is nonlinear, then the loading function must be examined to determine whether it is one-to-one or not. For example, MICKEY v1 [6] uses a nonlinear state update function to load the IV and secret key. MICKEY v1 suffers from state convergence during the loading and diffusion phases, as discussed by Helleseth et al [61].

Loading the secret key and IV into the internal state can be performed in one of these ways:

1. *Loading the key then IV.* The secret key is loaded first and then the IV using a state update function as shown in A5/1 stream cipher in Section 3.1. This loading process may result in state compression. If the state update function during the loading phase is not one-to-one, state convergence may occur during this phase. Each internal state may be a legitimate loaded state and that may lead to slid pairs after each clocking step. Another problem, it may end by all-zeros in one or more components of the cipher that is produced by weak key-IV combinations. For example, as shown in Section 3.4, the loading phase of A5/1 stream cipher may end by one, two or three registers containing all-zeros values.

2. *Loading the IV then secret key.* The IV is loaded first and then the secret key as shown in MICKEY stream cipher [6]. This loading process may result in some flaws: state compression, state convergence, slid pairs and weak key-IV combinations. Each internal state may be a legitimate loaded state and slid pairs may occur after each clocking step.

3. *Transferring the secret key and IV into the internal state simultaneously.* This loading process does not require more operations for the loading process. It transfers the key and IV bits directly into specific stages in the internal state such as Grain [58–60], Trivium [43] and Sfinks [29] stream ciphers. Provided the state space is large enough and the key and IV are loaded into different parts of the state, state compression and convergence cannot occur during this phase. Slid pairs may still occur. If a padding pattern is used, it should be considered carefully as discussed in Section 6.2.1, so all-zeros components in the loaded state may not occur during this phase and the occurrence of the slid pairs may be deferred to further clocking steps.

6.2.3 Autonomy in Loading Phase and Flaws

Ciphers are made up of several components, which can operate either interdependently or autonomously. Many of the security properties of the sequences produced by the components are only guaranteed under the condition that the components are operating autonomously. When this is not the case, more care needs to be taken to ensure that security problems are not introduced. For example, the well known properties of LFSR sequences depend on the LFSR being autonomous, having a primitive feedback function and a non-zero initial state. When a register is not autonomous then the possibility of an all-zero state can not be discounted and its properties can no longer be guaranteed.

Lack of autonomy during the initialisation process can arise in two ways: through non-autonomous feedback and through the use of non-autonomous clocking mechanisms. Non-autonomous feedback occurs when the content of the new stage of an internal state depends on an external input. Non-autonomous clocking mechanism is the situation when the clocking of any component of a cipher is governed by the contents of another component. These two types of non-autonomous mechanisms affect the security of the initialisation process in different ways.

During the loading phase, the non-autonomous feedback is more common than the use of non-autonomous clocking mechanisms. For example, the secret key and IV are transferred using state update function to form the loaded state using a non-autonomous feedback mechanism. This non-autonomous feedback mechanism of the loading phase is used in the A5/1 stream cipher [30]. It is a one-to-one function and there is no state convergence as shown in Section 3.2, but it introduces the weak key-IV problems as shown in Section 3.4. As another example, MICKEY v1 [6] uses a non-autonomous feedback mechanism in the loading phase. The non-autonomous function results in a non one-to-one state update function. So, state convergence may occur during the loading phase of MICKEY, as reported by Helleseth et al [61]. We defer discussion of non-autonomous clocking to Section 6.3 (Diffusion phase).

In non-autonomous feedback mechanism, the loading phase may load the secret key and IV using a state update function as mentioned above in the first and second loading methods of the loading process in Section 6.2.2. For each of these two loading methods, it is important to consider the behavior of the loading function to avoid any possibility of weak key-IV combinations that may occur at the end of the loading phase. For example, at the end of the loading phase of the A5/1 stream cipher, weak key-IV combination may result in all zero contents for one or more of the registers. Consequently, during the keystream generation process, those registers will generate a sequence of all zeros, resulting in a keystream that is either zeros or ones (when two or three registers contain all-zero values) as discussed in Section 3.4. Note that it is not practical to test this condition for each key-IV pair, due to the short time frame. Applying such a test for this condition will reduce the efficiency of the cipher.

In the case of transferring the key and IV directly into the internal state, weak key-IV can be avoided with non-zero key, IV or padding pattern that should be carefully designed to mix ones and zeros. This will prevent obtaining a zero internal state as the loaded state. For example, in Sfinks [29] there is one stage in the padding pattern that contains the value one. As well Trivium [43] contains three stages from the padding pattern that contains ones.

6.3 Diffusion Phase and Security Flaws

In the diffusion phase of the initialisation process, a number of iterations of the state update function are performed. The purpose of this phase is to diffuse the secret key and IV across the entire internal state. This section examines the required number of iterations and investigates the properties of the state update function that may affect the security of the diffusion phase.

Most stream ciphers use non-linear state update functions to perform the diffusion phase. This is important to prevent the secret key recovery attack for a given initial state (session key). Diffusion phases should be secure at least against the generic known attacks. For example, the diffusion phase should diffuse the key-IV across the entire internal state to provide resistance to differential attacks [20, 85]. This may be achieved by increasing the number of iterations.

6.3.1 Number of Iterations and Security

A common belief in symmetric key cryptography is that if the number of iterations during a nonlinear initialisation process is increased then the security of the cipher is increased. This belief is based on the concept that performing more mixing of the key-IV to prevent some attacks such as differential attacks. However, if state convergence occurs during the initialisation process, then increasing the number of iterations may decrease the number of obtainable initial states and may actually decrease the security and leave the stream cipher vulnerable to attacks such as TMTO attacks. This is the case of the A5/1 stream cipher as described in Section 3.2.

Increasing the number of iterations decreases the efficiency of the initialisation process. The qualitative curves in Figure 6.1 illustrate this concept of the impact of the number of iterations on the effective state space size due to the existence of state convergence, the complexity of attacks such as differential attacks and the efficiency of the initialisation process.

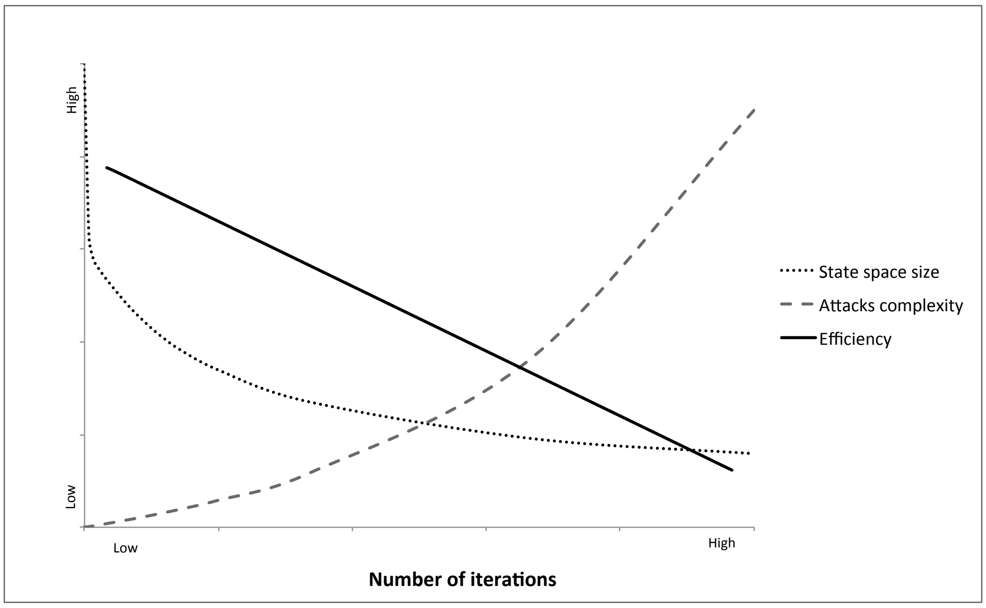


Figure 6.1: Qualitative effects of the state convergence

For each stream cipher, the optimal number of iterations during the initialisation process should be chosen carefully through extensive security analysis. It is necessary to determine the lower bound for complexities of attacks (e.g. differential, fault and algebraic attacks) at each iteration to draw the attack's complexity curve. State convergence and efficiency curves should be identified as a function of the number of iterations. Then these multidimensional curves should be drawn to decide the required number of iterations as a trade-off between these security and efficiency aspects.

If slid pairs and shifted keystreams occur during the initialisation process, then this may reduce the effective number of iterations during the diffusion phase and leave the stream cipher vulnerable to attack. For example, we illustrated slid pairs and shifted keystream in both A5/1 (in Section 3.3) and Sfinks (in Section 4.3) that may allow attackers to exploit this flaw to perform secret key recovery attacks.

6.3.2 Properties of State Update Function and Flaws

Properties of the state update function in the diffusion phase may affect the security of the initialisation process. These properties may introduce some security flaws such as state convergence, slid pairs and weak key-IV as shown in Chapters 3 to 5. This may reveal some information about the secret key given some keystream obtained from a particular session key (initial state). This section highlights the main points that have been identified in this research for the diffusion phase operation. These properties are: autonomous and non-autonomous, importance of one-to-one and the similarity in functions.

Autonomous Function

As discussed in Section 6.2.3, the components of stream ciphers can operate interdependently or autonomously during the loading phase. Also they can operate as well interdependently or autonomously during the diffusion phase based on the state update function and the structure of the initialisation process.

During diffusion phase, there are two reasons for lack of autonomy in stream ciphers: non-autonomous feedback and non-autonomous clocking mechanisms. Either both or one of these two types of non-autonomous mechanisms may occur in the diffusion phase of the initialisation process of some stream ciphers.

The non-autonomous feedback mechanism occurs during the diffusion phase of the initialisation process of stream ciphers when an external value is used as input to a component of the internal state. The external value may be a value which is from another component of stream cipher (such as another shift register, memory or output function). For example, for the Grain [58–60] stream cipher, during the diffusion phase the output bit of the h (output) function is fed back to both registers LFSR and NFSR and another external input into the NFSR which is from the LFSR register. In another example, CSA-SC as shown in Section 5.1 uses four output bits to feedback into register A, as well four bits from four Boolean functions are used as input to register B.

The non-autonomous feedback mechanism may result in undesirable initial state (session key) at the end of the diffusion phase. It may result in all zeros contents for the register that uses a non-autonomous feedback function. Therefore, this non-autonomous feedback mechanism may result in some key-IV combinations which are referred to as weak key-IV. A weak key-IV may be exploited by an attacker to perform secret key recovery. For example, in Grain stream cipher, Zhang and Wang [108] demonstrated some weak key-IV pairs that result in zeros contents in one register (LFSR) for the session key. Therefore, during the initialisation process, weak key-IV should be considered.

The non-autonomous clocking mechanism may occur during the diffusion phase of the initialisation process of stream ciphers where the clocking mechanism of a register depends of values from another component. For example, the A5/1 cipher as shown in Section 3.2 uses a majority clocking mechanism, where each register is clocked based on the whether a specific bit agrees with the majority value. This clocking mechanism may lead to a non one-to-one state update function which is discussed in the following section.

One-To-One Functions

A one-to-one state update function is a function where each distinct state maps to a distinct state after the update. That is, every state has a unique pre-image. In contrast, if many-to-one state update functions are used, the cipher may suffer from state convergence during the iteration of this function. State convergence may leave stream ciphers vulnerable to the attacks discussed in Section 2.7.1. Hence, state convergence from this point of view is an undesirable property of the initialisation process. To avoid the problem of state convergence, the state update function should be one-to-one. It is not only the individual components but also the combination of these components which should be one-to-one, as shown in Section 4.2. The disadvantage of the one-to-one function is that state recovery leads to the key recovery.

The many-to-one state update function can be caused by different ways. It can be caused by a non-autonomous clocking mechanism as illustrated in Section 3.2 for A5/1 stream cipher. It can be caused by the interaction between the components of the state update function which result in non one-to-one state update function while the individual components are one-to-one. This problem is discussed in Section 4.2 for the Sfinks stream cipher.

The one-to-one function during the initialisation and keystream generation processes has a advantage to prevent state convergence. Conversely, it has a draw back which is that state recovery may lead to key recovery. This is seen clearly in the Trivium stream cipher.

Similarity of Functions

The similarity of functions means that iterations of the state update functions of both the initialisation and keystream generation processes have commonalities. Similarity between the iterations of the state update function means that each iteration is identical. This similarity may lead to slid pairs after one or more of iterations. The similarity between the state update functions of the initialisation and keystream generation processes may lead to shifted keystream for given slid pairs.

Slid pairs and shifted keystreams can be obtained due to the similarity as mentioned above. This section discusses the effect of the diffusion phase on the similarity that may result in slid pairs and shifted keystreams. There are three conditions of similarity that are considered in Section 2.7.3 as follows:

a) similarity between iterations of the initialisation process.

b) similarity between iterations of the keystream generation process.

c) similarity between the state update functions for the initialisation and keystream generation processes.

Most stream ciphers satisfy conditions (a) and (b), but there is a wide variety in the extent to which condition (c) is satisfied.

As a result of similarity, the slid pairs can occur if a second loaded state is obtained from the first loaded state after a number of iterations. Therefore, slid pairs depends on both the loading and diffusion phases where the condition (a) is satisfied. Shifted keystream may occur if all three of the previous conditions are satisfied. So, shifted keystream for given slid pairs depends on the state update functions of both the diffusion phase and the keystream generation process. For example, the A5/1 stream cipher, as shown in Section 3.3, satisfies all the three conditions of similarity, so slid pairs and shifted keystream may occur after each clock and each slid pair produce a shifted keystream. Other examples include Sfinks (discussed in Section 4.3), CSA-SC (discussed in Section 5.3), Grain [10, 11, 45, 73, 76, 108], MICKEY [61] and Trivium [86] stream ciphers that satisfy the conditions (a) and (b). The third condition (c) is satisfied with some constraints for these stream ciphers. However, if the three conditions are satisfied for any cipher then this cipher may be vulnerable to slid pairs attacks.

6.4 Recommendations

It is an important that the initialisation process is secure and efficient for stream ciphers. A well-designed initialisation process should reduce the effect of security flaws and prevent attacks. Based on the ciphers examined in this thesis (Chapters 3, 4 and 5) and on previous analyses (Chapter 2), we present here some recommendations for the design of the initialisation process. These recommendations are for both the loading and diffusion phases as follows:

1- **State, key and IV sizes.** State size should be larger than 2.5 times the key size. Golić [54] reported that the internal size should be larger than the key size to prevent TMTO, and Babbage [9], and Biryukov and Shamir [23] stated that the state size should be twice the key size to prevent TMTO. After that Hong and Sarkar [66,67] in their revised TMTO attacks, reported that the IV size should be at least equal to the key size. Later on, Dunkelman and Keller [46] shown that the IV size should be at least 1.5 times the key size to prevent TMTO attacks.

   Based on the consideration in the 2nd recommendation (the next point), the total state size should be larger than the combined key and IV sizes to include a large padding pattern. So, the large padding pattern will reduce the probability of slid pairs to an acceptable level. To sum up the scattered recommendations in the literature, the state size should be more than 2.5 times the key size.

2- **Padding pattern.** The padding pattern should not be identical or cyclic. Identical means that the padding is all-zeros or all-ones, and cyclic means that the padding consists of repeated specific pattern. It is important to avoid these patterns in order to ensure that the same pattern can not be reproduced with fewer clocking steps than the maximum length of the padding pattern. This condition will then defer the occurrence of slid pairs to the maximum possible shift. In most cases, this will thereby reduce the chance of finding slid pairs and also reduce the probability of obtaining shifted keystreams.

3- **One-to-one functions.** The state update function should be one-to-one. Both the individual components and the combination of these components should be one-to-one. This is required to prevent the occurrence of state convergence. Conversely, this leads to another problem: inversion can be used in the case of state recovery to effect key recovery.

4- **Similarity of functions.** The similarity of the state update functions within and between the initialisation and keystream generation processes should be reduced. If this similarity is reduced then the occurrence of the slid pairs and shifted keystream will be reduced also. This reduction of the degree of similarity should be considered along with consideration for efficiency. It may affect efficiency negatively. De Cannière et al [42] recommended involving a counter in each step of the iterations. A degree of difference between the state update functions of the initialisation and keystream generation processes is recommended for stream ciphers. This recommendation is based on the accepted probability of the occurrence of slid pairs and shifted keystreams, and also on efficiency. If the initialisation state update function differs from that of the keystream generation process, then this may prevent shifted keystream and key recovery for a given state recovery but it may affect negatively the efficiency of the cipher. Therefore, the trade-off between these aspects is important.

5- **Non-autonomous feedback functions.** The use of non-autonomous feedback functions should be avoided or add another mechanism to prevent weak state. This consideration should prevent weak key-IV combination during the loading or diffusion phases. Weak session keys may leave the cipher vulnerable to attacks.

6- **Diffusion process.** The diffusion process should ensure that the key and IV bits are both diffused and distributed across the entire the state in a non-linear way. An appropriate diffusion process provides resistance against specific attacks which depend on the diffusion process, such as differential and fault attacks.

7- **Number of iterations.** Number of iterations, during the diffusion phase, should be in the optimal range. Increasing the number of iterations may increase the resistance to some attacks such as differential attacks but may leave the cipher vulnerable for another potential attack such as TMTO attacks (e.g. if state convergence is present). Increasing the number of iterations also decreases the efficiency of the cipher. Therefore, the number of iterations during the initialisation process should be based on extensive security analysis and the accepted efficiency. This process is a trade-off between the security aspects and efficiency.

8- **Integration between these features.** A trade-off between the previous features and properties must be considered. The study of these features and properties may have a degree of conflict (advantages and disadvantages). Therefore, the trade-off between these features and properties should be considered by the designers of initialisation process.

6.5 Summary

Stream cipher proposals usually include both design specifications and an analysis section outlining resistance against generic attacks. The focus of the security analysis is generally on the keystream generation function. Less attention is paid to the analysis of the initialisation process, though it is considered in some proposals. We recommend that stream cipher designers consider carefully the initialisation process, and perform sufficient analysis to ensure that both the loading and diffusion phases of this process are secure against at least the known attacks and prevent the known flaws discussed above.

Based on discussion above, we have provide a series of recommendations on the following aspects of stream cipher initialisation processes:

- **State, key and IV sizes.** State size should be larger than 2.5 times the key size.
- **Padding pattern.** The padding pattern should not be identical or cyclic.
- **One-to-one functions**.The state update function should be one-to-one.
- **Similarity of functions**. The similarity of the state update functions within and between the initialisation and keystream generation processes should be reduced.
- **Non-autonomous feedback functions**. The use of non-autonomous feedback functions should be avoided or add another mechanism to prevent weak state.

- **Diffusion process**. The diffusion process should ensure that the key and IV bits are both diffused and distributed across the entire the state in a non-linear way.
- **Number of iterations during diffusion process**. Number of iterations, during the diffusion phase, should be in the optimal range.
- **Integration between these properties and features**. A trade-off between the previous features and properties must be considered.

Chapter 7

Conclusion and Future Work

A well-designed initialisation process (comprising both loading and diffusion phases) should not reveal any information about the secret key, or possess properties that may help to facilitate attacks. The initialisation process should ensure that performing a key recovery attack is hard even if state recovery has occurred, because the mathematical relationships between the key-IV pair and the keystream are hard to establish. For real-time applications, efficient initialisation processes are required particularly since the initialisation process or rekeying is performed frequently. Thus, the initialisation process is significant with respect to both the security and efficiency of stream ciphers.

As expressed in Section 1.2, the main aim and objective of this research project was to examine the initialisation processes of shift-register based stream ciphers and identify features which might reduce the security of ciphers. This examination was performed by investigating the features and properties of the initialisation processes of three specific stream ciphers: A5/1, Sfinks and CSA-SC. From this perspective, specific features and properties of the loading and diffusion phases (the phases of the initialisation process) of these ciphers were investigated. The final aim of the research project was to use both the literature review and the results of the investigations outlined in this thesis to propose design criteria and recommendations for improved initialisation processes for shift-register based stream ciphers.

This research was conducted using both theoretical analysis and computer simulations. The theoretical analysis is based on the mathematical and statistical analysis of the initialisation processes of these three ciphers. The computer simulation is performed using C codes and the MAGMA software package. It is used to implement and examine the flaws in the initialisation processes and also to work with relevant systems of equations.

This chapter is organised as follows: Section 7.1 reviews the contributions of this thesis. Section 7.2 explores some directions for future work in the security and efficiency of the initialisation process of stream ciphers.

7.1 Review of Contributions

This section summarises the contributions to knowledge regarding the initialisation process of stream ciphers presented in this thesis. We have examined the initialisation process of three stream ciphers: A5/1, Sfinks and the Common Scrambling Algorithm Stream Cipher (CSA-SC). These three ciphers are all based on shift registers, and each cipher has its own specific features and properties. From this analysis and existing public literature, design criteria and recommendations of the initialisation processes of stream ciphers are provided.

7.1.1 Analysis of the Initialisation Process of A5/1

This contribution focused on the analysis of the initialisation process of a well-known stream cipher, A5/1 [30], which is used to provide confidentiality for GSM mobile phone communications. We have identified two new security flaws in this initialisation process: slid pairs and weak key-IVs, and further quantified the extent of a third flaw, namely state convergence,.

State Convergence in the A5/1 Stream Cipher

We extended Golić's work [54,55] for a further 5 iterations and gave an extrapolation of the number of distinct internal states after 100 iterations (at the end of the initialisation process). We show that the total number of distinct states after six iterations was reduced to approximately half of the total number of internal states, and estimate the number of the distinct states at the end of initialisation process to be approximately 5%, which is equivalent to a reduction in the key size by 4.3 bits.

Slid Pairs in the A5/1 Stream Cipher

In this cipher, the same state update function is used for both the initialisation and keystream generation processes. We showed that slid pairs may occur after each clocking step of the initialisation process and that each slid pair generates shifted keystream. In this work, we considered slid pairs after 1 and 2 clocking steps and showed that there are around 2^{45} and $2^{45.49}$ slid pairs after 1 and 2 clocking steps respectively. Moreover, we presented a ciphertext-only attack for the A5/1 stream cipher based on the slid pairs attacks that can recover the secret key.

Weak Key-IV pairs in the A5/1 Stream Cipher

This work focused on the non-autonomous feedback mechanism during the loading phase, where the secret key and IV bits are loaded sequentially using the state update function. The impact of the non-autonomous feedback mechanism on the content of the registers results in the possibility that one, two or three registers containing all-zeros. In the case where two or three registers contain all-zeros, the output keystream will be all-zeros or all-ones. We demonstrated a ciphertext-only attack on A5/1 in these cases.

7.1.2 Analysis of the Initialisation Process of Sfinks

This contribution analysed the initialisation process of the Sfinks stream cipher [29] submitted to the eSTREAM project [50] in PROFILE 2A. To our knowledge, analysis of the initialisation process of Sfinks has not appeared in the public literature. We have identified two security flaws in the Sfinks initialisation process: state convergence and slid pairs.

State Convergence in the Sfinks Stream Cipher

We showed that state convergence can occur even if the individual components of the state update function are one-to-one. The combination of the individual components should also be one-to-one to prevent the state convergence. However, this is not the case for the Sfinks stream cipher. We have demonstrated a state space reduction of Sfinks at the end of the initialisation process. Although the impact of this convergence on the security of Sfinks is minor, it can be avoided entirely by more careful design of the tap positions (so that the distance between any input and output of the S-box does not match the number of the delay steps).

Slid pairs in the Sfinks Stream Cipher

This work demonstrated the effect of the properties of the padding pattern and the state update functions of both the initialisation and keystream generation processes on the slid pairs attacks. The specification of padding pattern and the state update function of the initialisation process defer the occurrences of slid pairs in Sfinks to occur only after 17 iterations. For a given slid pair, shifted keystream may be obtained with a probability that is based on the similarity of these two state update functions.

This work also demonstrated the effect of the padding pattern on the slid pairs. We have examined two modified versions of the padding pattern of Sfinks. If the padding pattern is identical, which is all zeros (just change the content of one bit, s^{95}), then slid pairs may occur after any clocking step. This means slid pairs may occur after the 1st clocking step instead of the 17th and also with less constraints and higher probability. Conversely, if the content of the stage s^{40} in the padding format is changed to one, then the occurrence of slid pair will be deferred to 24 iterations and the relevant constraints on the state contents will also be more complicated.

7.1.3 Analysis of the Initialisation Process of CSA-SC

This research concentrated on the analysis of the initialisation process of a well-known stream cipher for the digital TV: CSA-SC [104]. Based on the public literature, there is no existing analysis of the initialisation process of CSA-SC. So, this is the first analysis of the initialisation process of the CSA-SC. The CSA-SC is word based in its operation (using 4-bit words). The IV bits are loaded during the diffusion phase. We have identified two security flaws in this initialisation process: state convergence and slid pairs.

State Convergence in the CSA-SC

We have demonstrated that the state update function during the initialisation process is not one-to-one, although the keystream generation process is one-to-one. So state convergence occurs only during the initialisation process. This is due to the non-autonomous feedback mechanism during the initialisation process, where the output bits from the output function are fed back into one of the registers. The state convergence may occur from the 3$^{\text{rd}}$ clocking step until the end of the diffusion phase, as the padding pattern prevents the state convergence for the 1$^{\text{st}}$ and 2$^{\text{nd}}$ clocking steps.

Slid Pairs in the CSA-SC

We have demonstrated the existence of slid pairs and shifted keystream in the CSA-SC. In this cipher, slid pairs may occur after the first clocking step. The word based operation affects the complexity of obtaining the slid pairs and shifted keystream. We have demonstrated that the highest probability of slid pairs occurs when the shifted steps are four (due to the matching of the number of insertion of IV words which is four times).

7.1.4 Criteria for the Initialisation Process of Stream Ciphers

An in-depth analysis of the features and properties of the loading and diffusion phases of the initialisation processes of shift-register based stream ciphers was presented. This analysis was based on the previous analysis in the literature as shown in Chapter 2 and our work in Chapters 3, 4 and 5. Based on this analysis, we also presented eight recommendations for the initialisation process that should minimise the occurrence of the flaws identified in this analysis and reduce the severity of their effects. These recommendations cover topics such as the size of the cipher's state space, the format of the loaded state and desirable properties of the state update functions used during the initialisation process. These recommendations should enhance the ability of future designers to design stream ciphers that are both secure and efficient.

7.2 Future Work

The initialisation process of stream ciphers has not been analysed thoroughly in the literature. Many stream cipher proposals seem to use an ad-hoc approach to the initialisation process and less attention has been paid to the security analysis of the initialisation process than to the keystream generation process. However, the security and efficiency of the initialisation process are important aspects, with potentially serious impact. This research area needs more attention. An outline of possible future research work is given in this section.

The case when one register of A5/1 contains all-zeros could be exploited in an attack if it can be distinguished efficiently. As presented in Section 3.4.3 the standard statistical tests applied to A5/1 were not able to distinguish between the case where a single register contains all-zeros and the case where no register contain all-zeros. Further analysis into this may find a possible distinguisher where a weak key-IV combination of this sort is used.

The flaw of weak Key-IV pairs in Sfinks cipher needs further investigation. At the end of the initialisation process of Sfinks may or may not end with zero state (including the memory pipeline) as discussed in Section 4.4. As we argue, it may occur with negligible probability and complicated conditions. Further investigation is required to clarify the occurrence of weak Key-IV.

The analysis of the state convergence of the CSA-SC when $q^{t-1} = 1$ is left for future work. It is a potential research point to perform theoretical analysis to find out the proportion of state convergence when $q^{t-1} = 1$. It is recommended to perform theoretical analysis and computer simulation using a number of random key-IV combinations to find out the number of distinct initial states (session keys). This will be the total number of distinct states for the random key-IVs (for both $q^{t-1} = 0$ and 1).

For a state update function, we have recommended this function to be one-to-one to avoid state convergence. Moreover, this function should be nonlinear to increase the complexity of inverting. This difficult to invert property ensures that the performing of the key recovery attack is hard even the if state recovery has occurred. Now, the open problem is to construct a one-to-one function which is difficult to invert and which can be used for the state update process during the diffusion phase of stream ciphers.

The initialisation processes of stream ciphers requires more analysis. This investigation may add more knowledge for types of flaw identified in this thesis or identify other flaws and their causes that have not yet been addressed. Therefore, we recommend further investigation into the initialisation processes of the stream ciphers that are presented in Table 6.1.

Security and efficiency are not always independent properties. Therefore, the analysis of the initialisation process should be considered in terms of security and efficiency (trade-off between them). We recommend to consider the analysis of the efficiency in the loading and diffusion phases.

The security of the diffusion phase depends on the state update function and the number of iterations performed before the keystream generation begins. Increasing the number of iterations may prevent some generic attacks such as differential attacks. From this research, we have demonstrated that increasing the number of iterations may reduce the state space if state convergence exists, which may leave the cipher vulnerable to TMTO attacks. Increasing the number of iterations also reduces the efficiency of the initialisation process. The optimal number of iterations in the diffusion phase for each cipher is another recommended area for further research. From this point of view, development of a tool to help estimate the optimal number of iterations performed during the diffusion phase would be a worthwhile goal.

Studying the impact of flaws such as state convergence and the existence of slid pairs on other generic attacks is another significant area related to the analysis of initialisation processes. Theses flaws may facilitate some generic attacks; for example state convergence may reduce the complexity of TMTO attacks. On the other hand, some properties may reduce the occurrence or the impact of some flaws in the same time these properties may leave the cipher vulnerable for other attacks.

The flaws that have been studied in this thesis are: state convergence, slid pairs and weak key-IV combinations. It is important to investigate their impact if they occur in the same cipher at the same operation (for example, for the same conversation of the GSM system). For example state convergence and slid pair may occur in the same cipher during the initialisation process. This may cause a further reduction of the attack complexity of the cipher and may leave the cipher vulnerable to other attacks.

This research investigated flaws during the initialisation processes of the shift-register based stream ciphers. Other ciphers that are not based on shift-register (e.g. RC4 stream cipher) need investigation to identify flaws that may occur during the initialisation processes.

Appendix A

A5/1 Algebraic Representation

A.1 The Key and IV Relationship for Slid Pairs

The key-IV combinations of the first and second states are described for the operation cases of A5/1 stream cipher as shown in Section 3.3 as follows:

Clocking A, B and C registers (case 1)

This case assumes that the contents of the three stages s_a^8, s_b^{10} and s_c^{10} are the same ($s_a^8 = s_b^{10} = s_c^{10}$). This assumption is included in the key dependent calculation that is presented later.

The secret key for both pairs are the same (according to our assumption). The relationship between the first IV, v, and the second IV, v' is computed and shown as follows.
Note: $c_0, c_1 \ldots c_{21}$ are secret key dependent constants (each c_i is obtained by XORing some of key bits). They can be calculated easily using the Gaussian elimination of the system of equations.

$\delta_0 = c_0 \oplus v_1$

$\delta_1 = c_1 \oplus v_0 \oplus v_1 \oplus v_2$

$\delta_2 = c_2 \oplus v_2 \oplus v_3$

$\delta_3 = c_3 \oplus v_3 \oplus v_4$

$\delta_4 = c_4 \oplus v_4 \oplus v_5$

$\delta_5 = c_5 \oplus v_5 \oplus v_6$

$\delta_6 = c_6 \oplus v_6 \oplus v_7$

$\delta_7 = c_7 \oplus v_7 \oplus v_8$

$\delta_8 = c_8 \oplus v_8 \oplus v_9$

$\delta_9 = c_9 \oplus v_9 \oplus v_{10}$

$\delta_{10} = c_{10} \oplus v_{10} \oplus v_{11}$

$\delta_{11} = c_{11} \oplus v_{11} \oplus v_{12}$

$\delta_{12} = c_{12} \oplus v_{12} \oplus v_{13}$

$\delta_{13} = c_{13} \oplus v_0 \oplus v_{13} \oplus v_{14}$

$\delta_{14} = c_{14} \oplus v_0 \oplus v_{14} \oplus v_{15}$

$\delta_{15} = c_{15} \oplus v_0 \oplus v_{15} \oplus v_{16}$

$\delta_{16} = c_{16} \oplus v_0 \oplus v_{16} \oplus v_{17}$

$\delta_{17} = c_{17} \oplus v_{17} \oplus v_{18}$

$\delta_{18} = c_{18} \oplus v_0 \oplus v_{18} \oplus v_{19}$

$\delta_{19} = c_{19} \oplus v_{19} \oplus v_{20}$

$\delta_{20} = c_{20} \oplus v_0 \oplus v_{20} \oplus v_{21}$

$\delta_{21} = c_{21} \oplus v_{21}$

From the Gaussian elimination of the remaining system equations of the matrix representation, there are 20 free key bits which are free to be chosen. These bits are $\{k_{42}$, k_{44}, k_{45}, k_{47}, k_{48}, k_{49}, k_{50}, k_{51}, k_{52}, k_{53}, k_{54}, k_{55}, k_{56}, k_{57}, k_{58}, k_{59}, k_{60}, k_{61}, $k_{62}, k_{63}\}$. Therefore, 44 bits are dependent on the 20 key bits and 4 IV bits, $\{v_0, v_3, v_{11}, v_{13}, \}$. Therefore, there is a slid pair shifted by one bit for the same key and two different IV's with probability of 2^{-44}.The following equations show how to form the 44 key bits from other 20 key bits and 4 IV bits.

$$
\begin{aligned}
k_0 =& k_{44} \oplus k_{45} \oplus k_{47} \oplus k_{50} \oplus k_{51} \oplus k_{54} \oplus k_{57} \oplus k_{62} \oplus v_{11} \oplus v_{13} \\
k_1 =& k_{42} \oplus k_{45} \oplus k_{50} \oplus k_{51} \oplus k_{52} \oplus k_{55} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus k_{63} \oplus v_0 \\
k_2 =& k_{44} \oplus k_{45} \oplus k_{47} \oplus k_{48} \oplus k_{50} \oplus k_{52} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{58} \oplus k_{59} \oplus k_{61} \oplus k_{62} \oplus \\
& k_{63} \oplus v_0 \oplus v_3 \oplus v_{11} \oplus v_{13} \\
k_3 =& k_{42} \oplus k_{48} \oplus k_{49} \oplus k_{50} \oplus k_{52} \oplus k_{53} \oplus k_{56} \oplus k_{59} \oplus k_{62} \oplus k_{63} \oplus v_{11} \oplus v_{13} \\
k_4 =& k_{42} \oplus k_{47} \oplus k_{48} \oplus k_{53} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{59} \oplus k_{61} \oplus k_{63} \oplus v_0 \oplus v_3 \\
k_5 =& k_{45} \oplus k_{48} \oplus k_{50} \oplus k_{52} \oplus k_{53} \oplus k_{55} \oplus k_{57} \oplus k_{59} \oplus k_{61} \oplus k_{63} \oplus v_0 \oplus v_3 \oplus v_{11} \oplus v_{13} \\
k_6 =& k_{42} \oplus k_{45} \oplus k_{49} \oplus k_{50} \oplus k_{52} \oplus k_{53} \oplus k_{55} \oplus k_{56} \oplus k_{58} \oplus k_{62} \oplus v_{11} \oplus v_{13} \\
k_7 =& k_{42} \oplus k_{44} \oplus k_{45} \oplus k_{47} \oplus k_{49} \oplus k_{51} \oplus k_{56} \oplus k_{58} \oplus k_{59} \oplus k_{61} \oplus v_0 \oplus v_3 \oplus v_{11} \oplus v_{13} \\
k_8 =& k_{42} \oplus k_{49} \oplus k_{50} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{57} \oplus k_{58} \oplus k_{60} \oplus k_{61} \oplus k_{63} \oplus v_0 \oplus v_3 \oplus \\
& v_{11} \oplus v_{13} \\
k_9 =& k_{42} \oplus k_{44} \oplus k_{49} \oplus k_{50} \oplus k_{52} \oplus k_{57} \oplus k_{59} \oplus k_{60} \oplus v_0 \oplus v_3 \\
k_{10} =& k_{44} \oplus k_{45} \oplus k_{47} \oplus k_{49} \oplus k_{50} \oplus k_{52} \oplus k_{55} \oplus k_{57} \oplus k_{63} \oplus v_3 \\
k_{11} =& k_{49} \oplus k_{51} \oplus k_{52} \oplus k_{54} \oplus k_{57} \oplus k_{58} \oplus k_{61} \oplus k_{62} \oplus v_0 \oplus v_{11} \oplus v_{13} \\
k_{12} =& k_{42} \oplus k_{48} \oplus k_{49} \oplus k_{50} \oplus k_{51} \oplus k_{52} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus k_{61} \oplus k_{63} \oplus v_0 \\
k_{13} =& k_{44} \oplus k_{45} \oplus k_{49} \oplus k_{50} \oplus k_{52} \oplus k_{55} \oplus k_{57} \oplus k_{60} \oplus k_{61} \oplus k_{62} \oplus v_3 \oplus v_{11} \oplus v_{13} \\
k_{14} =& k_{42} \oplus k_{47} \oplus k_{48} \oplus k_{49} \oplus k_{51} \oplus k_{52} \oplus k_{53} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{62} \oplus v_{11} \oplus v_{13} \\
k_{15} =& k_{42} \oplus k_{47} \oplus k_{51} \oplus k_{52} \oplus k_{53} \oplus k_{56} \oplus k_{57} \oplus k_{60} \oplus k_{61} \oplus k_{63} \oplus v_3 \\
k_{16} =& k_{45} \oplus k_{47} \oplus k_{50} \oplus k_{51} \oplus k_{54} \oplus k_{61} \oplus v_3 \oplus v_{11} \oplus v_{13} \\
k_{17} =& k_{42} \oplus k_{45} \oplus k_{47} \oplus k_{49} \oplus k_{50} \oplus k_{51} \oplus k_{55} \oplus k_{57} \oplus k_{61} \oplus k_{62} \oplus k_{63} \oplus v_{11} \oplus v_{13} \\
k_{18} =& k_{42} \oplus k_{48} \oplus k_{49} \oplus k_{50} \oplus k_{52} \oplus k_{53} \oplus k_{56} \oplus k_{61} \oplus v_3 \\
k_{19} =& k_{45} \oplus k_{47} \oplus k_{48} \oplus k_{49} \oplus k_{52} \oplus k_{54} \oplus k_{58} \oplus v_0 \oplus v_3 \oplus v_{11} \oplus v_{13} \\
k_{20} =& k_{42} \oplus k_{45} \oplus k_{48} \oplus k_{49} \oplus k_{51} \oplus k_{52} \oplus k_{53} \oplus k_{54} \oplus k_{59} \oplus k_{60} \oplus v_0 \oplus v_{11} \oplus v_{13} \\
k_{21} =& k_{42} \oplus k_{44} \oplus k_{45} \oplus k_{48} \oplus k_{49} \oplus k_{51} \oplus k_{52} \oplus k_{54} \oplus k_{58} \oplus k_{61} \oplus v_3 \oplus v_{11} \oplus v_{13} \\
k_{22} =& k_{42} \oplus k_{44} \oplus k_{45} \oplus k_{49} \oplus k_{54} \oplus k_{55} \oplus k_{58} \oplus k_{60} \oplus k_{62} \oplus v_3 \\
k_{23} =& k_{44} \oplus k_{45} \oplus k_{47} \oplus k_{52} \oplus k_{55} \oplus k_{58} \oplus k_{60} \oplus k_{61} \oplus k_{62} \oplus v_3 \\
k_{24} =& k_{49} \oplus k_{50} \oplus k_{52} \oplus k_{54} \oplus k_{57} \oplus k_{59} \oplus k_{63} \oplus v_{11} \oplus v_{13} \\
k_{25} =& k_{42} \oplus k_{47} \oplus k_{50} \oplus k_{51} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{58} \oplus k_{60} \oplus k_{62} \oplus k_{63} \oplus v_0 \\
k_{26} =& k_{44} \oplus k_{45} \oplus k_{48} \oplus k_{52} \oplus k_{53} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{61} \oplus k_{63} \oplus v_3 \oplus v_{11} \oplus v_{13}
\end{aligned}
$$

$$k_{27} = k_{42} \oplus k_{44} \oplus k_{45} \oplus k_{48} \oplus k_{51} \oplus k_{52} \oplus k_{53} \oplus k_{57} \oplus k_{58} \oplus k_{59} \oplus k_{61} \oplus k_{63} \oplus v_0$$

$$k_{28} = k_{45} \oplus k_{48} \oplus k_{49} \oplus k_{52} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{57} \oplus k_{59} \oplus k_{62} \oplus k_{63} \oplus v_0 \oplus v_3$$

$$k_{29} = k_{45} \oplus k_{47} \oplus k_{49} \oplus k_{51} \oplus k_{52} \oplus k_{54} \oplus k_{57} \oplus k_{58} \oplus k_{62} \oplus v_{11} \oplus v_{13}$$

$$k_{30} = k_{42} \oplus k_{52} \oplus k_{53} \oplus k_{55} \oplus k_{58} \oplus k_{60} \oplus k_{63} \oplus v_0$$

$$k_{31} = k_{44} \oplus k_{45} \oplus k_{51} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{58} \oplus k_{61} \oplus v_3 \oplus v_{11} \oplus v_{13}$$

$$k_{32} = k_{42} \oplus k_{44} \oplus k_{45} \oplus k_{48} \oplus k_{49} \oplus k_{51} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{57} \oplus k_{59} \oplus k_{61} \oplus k_{63}$$

$$k_{33} = k_{45} \oplus k_{47} \oplus k_{50} \oplus k_{51} \oplus k_{52} \oplus k_{54} \oplus k_{55} \oplus k_{57} \oplus k_{60} \oplus k_{61} \oplus k_{62} \oplus v_3$$

$$k_{34} = k_{44} \oplus k_{47} \oplus k_{49} \oplus k_{55} \oplus k_{58} \oplus k_{62} \oplus k_{63}$$

$$k_{35} = k_{45} \oplus k_{48} \oplus k_{50} \oplus k_{56} \oplus k_{59} \oplus k_{63} \oplus v_0$$

$$k_{36} = k_{44} \oplus k_{45} \oplus k_{47} \oplus k_{51} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{57} \oplus k_{59} \oplus k_{60} \oplus k_{61} \oplus v_0 \oplus \\ v_{11} \oplus v_{13}$$

$$k_{37} = k_{42} \oplus k_{44} \oplus k_{48} \oplus k_{49} \oplus k_{52} \oplus k_{54} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus k_{63} \oplus v_{11} \oplus v_{13}$$

$$k_{38} = k_{42} \oplus k_{44} \oplus k_{48} \oplus k_{50} \oplus k_{53} \oplus k_{55} \oplus k_{56} \oplus k_{62} \oplus v_0 \oplus v_3 \oplus v_{11} \oplus v_{13}$$

$$k_{39} = k_{42} \oplus k_{44} \oplus k_{45} \oplus k_{52} \oplus k_{55} \oplus k_{58} \oplus k_{60} \oplus k_{61} \oplus k_{62} \oplus k_{63} \oplus v_3$$

$$k_{40} = k_{44} \oplus k_{45} \oplus k_{47} \oplus k_{50} \oplus k_{52} \oplus k_{53} \oplus k_{58} \oplus k_{60} \oplus k_{61} \oplus v_0 \oplus v_3$$

$$k_{41} = k_{47} \oplus k_{48} \oplus k_{50} \oplus k_{52} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{59} \oplus k_{60} \oplus k_{61} \oplus k_{63} \oplus v_0 \oplus v_{11} \oplus v_{13}$$

$$k_{43} = k_{48} \oplus k_{49} \oplus k_{50} \oplus k_{51} \oplus k_{53} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus k_{61} \oplus k_{62} \oplus k_{63} \oplus v_3$$

$$k_{46} = k_{47} \oplus k_{48} \oplus k_{50} \oplus k_{53} \oplus k_{54} \oplus k_{56} \oplus k_{59} \oplus k_{60} \oplus k_{62} \oplus k_{63} \oplus v_0$$

Clocking *A* and *B* registers (case 2)

This case assumes that the contents of the two stages s_a^8 and s_b^{10} are the same but differ from s_c^{10}, ($s_a^8 = s_b^{10} \neq s_c^{10}$). This assumption is included in the key dependent calculation that is presented later.

The secret key for both pairs are the same (according to our assumption). The relationship between the first IV, v, and the second IV, v' is computed and shown as follows.

Note: $c'_0, c'_1 \ldots c'_{21}$ are secret key dependent constants (each c'_i is obtained by XORing some of key bits). They can be calculated using the Gaussian elimination of the system of equations.

$$\begin{aligned}
\delta_0 =& c'_0 \oplus v_1\\
\delta_1 =& c'_1 \oplus v_0 \oplus v_1 \oplus v_2\\
\delta_2 =& c'_2 \oplus v_2 \oplus v_3\\
\delta_3 =& c'_3 \oplus v_3 \oplus v_4\\
\delta_4 =& c'_4 \oplus v_4 \oplus v_5\\
\delta_5 =& c'_5 \oplus v_5 \oplus v_6\\
\delta_6 =& c'_6 \oplus v_6 \oplus v_7\\
\delta_7 =& c'_7 \oplus v_7 \oplus v_8\\
\delta_8 =& c'_8 \oplus v_8 \oplus v_9\\
\delta_9 =& c'_9 \oplus v_9 \oplus v_{10}\\
\delta_{10} =& c'_{10} \oplus v_{10} \oplus v_{11}\\
\delta_{11} =& c'_{11} \oplus v_{11} \oplus v_{12}\\
\delta_{12} =& c'_{12} \oplus v_{12} \oplus v_{13}\\
\delta_{13} =& c'_{13} \oplus v_0 \oplus v_{13} \oplus v_{14}\\
\delta_{14} =& c'_{14} \oplus v_0 \oplus v_{14} \oplus v_{15}\\
\delta_{15} =& c'_{15} \oplus v_0 \oplus v_{15} \oplus v_{16}\\
\delta_{16} =& c'_{16} \oplus v_0 \oplus v_{16} \oplus v_{17}\\
\delta_{17} =& c'_{17} \oplus v_{17} \oplus v_{18}\\
\delta_{18} =& c'_{18} \oplus v_0 \oplus v_{18} \oplus v_{19}\\
\delta_{19} =& c'_{19} \oplus v_{19} \oplus v_{20}\\
\delta_{20} =& c'_{20} \oplus v_0 \oplus v_{20} \oplus v_{21}\\
\delta_{21} =& c'_{21} \oplus v_{21}
\end{aligned}$$

From the Gaussian elimination of the remaining system equations, there are 22 free key bits which are free to be chosen. These bits are $\{k_{42}$, k_{43}, k_{44}, k_{45}, k_{47}, k_{48}, k_{49}, k_{50}, k_{51}, k_{52}, k_{53}, k_{54}, k_{55}, k_{56}, k_{57}, k_{58}, k_{59}, k_{60}, k_{61}, $k_{62}, k_{63}\}$. Therefore, there are 42 key bits that are dependent on other 22 key bits and 22 IV bits. So, there is a slid pair shifted by one bit for the same key and two different IV's with probability of 2^{-42}.The following equations show how to form the 42 key bits from other 22 key bits and 22 IV bits.

$$\begin{aligned}
k_0 =& k_{42} \oplus k_{44} \oplus k_{45} \oplus k_{51} \oplus k_{53} \oplus k_{55} \oplus k_{59} \oplus k_{60} \oplus k_{61} \oplus k_{63} \oplus v_1 \oplus v_2 \oplus v_3 \oplus v_4 \oplus \\
& v_6 \oplus v_7 \oplus v_9 \oplus v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{21} \oplus 1 \\
k_1 =& k_{42} \oplus k_{45} \oplus k_{47} \oplus k_{51} \oplus k_{52} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{57} \oplus k_{58} \oplus k_{60} \oplus k_{61} \oplus k_{63} \oplus \\
& v_1 \oplus v_2 \oplus v_3 \oplus v_4 \oplus v_5 \oplus v_9 \oplus v_{10} \oplus v_{11} \oplus v_{12} \oplus v_{15} \oplus v_{16} \oplus v_{19} \oplus v_{20} \oplus v_{21} \\
k_2 =& k_{43} \oplus k_{46} \oplus k_{48} \oplus k_{52} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{58} \oplus k_{59} \oplus k_{61} \oplus k_{62} \oplus v_0 \oplus \\
& v_2 \oplus v_3 \oplus v_4 \oplus v_5 \oplus v_6 \oplus v_{10} \oplus v_{11} \oplus v_{12} \oplus v_{13} \oplus v_{16} \oplus v_{17} \oplus v_{20} \oplus v_{21} \\
k_3 =& k_{44} \oplus k_{47} \oplus k_{49} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{57} \oplus k_{59} \oplus k_{60} \oplus k_{62} \oplus k_{63} \oplus v_1 \oplus \\
& v_3 \oplus v_4 \oplus v_5 \oplus v_6 \oplus v_7 \oplus v_{11} \oplus v_{12} \oplus v_{13} \oplus v_{14} \oplus v_{17} \oplus v_{18} \oplus v_{21} \\
k_4 =& k_{45} \oplus k_{48} \oplus k_{50} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{60} \oplus k_{61} \oplus k_{63} \oplus v_0 \oplus v_2 \oplus \\
& v_4 \oplus v_5 \oplus v_6 \oplus v_7 \oplus v_8 \oplus v_{12} \oplus v_{13} \oplus v_{14} \oplus v_{15} \oplus v_{18} \oplus v_{19} \\
k_5 =& k_{46} \oplus k_{49} \oplus k_{51} \oplus k_{55} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{59} \oplus k_{61} \oplus k_{62} \oplus v_0 \oplus v_1 \oplus v_3 \oplus \\
& v_5 \oplus v_6 \oplus v_7 \oplus v_8 \oplus v_9 \oplus v_{13} \oplus v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{19} \oplus v_{20} \\
k_6 =& k_{47} \oplus k_{50} \oplus k_{52} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus k_{62} \oplus k_{63} \oplus v_1 \oplus v_2 \oplus v_4 \oplus \\
& v_6 \oplus v_7 \oplus v_8 \oplus v_9 \oplus v_{10} \oplus v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{20} \oplus v_{21} \\
k_7 =& k_{48} \oplus k_{51} \oplus k_{53} \oplus k_{57} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus k_{61} \oplus k_{63} \oplus v_0 \oplus v_2 \oplus v_3 \oplus v_5 \oplus \\
& v_7 \oplus v_8 \oplus v_9 \oplus v_{10} \oplus v_{11} \oplus v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{18} \oplus v_{21} \\
k_8 =& k_{49} \oplus k_{52} \oplus k_{54} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus k_{61} \oplus k_{62} \oplus v_0 \oplus v_1 \oplus v_3 \oplus v_4 \oplus v_6 \oplus v_8 \oplus \\
& v_9 \oplus v_{10} \oplus v_{11} \oplus v_{12} \oplus v_{16} \oplus v_{17} \oplus v_{18} \oplus v_{19} \\
k_9 =& k_{50} \oplus k_{53} \oplus k_{55} \oplus k_{59} \oplus k_{60} \oplus k_{61} \oplus k_{62} \oplus k_{63} \oplus v_1 \oplus v_2 \oplus v_4 \oplus v_5 \oplus v_7 \oplus v_9 \oplus \\
& v_{10} \oplus v_{11} \oplus v_{12} \oplus v_{13} \oplus v_{17} \oplus v_{18} \oplus v_{19} \oplus v_{20} \\
k_{10} =& k_{51} \oplus k_{54} \oplus k_{56} \oplus k_{60} \oplus k_{61} \oplus k_{62} \oplus k_{63} \oplus v_0 \oplus v_2 \oplus v_3 \oplus v_5 \oplus v_6 \oplus v_8 \oplus v_{10} \oplus \\
& v_{11} \oplus v_{12} \oplus v_{13} \oplus v_{14} \oplus v_{18} \oplus v_{19} \oplus v_{20} \oplus v_{21} \\
k_{11} =& k_{52} \oplus k_{55} \oplus k_{57} \oplus k_{61} \oplus k_{62} \oplus k_{63} \oplus v_0 \oplus v_1 \oplus v_3 \oplus v_4 \oplus v_6 \oplus v_7 \oplus v_9 \oplus v_{11} \oplus \\
& v_{12} \oplus v_{13} \oplus v_{14} \oplus v_{15} \oplus v_{19} \oplus v_{20} \oplus v_{21} \\
k_{12} =& k_{53} \oplus k_{56} \oplus k_{58} \oplus k_{62} \oplus k_{63} \oplus v_0 \oplus v_1 \oplus v_2 \oplus v_4 \oplus v_5 \oplus v_7 \oplus v_8 \oplus v_{10} \oplus v_{12} \oplus \\
& v_{13} \oplus v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{20} \oplus v_{21} \\
k_{13} =& k_{54} \oplus k_{57} \oplus k_{59} \oplus k_{63} \oplus v_0 \oplus v_1 \oplus v_2 \oplus v_3 \oplus v_5 \oplus v_6 \oplus v_8 \oplus v_9 \oplus v_{11} \oplus v_{13} \oplus \\
& v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{21}
\end{aligned}$$

$$\begin{aligned}
k_{14} =& k_{42} \oplus k_{44} \oplus k_{45} \oplus k_{51} \oplus k_{53} \oplus k_{58} \oplus k_{59} \oplus k_{61} \oplus k_{63} \oplus v_0 \oplus v_{10} \oplus v_{12} \oplus v_{17} \oplus \\
& v_{18} \oplus v_{21} \oplus 1 \\
k_{15} =& k_{42} \oplus k_{45} \oplus k_{47} \oplus k_{51} \oplus k_{52} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus \\
& k_{63} \oplus v_7 \oplus v_8 \oplus v_9 \oplus v_{12} \oplus v_{13} \oplus v_{17} \oplus v_{18} \oplus v_{20} \oplus v_{21} \\
k_{16} =& k_{43} \oplus k_{46} \oplus k_{48} \oplus k_{52} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus k_{61} \oplus \\
& v_0 \oplus v_8 \oplus v_9 \oplus v_{10} \oplus v_{13} \oplus v_{14} \oplus v_{18} \oplus v_{19} \oplus v_{21} \\
k_{17} =& k_{42} \oplus k_{45} \oplus k_{47} \oplus k_{49} \oplus k_{51} \oplus k_{54} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{62} \oplus k_{63} \oplus v_2 \oplus v_3 \oplus \\
& v_4 \oplus v_6 \oplus v_7 \oplus v_{10} \oplus v_{11} \oplus v_{16} \oplus v_{19} \oplus v_{20} \oplus v_{21} \oplus 1 \\
k_{18} =& k_{44} \oplus k_{45} \oplus k_{47} \oplus k_{48} \oplus k_{50} \oplus k_{52} \oplus k_{55} \oplus k_{56} \oplus k_{60} \oplus k_{61} \oplus k_{62} \oplus k_{63} \oplus v_2 \oplus \\
& v_5 \oplus v_6 \oplus v_7 \oplus v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{19} \oplus v_{21} \oplus 1 \\
k_{19} =& k_{43} \oplus k_{44} \oplus k_{47} \oplus k_{48} \oplus k_{49} \oplus k_{51} \oplus k_{53} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus k_{63} \oplus v_2 \oplus v_4 \oplus \\
& v_7 \oplus v_{11} \oplus v_{12} \oplus v_{14} \oplus v_{19} \oplus 1 \\
k_{20} =& k_{42} \oplus k_{43} \oplus k_{44} \oplus k_{45} \oplus k_{46} \oplus k_{47} \oplus k_{48} \oplus k_{49} \oplus k_{50} \oplus k_{51} \oplus k_{52} \oplus k_{53} \oplus k_{54} \oplus \\
& k_{55} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus k_{61} \oplus k_{62} \oplus k_{63} \oplus v_1 \oplus v_3 \oplus v_5 \oplus v_7 \oplus \\
& v_9 \oplus v_{11} \oplus v_{13} \oplus v_{15} \oplus v_{17} \oplus v_{19} \oplus v_{21} \\
k_{21} =& k_{42} \oplus k_{43} \oplus k_{46} \oplus k_{47} \oplus k_{48} \oplus k_{49} \oplus k_{50} \oplus k_{52} \oplus k_{54} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{62} \oplus \\
& v_0 \oplus v_1 \oplus v_3 \oplus v_7 \oplus v_8 \oplus v_9 \oplus v_{10} \oplus v_{12} \oplus v_{15} \oplus v_{18} \oplus v_{20} \oplus v_{21} \oplus 1 \\
k_{22} =& k_{45} \oplus k_{46} \oplus k_{48} \oplus k_{49} \oplus k_{50} \oplus k_{51} \oplus k_{53} \oplus k_{55} \oplus k_{56} \oplus k_{60} \oplus k_{61} \oplus k_{62} \oplus k_{63} \oplus \\
& v_0 \oplus v_1 \oplus v_3 \oplus v_6 \oplus v_9 \oplus v_{10} \oplus v_{12} \oplus v_{13} \oplus v_{14} \oplus v_{15} \oplus v_{17} \oplus v_{20} \oplus v_{21} \oplus 1 \\
k_{23} =& k_{42} \oplus k_{43} \oplus k_{49} \oplus k_{50} \oplus k_{52} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{58} \oplus k_{61} \oplus v_2 \oplus v_4 \oplus v_8 \oplus \\
& v_9 \oplus v_{10} \oplus v_{12} \oplus v_{13} \oplus v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{18} \oplus v_{19} \oplus v_{20} \\
k_{24} =& k_{43} \oplus k_{44} \oplus k_{50} \oplus k_{51} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{59} \oplus k_{62} \oplus v_3 \oplus v_5 \oplus v_9 \oplus \\
& v_{10} \oplus v_{11} \oplus v_{13} \oplus v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{18} \oplus v_{19} \oplus v_{20} \oplus v_{21} \\
k_{25} =& k_{44} \oplus k_{45} \oplus k_{51} \oplus k_{52} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{57} \oplus k_{60} \oplus k_{63} \oplus v_4 \oplus v_6 \oplus v_{10} \oplus \\
& v_{11} \oplus v_{12} \oplus v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{18} \oplus v_{19} \oplus v_{20} \oplus v_{21} \\
k_{26} =& k_{45} \oplus k_{46} \oplus k_{52} \oplus k_{53} \oplus k_{55} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{61} \oplus v_0 \oplus v_5 \oplus v_7 \oplus v_{11} \oplus \\
& v_{12} \oplus v_{13} \oplus v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{18} \oplus v_{19} \oplus v_{20} \oplus v_{21}
\end{aligned}$$

$$\begin{aligned}
k_{27} =& k_{46} \oplus k_{47} \oplus k_{53} \oplus k_{54} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{59} \oplus k_{62} \oplus v_1 \oplus v_6 \oplus v_8 \oplus v_{12} \oplus \\
& v_{13} \oplus v_{14} \oplus v_{16} \oplus v_{17} \oplus v_{18} \oplus v_{19} \oplus v_{20} \oplus v_{21} \\
k_{28} =& k_{47} \oplus k_{48} \oplus k_{54} \oplus k_{55} \oplus k_{57} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus k_{63} \oplus v_2 \oplus v_7 \oplus v_9 \oplus v_{13} \oplus \\
& v_{14} \oplus v_{15} \oplus v_{17} \oplus v_{18} \oplus v_{19} \oplus v_{20} \oplus v_{21} \\
k_{29} =& k_{48} \oplus k_{49} \oplus k_{55} \oplus k_{56} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus k_{61} \oplus v_0 \oplus v_3 \oplus v_8 \oplus v_{10} \oplus v_{14} \oplus \\
& v_{15} \oplus v_{16} \oplus v_{18} \oplus v_{19} \oplus v_{20} \oplus v_{21} \\
k_{30} =& k_{49} \oplus k_{50} \oplus k_{56} \oplus k_{57} \oplus k_{59} \oplus k_{60} \oplus k_{61} \oplus k_{62} \oplus v_1 \oplus v_4 \oplus v_9 \oplus v_{11} \oplus v_{15} \oplus \\
& v_{16} \oplus v_{17} \oplus v_{19} \oplus v_{20} \oplus v_{21} \\
k_{31} =& k_{50} \oplus k_{51} \oplus k_{57} \oplus k_{58} \oplus k_{60} \oplus k_{61} \oplus k_{62} \oplus k_{63} \oplus v_2 \oplus v_5 \oplus v_{10} \oplus v_{12} \oplus v_{16} \oplus \\
& v_{17} \oplus v_{18} \oplus v_{20} \oplus v_{21} \\
k_{32} =& k_{51} \oplus k_{52} \oplus k_{58} \oplus k_{59} \oplus k_{61} \oplus k_{62} \oplus k_{63} \oplus v_0 \oplus v_3 \oplus v_6 \oplus v_{11} \oplus v_{13} \oplus v_{17} \oplus \\
& v_{18} \oplus v_{19} \oplus v_{21} \\
k_{33} =& k_{52} \oplus k_{53} \oplus k_{59} \oplus k_{60} \oplus k_{62} \oplus k_{63} \oplus v_0 \oplus v_1 \oplus v_4 \oplus v_7 \oplus v_{12} \oplus v_{14} \oplus v_{18} \oplus \\
& v_{19} \oplus v_{20} \\
k_{34} =& k_{53} \oplus k_{54} \oplus k_{60} \oplus k_{61} \oplus k_{63} \oplus v_0 \oplus v_1 \oplus v_2 \oplus v_5 \oplus v_8 \oplus v_{13} \oplus v_{15} \oplus v_{19} \oplus v_{20} \oplus v_{21} \\
k_{35} =& k_{42} \oplus k_{44} \oplus k_{45} \oplus k_{51} \oplus k_{53} \oplus k_{54} \oplus k_{59} \oplus k_{60} \oplus k_{62} \oplus k_{63} \oplus v_0 \oplus v_4 \oplus v_7 \oplus \\
& v_{15} \oplus v_{20} \oplus 1 \\
k_{36} =& k_{44} \oplus k_{47} \oplus k_{52} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{59} \oplus k_{62} \oplus k_{63} \oplus v_1 \oplus v_2 \oplus \\
& v_3 \oplus v_4 \oplus v_5 \oplus v_6 \oplus v_{11} \oplus v_{12} \oplus v_{14} \oplus v_{15} \oplus v_{17} \oplus v_{19} \oplus v_{20} \oplus v_{21} \oplus 1 \\
k_{37} =& k_{42} \oplus k_{43} \oplus k_{45} \oplus k_{46} \oplus k_{47} \oplus k_{48} \oplus k_{51} \oplus k_{59} \oplus k_{60} \oplus k_{62} \oplus v_1 \oplus v_2 \oplus v_3 \oplus \\
& v_4 \oplus v_5 \oplus v_6 \oplus v_8 \oplus v_9 \oplus v_{11} \oplus v_{13} \oplus v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{18} \oplus v_{19} \\
k_{38} =& k_{42} \oplus k_{43} \oplus k_{45} \oplus k_{46} \oplus k_{47} \oplus k_{48} \oplus k_{49} \oplus k_{51} \oplus k_{52} \oplus k_{53} \oplus k_{55} \oplus k_{59} \oplus v_1 \oplus \\
& v_5 \oplus v_{10} \oplus v_{12} \oplus v_{15} \oplus v_{17} \oplus v_{18} \oplus v_{19} \oplus v_{20} \oplus v_{21} \oplus 1 \\
k_{39} =& k_{42} \oplus k_{44} \oplus k_{48} \oplus k_{49} \oplus k_{50} \oplus k_{51} \oplus k_{52} \oplus k_{54} \oplus k_{55} \oplus k_{57} \oplus k_{58} \oplus k_{60} \oplus k_{62} \oplus \\
& k_{63} \oplus v_0 \oplus v_1 \oplus v_2 \oplus v_6 \oplus v_7 \oplus v_8 \oplus v_9 \oplus v_{12} \oplus v_{13} \oplus v_{16} \oplus v_{17} \oplus v_{18} \\
k_{40} =& k_{43} \oplus k_{45} \oplus k_{49} \oplus k_{50} \oplus k_{51} \oplus k_{52} \oplus k_{53} \oplus k_{55} \oplus k_{56} \oplus k_{58} \oplus k_{59} \oplus k_{61} \oplus k_{63} \oplus \\
& v_0 \oplus v_1 \oplus v_2 \oplus v_3 \oplus v_7 \oplus v_8 \oplus v_9 \oplus v_{10} \oplus v_{13} \oplus v_{14} \oplus v_{17} \oplus v_{18} \oplus v_{19} \\
k_{41} =& k_{42} \oplus k_{45} \oplus k_{46} \oplus k_{50} \oplus k_{52} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{57} \oplus k_{61} \oplus k_{62} \oplus k_{63} \oplus v_0 \oplus \\
& v_6 \oplus v_7 \oplus v_8 \oplus v_{10} \oplus v_{11} \oplus v_{16} \oplus v_{18} \oplus v_{19} \oplus v_{20} \oplus v_{21} \oplus 1
\end{aligned}$$

Clocking A and C registers (case 3)

This case assumes that the contents of the two stages s_a^8 and s_c^{10} are the same but differ from s_b^{10}, ($s_a^8 = s_c^{10} \neq s_b^{10}$). This assumption is included in the key dependent calculation that is presented later.

The secret key for both pairs are the same (according to our assumption). The relationship between the first IV, v, and the second IV, v' is computed, and the relationship between the first IV, v, and the second IV, v' is as follows.
Note: $e_0, e_1 \ldots e_{21}$ are secret key dependent constants (each e_i is obtained by XORing some of key bits). They can be calculated using the Gaussian elimination of the system of equations.

$$\begin{aligned}
\delta_0 =& e_0 \oplus v_0 \oplus v_1 \oplus v_{19} \oplus v_{21}\\
\delta_1 =& e_1 \oplus v_0 \oplus v_2 \oplus v_{19} \oplus v_{20} \oplus v_{21}\\
\delta_2 =& e_2 \oplus v_0 \oplus v_1 \oplus v_2 \oplus v_3 \oplus v_{19}\\
\delta_3 =& e_3 \oplus v_3 \oplus v_4\\
\delta_4 =& e_4 \oplus v_4 \oplus v_5\\
\delta_5 =& e_5 \oplus v_5 \oplus v_6\\
\delta_6 =& e_6 \oplus v_6 \oplus v_7\\
\delta_7 =& e_7 \oplus v_7 \oplus v_8\\
\delta_8 =& e_8 \oplus v_8 \oplus v_9\\
\delta_9 =& e_9 \oplus v_9 \oplus v_{10}\\
\delta_{10} =& e_{10} \oplus v_{10} \oplus v_{11}\\
\delta_{11} =& e_{11} \oplus v_{11} \oplus v_{12}\\
\delta_{12} =& e_{12} \oplus v_{12} \oplus v_{13}\\
\delta_{13} =& e_{13} \oplus v_0 \oplus v_{13} \oplus v_{14}\\
\delta_{14} =& e_{14} \oplus v_{14} \oplus v_{15} \oplus v_{19} \oplus v_{21}\\
\delta_{15} =& e_{15} \oplus v_0 \oplus v_1 \oplus v_{15} \oplus v_{16} \oplus v_{19} \oplus v_{20} \oplus v_{21}\\
\delta_{16} =& e_{16} \oplus v_1 \oplus v_{16} \oplus v_{17} \oplus v_{19}\\
\delta_{17} =& e_{17} \oplus v_0 \oplus v_{17} \oplus v_{18} \oplus v_{19} \oplus v_{21}
\end{aligned}$$

$$\delta_{18} = e_{18} \oplus v_1 \oplus v_{18} \oplus v_{19} \oplus v_{20}$$
$$\delta_{19} = e_{19} \oplus 0$$
$$\delta_{20} = e_{20} \oplus 0$$
$$\delta_{21} = e_{21} \oplus v_0 \oplus v_1 \oplus v_{19} \oplus v_{21}$$

From the the Gaussian elimination of the remaining system equations, there are 21 free key bits which are free to be chosen. These key bits are $\{k_{42}$, k_{44}, k_{45}, k_{47}, k_{48}, k_{49}, k_{50}, k_{51}, k_{52}, k_{53}, k_{54}, k_{55}, k_{56}, k_{57}, k_{58}, k_{59}, k_{60}, k_{61}, $k_{62}, k_{63}\}$. Therefore, 43 key bits are dependent on other 21 key bits and 22 IV bits. So, there is a slid pair shifted by one bit for the same key and two different IVs with probability of 2^{-43}. The following system of equations show how to form the 43 key bits from other 21 key bits and 22 IV bits.

$$k_0 = k_{42} \oplus k_{44} \oplus k_{47} \oplus k_{49} \oplus k_{50} \oplus k_{51} \oplus k_{53} \oplus k_{54} \oplus k_{60} \oplus k_{61} \oplus v_0 \oplus v_4 \oplus v_5 \oplus v_7 \oplus v_{12} \oplus v_{14} \oplus v_{17} \oplus v_{19} \oplus v_{20} \oplus v_{21}$$
$$k_1 = k_{44} \oplus k_{45} \oplus k_{46} \oplus k_{49} \oplus k_{50} \oplus k_{51} \oplus k_{53} \oplus k_{57} \oplus k_{58} \oplus v_3 \oplus v_6 \oplus v_{11} \oplus v_{12} \oplus v_{14} \oplus v_{17} \oplus v_{20} \oplus v_{21} \oplus 1$$
$$k_2 = k_{44} \oplus k_{46} \oplus k_{49} \oplus k_{51} \oplus k_{52} \oplus k_{53} \oplus k_{55} \oplus k_{56} \oplus k_{62} \oplus k_{63} \oplus v_2 \oplus v_6 \oplus v_7 \oplus v_9 \oplus v_{14} \oplus v_{16} \oplus v_{19} \oplus v_{21}$$
$$k_3 = k_{45} \oplus k_{47} \oplus k_{50} \oplus k_{52} \oplus k_{53} \oplus k_{54} \oplus k_{56} \oplus k_{57} \oplus k_{63} \oplus v_0 \oplus v_3 \oplus v_7 \oplus v_8 \oplus v_{10} \oplus v_{15} \oplus v_{17} \oplus v_{20}$$
$$k_4 = k_{46} \oplus k_{48} \oplus k_{51} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{57} \oplus k_{58} \oplus v_0 \oplus v_1 \oplus v_4 \oplus v_8 \oplus v_9 \oplus v_{11} \oplus v_{16} \oplus v_{18} \oplus v_{21}$$
$$k_5 = k_{47} \oplus k_{49} \oplus k_{52} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{58} \oplus k_{59} \oplus v_1 \oplus v_2 \oplus v_5 \oplus v_9 \oplus v_{10} \oplus v_{12} \oplus v_{17} \oplus v_{19}$$
$$k_6 = k_{48} \oplus k_{50} \oplus k_{53} \oplus k_{55} \oplus k_{56} \oplus k_{57} \oplus k_{59} \oplus k_{60} \oplus v_2 \oplus v_3 \oplus v_6 \oplus v_{10} \oplus v_{11} \oplus v_{13} \oplus v_{18} \oplus v_{20}$$
$$k_7 = k_{49} \oplus k_{51} \oplus k_{54} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{60} \oplus k_{61} \oplus v_3 \oplus v_4 \oplus v_7 \oplus v_{11} \oplus v_{12} \oplus v_{14} \oplus v_{19} \oplus v_{21}$$

$$
\begin{aligned}
k_8 =& k_{42} \oplus k_{44} \oplus k_{47} \oplus k_{49} \oplus k_{51} \oplus k_{52} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{57} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus \\
& k_{62} \oplus v_0 \oplus v_7 \oplus v_8 \oplus v_{13} \oplus v_{14} \oplus v_{15} \oplus v_{17} \oplus v_{19} \oplus v_{21} \\
k_9 =& k_{44} \oplus k_{45} \oplus k_{46} \oplus k_{49} \oplus k_{50} \oplus k_{56} \oplus k_{57} \oplus k_{59} \oplus k_{60} \oplus k_{62} \oplus k_{63} \oplus v_3 \oplus v_5 \oplus v_9 \\
& \oplus v_{11} \oplus v_{12} \oplus v_{13} \oplus v_{16} \oplus v_{17} \oplus v_{20} \oplus 1 \\
k_{10} =& k_{44} \oplus k_{46} \oplus k_{49} \oplus k_{51} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{57} \oplus k_{59} \oplus k_{60} \oplus k_{61} \oplus k_{62} \oplus v_0 \\
& \oplus v_2 \oplus v_9 \oplus v_{10} \oplus v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{19} \oplus v_{21} \\
k_{11} =& k_{45} \oplus k_{47} \oplus k_{50} \oplus k_{52} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{60} \oplus k_{61} \oplus k_{62} \oplus k_{63} \oplus v_1 \\
& \oplus v_3 \oplus v_{10} \oplus v_{11} \oplus v_{16} \oplus v_{17} \oplus v_{18} \oplus v_{20} \\
k_{12} =& k_{46} \oplus k_{48} \oplus k_{51} \oplus k_{53} \oplus k_{55} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{59} \oplus k_{61} \oplus k_{62} \oplus k_{63} \oplus v_0 \oplus v_2 \\
& \oplus v_4 \oplus v_{11} \oplus v_{12} \oplus v_{17} \oplus v_{18} \oplus v_{19} \oplus v_{21} \\
k_{13} =& k_{47} \oplus k_{49} \oplus k_{52} \oplus k_{54} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus k_{62} \oplus k_{63} \oplus v_0 \oplus v_1 \oplus v_3 \\
& \oplus v_5 \oplus v_{12} \oplus v_{13} \oplus v_{18} \oplus v_{19} \oplus v_{20} \\
k_{14} =& k_{42} \oplus k_{44} \oplus k_{47} \oplus k_{48} \oplus k_{49} \oplus k_{51} \oplus k_{54} \oplus k_{55} \oplus k_{57} \oplus k_{58} \oplus k_{59} \oplus k_{63} \oplus v_1 \oplus v_2 \\
& \oplus v_5 \oplus v_6 \oplus v_7 \oplus v_{12} \oplus v_{13} \oplus v_{17} \\
k_{15} =& k_{44} \oplus k_{45} \oplus k_{46} \oplus k_{50} \oplus k_{53} \oplus k_{54} \oplus k_{56} \oplus k_{57} \oplus k_{59} \oplus k_{60} \oplus k_{61} \oplus k_{62} \oplus v_0 \oplus v_1 \\
& \oplus v_2 \oplus v_5 \oplus v_6 \oplus v_7 \oplus v_{11} \oplus v_{12} \oplus v_{15} \oplus v_{17} \oplus 1 \\
k_{16} =& k_{44} \oplus k_{46} \oplus k_{49} \oplus k_{50} \oplus k_{51} \oplus k_{53} \oplus k_{56} \oplus k_{57} \oplus k_{59} \oplus k_{60} \oplus k_{61} \oplus v_1 \oplus v_3 \oplus v_4 \\
& \oplus v_7 \oplus v_8 \oplus v_9 \oplus v_{14} \oplus v_{15} \oplus v_{19} \\
k_{17} =& k_{42} \oplus k_{44} \oplus k_{45} \oplus k_{49} \oplus k_{52} \oplus k_{53} \oplus k_{57} \oplus k_{58} \oplus k_{62} \oplus v_0 \oplus v_2 \oplus v_7 \oplus v_8 \oplus v_9 \\
& \oplus v_{10} \oplus v_{12} \oplus v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{19} \oplus v_{21} \\
k_{18} =& k_{42} \oplus k_{45} \oplus k_{47} \oplus k_{48} \oplus k_{51} \oplus k_{52} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{57} \oplus k_{59} \oplus k_{60} \oplus k_{62} \oplus \\
& k_{63} \oplus v_0 \oplus v_4 \oplus v_7 \oplus v_9 \oplus v_{10} \oplus v_{16} \oplus v_{17} \oplus v_{19} \oplus v_{21} \oplus 1 \\
k_{19} =& k_{42} \oplus k_{44} \oplus k_{45} \oplus k_{51} \oplus k_{55} \oplus k_{57} \oplus k_{58} \oplus k_{59} \oplus k_{61} \oplus v_2 \oplus v_3 \oplus v_5 \oplus v_6 \oplus v_7 \\
& \oplus v_9 \oplus v_{10} \oplus v_{12} \oplus v_{14} \oplus v_{16} \oplus v_{17} \oplus v_{18} \oplus v_{21} \oplus 1 \\
k_{20} =& k_{47} \oplus k_{48} \oplus k_{50} \oplus k_{57} \oplus k_{58} \oplus k_{60} \oplus k_{61} \oplus k_{62} \oplus k_{63} \oplus v_1 \oplus v_2 \oplus v_5 \oplus v_7 \oplus v_9 \\
& \oplus v_{10} \oplus v_{13} \oplus v_{15} \oplus v_{16} \oplus v_{18} \oplus 1 \\
k_{21} =& k_{42} \oplus k_{45} \oplus k_{48} \oplus k_{49} \oplus k_{55} \oplus k_{56} \oplus k_{60} \oplus v_3 \oplus v_5 \oplus v_7 \oplus v_8 \oplus v_9 \oplus v_{10} \oplus v_{11} \\
& \oplus v_{13} \oplus v_{14} \oplus v_{15} \oplus v_{18} \oplus v_{19} \oplus v_{20} \oplus v_{21}
\end{aligned}
$$

$$\begin{aligned}
k_{22} =& k_{44} \oplus k_{48} \oplus k_{50} \oplus k_{52} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{58} \oplus k_{62} \oplus v_1 \oplus v_3 \oplus v_4 \oplus v_5 \\
& \oplus v_6 \oplus v_9 \oplus v_{10} \oplus v_{13} \oplus v_{16} \oplus v_{17} \oplus v_{18} \oplus v_{19} \oplus v_{20} \oplus v_{21} \oplus 1 \\
k_{23} =& k_{42} \oplus k_{49} \oplus k_{53} \oplus k_{54} \oplus k_{57} \oplus k_{58} \oplus k_{60} \oplus k_{61} \oplus k_{62} \oplus v_0 \oplus v_4 \oplus v_9 \oplus v_{10} \oplus v_{11} \\
& \oplus v_{13} \oplus v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{19} \\
k_{24} =& k_{44} \oplus k_{46} \oplus k_{48} \oplus k_{49} \oplus k_{50} \oplus k_{52} \oplus k_{53} \oplus k_{57} \oplus k_{59} \oplus k_{63} \oplus v_3 \oplus v_8 \oplus v_{10} \oplus v_{13} \\
& \oplus v_{16} \oplus v_{18} \oplus v_{20} \oplus 1 \\
k_{25} =& k_{42} \oplus k_{47} \oplus k_{49} \oplus k_{50} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{59} \oplus k_{61} \oplus k_{62} \oplus k_{63} \oplus v_2 \oplus v_4 \\
& \oplus v_5 \oplus v_6 \oplus v_7 \oplus v_{11} \oplus v_{13} \oplus v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{18} \oplus v_{19} \oplus v_{20} \\
k_{26} =& k_{42} \oplus k_{46} \oplus k_{47} \oplus k_{52} \oplus k_{54} \oplus k_{56} \oplus k_{58} \oplus k_{63} \oplus v_1 \oplus v_4 \oplus v_5 \oplus v_6 \oplus v_{11} \oplus v_{12} \\
& \oplus v_{13} \oplus v_{14} \oplus v_{16} \oplus v_{17} \oplus v_{18} \oplus 1 \\
k_{27} =& k_{42} \oplus k_{44} \oplus k_{45} \oplus k_{46} \oplus k_{47} \oplus k_{49} \oplus k_{51} \oplus k_{52} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{60} \oplus k_{63} \oplus v_1 \\
& \oplus v_3 \oplus v_5 \oplus v_8 \oplus v_9 \oplus v_{11} \oplus v_{13} \oplus v_{15} \oplus v_{16} \oplus v_{17} \\
& \oplus v_{18} \oplus v_{19} \oplus v_{20} \oplus v_{21} \oplus 1 \\
k_{28} =& k_{53} \oplus k_{55} \oplus k_{59} \oplus k_{63} \oplus v_0 \oplus v_1 \oplus v_3 \oplus v_5 \oplus v_8 \oplus v_{10} \oplus v_{11} \oplus v_{12} \oplus v_{14} \oplus v_{18} \\
& \oplus v_{20} \oplus v_{21} \oplus 1 \\
k_{29} =& k_{44} \oplus k_{45} \oplus k_{47} \oplus k_{49} \oplus k_{50} \oplus k_{53} \oplus k_{55} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus k_{62} \oplus k_{63} \oplus v_0 \oplus v_1 \\
& \oplus v_{11} \oplus v_{14} \oplus v_{16} \oplus v_{18} \oplus v_{21} \\
k_{30} =& k_{45} \oplus k_{46} \oplus k_{48} \oplus k_{50} \oplus k_{51} \oplus k_{54} \oplus k_{56} \oplus k_{59} \oplus k_{60} \oplus k_{61} \oplus k_{63} \oplus v_0 \oplus v_1 \oplus v_2 \\
& \oplus v_{12} \oplus v_{15} \oplus v_{17} \oplus v_{19} \\
k_{31} =& k_{46} \oplus k_{47} \oplus k_{49} \oplus k_{51} \oplus k_{52} \oplus k_{55} \oplus k_{57} \oplus k_{60} \oplus k_{61} \oplus k_{62} \oplus v_0 \oplus v_1 \oplus v_2 \oplus v_3 \\
& \oplus v_{13} \oplus v_{16} \oplus v_{18} \oplus v_{20} \\
k_{32} =& k_{47} \oplus k_{48} \oplus k_{50} \oplus k_{52} \oplus k_{53} \oplus k_{56} \oplus k_{58} \oplus k_{61} \oplus k_{62} \oplus k_{63} \oplus v_1 \oplus v_2 \oplus v_3 \oplus v_4 \\
& \oplus v_{14} \oplus v_{17} \oplus v_{19} \oplus v_{21} \\
k_{33} =& k_{48} \oplus k_{49} \oplus k_{51} \oplus k_{53} \oplus k_{54} \oplus k_{57} \oplus k_{59} \oplus k_{62} \oplus k_{63} \oplus v_0 \\
& \oplus v_2 \oplus v_3 \oplus v_4 \oplus v_5 \oplus v_{15} \oplus v_{18} \oplus v_{20} \\
k_{34} =& k_{49} \oplus k_{50} \oplus k_{52} \oplus k_{54} \oplus k_{55} \oplus k_{58} \oplus k_{60} \oplus k_{63} \oplus v_0 \oplus v_1 \oplus v_3 \oplus v_4 \oplus v_5 \oplus v_6 \\
& \oplus v_{16} \oplus v_{19} \oplus v_{21} \\
k_{35} =& k_{42} \oplus k_{44} \oplus k_{47} \oplus k_{49} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{59} \oplus k_{60} \oplus v_1 \oplus v_2 \oplus v_6 \oplus v_{12} \oplus \\
& v_{14} \oplus v_{19} \oplus v_{21}
\end{aligned}$$

$$\begin{aligned}
k_{36} =& k_{42} \oplus k_{45} \oplus k_{46} \oplus k_{47} \oplus k_{51} \oplus k_{52} \oplus k_{56} \oplus k_{58} \oplus k_{61} \oplus k_{62} \oplus v_0 \oplus v_1 \\
&\oplus v_2 \oplus v_4 \oplus v_8 \oplus v_{11} \oplus v_{18} \oplus v_{19} \oplus v_{21} \oplus 1 \\
k_{37} =& k_{42} \oplus k_{44} \oplus k_{45} \oplus k_{47} \oplus k_{49} \oplus k_{51} \oplus k_{54} \oplus k_{56} \oplus k_{60} \oplus k_{62} \oplus v_0 \oplus v_5 \\
&\oplus v_6 \oplus v_7 \oplus v_8 \oplus v_{11} \oplus v_{14} \oplus v_{16} \oplus v_{19} \oplus v_{21} \oplus 1 \\
k_{38} =& k_{42} \oplus k_{44} \oplus k_{49} \oplus k_{50} \oplus k_{51} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{59} \oplus k_{60} \oplus k_{61} \\
&\oplus v_0 \oplus v_2 \oplus v_3 \oplus v_{11} \oplus v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{21} \oplus 1 \\
k_{39} =& k_{46} \oplus k_{47} \oplus k_{48} \oplus k_{51} \oplus k_{54} \oplus k_{55} \oplus k_{59} \oplus k_{60} \oplus k_{62} \oplus k_{63} \oplus v_2 \oplus v_5 \oplus v_6 \\
&\oplus v_8 \oplus v_9 \oplus v_{11} \oplus v_{12} \oplus v_{15} \oplus v_{18} \oplus v_{19} \oplus 1 \\
k_{40} =& k_{42} \oplus k_{45} \oplus k_{47} \oplus k_{48} \oplus k_{49} \oplus k_{51} \oplus k_{52} \oplus k_{58} \oplus k_{59} \oplus k_{62} \oplus v_2 \oplus v_3 \oplus v_5 \\
&\oplus v_{10} \oplus v_{12} \oplus v_{15} \oplus v_{17} \oplus v_{18} \oplus v_{19} \oplus v_{21} \\
k_{41} =& k_{44} \oplus k_{50} \oplus k_{54} \oplus k_{55} \oplus k_{57} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus k_{61} \oplus k_{62} \oplus k_{63} \oplus v_1 \oplus v_4 \\
&\oplus v_5 \oplus v_6 \oplus v_8 \oplus v_{12} \oplus v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{19} \oplus v_{20} \oplus 1 \\
k_{43} =& k_{44} \oplus k_{46} \oplus k_{48} \oplus k_{49} \oplus k_{52} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{57} \oplus k_{58} \oplus k_{61} \oplus k_{62} \oplus v_1 \\
&\oplus v_3 \oplus v_5 \oplus v_8 \oplus v_{11} \oplus v_{12} \oplus v_{13} \oplus v_{14} \oplus v_{15} \oplus v_{17} \oplus v_{18} \oplus 1
\end{aligned}$$

Clocking B and C registers (case 4)

This case assumes that the contents of the two stages s_b^{10} and s_c^{10} are the same but differ from s_a^8, ($s_a^8 \neq s_b^{10} = s_c^{10}$). This assumption is included in the key dependent calculation that is presented later.

The secret key for both pairs are the same (according to our assumption). The relationship between the first IV, v, and the second IV, v' is computed and shown as follows: Note: $e'_0, e'_1 \ldots e'_{21}$ are secret key dependent constants (each e'_i is obtained by XORing some of key bits). They can be calculated using the Gaussian elimination of the system of equations.

$$
\begin{aligned}
\delta_0 &= e'_0 \oplus v_0 \oplus v_{19} \oplus v_{21}\\
\delta_1 &= e'_1 \oplus v_1 \oplus v_{19} \oplus v_{20} \oplus v_{21}\\
\delta_2 &= e'_2 \oplus v_0 \oplus v_1 \oplus v_{19}\\
\delta_3 &= e'_3 \oplus 0\\
\delta_4 &= e'_4 \oplus 0\\
\delta_5 &= e'_5 \oplus 0\\
\delta_6 &= e'_6 \oplus 0\\
\delta_7 &= e'_7 \oplus 0\\
\delta_8 &= e'_8 \oplus 0\\
\delta_9 &= e'_9 \oplus 0\\
\delta_{10} &= e'_{10} \oplus 0\\
\delta_{11} &= e'_{11} \oplus 0\\
\delta_{12} &= e'_{12} \oplus 0\\
\delta_{13} &= e'_{13} \oplus 0\\
\delta_{14} &= e'_{14} \oplus v_0 \oplus v_{19} \oplus v_{21}\\
\delta_{15} &= e'_{15} \oplus v_1 \oplus v_{19} \oplus v_{20} \oplus v_{21}\\
\delta_{16} &= e'_{16} \oplus v_0 \oplus v_1 \oplus v_{19}\\
\delta_{17} &= e'_{17} \oplus v_0 \oplus v_{19} \oplus v_{21}\\
\delta_{18} &= e'_{18} \oplus v_0 \oplus v_1 \oplus v_{20}\\
\delta_{19} &= e'_{19} \oplus v_{19} \oplus v_{20}\\
\delta_{20} &= e'_{20} \oplus v_0 \oplus v_{20} \oplus v_{21}\\
\delta_{21} &= e'_{21} \oplus v_0 \oplus v_1 \oplus v_{19}
\end{aligned}
$$

From the Gaussian elimination of the remaining system equations, there are 20 free key bits which are free to be chosen. These key bits are $\{k_{44}, k_{45}, k_{47}, k_{48}, k_{49}, k_{50}, k_{51}, k_{52}, k_{53}, k_{54}, k_{55}, k_{56}, k_{57}, k_{58}, k_{59}, k_{60}, k_{61}, k_{62}, k_{63}\}$. Therefore, there are 42 key bits are dependent on other 22 key bits and 22 IV bits. Therefore, there is a slid pair shifted by one bit for the same key and two different IVs with probability of 2^{-42}. The following system of equations show how to form the 42 key bits from other 22 key bits and 22 IV bits.

$$\begin{aligned}
k_0 =& k_{44} \oplus k_{45} \oplus k_{47} \oplus k_{48} \oplus k_{50} \oplus k_{51} \oplus k_{53} \oplus k_{54} \oplus k_{56} \oplus k_{57} \oplus k_{60} \oplus k_{61} \oplus k_{62} \oplus \\
& k_{63} \oplus v_4 \oplus v_5 \oplus v_{12} \oplus v_{14} \oplus v_{15} \oplus v_{18} \\
k_1 =& k_{44} \oplus k_{46} \oplus k_{47} \oplus k_{49} \oplus k_{50} \oplus k_{52} \oplus k_{53} \oplus k_{55} \oplus k_{56} \oplus k_{58} \oplus k_{60} \oplus v_0 \oplus v_4 \oplus v_6 \oplus \\
& v_{12} \oplus v_{13} \oplus v_{14} \oplus v_{16} \oplus v_{18} \oplus v_{19} \\
k_2 =& k_{44} \oplus k_{59} \oplus k_{60} \oplus k_{62} \oplus k_{63} \oplus v_1 \oplus v_4 \oplus v_7 \oplus v_{12} \oplus v_{13} \oplus v_{17} \oplus v_{18} \oplus v_{19} \oplus v_{20} \\
k_3 =& k_{45} \oplus k_{46} \oplus k_{47} \oplus k_{48} \oplus k_{50} \oplus k_{51} \oplus k_{52} \oplus k_{56} \oplus k_{57} \oplus k_{59} \oplus v_0 \oplus v_3 \oplus v_7 \oplus v_8 \oplus \\
& v_9 \oplus v_{11} \oplus v_{15} \oplus v_{16} \oplus v_{19} \oplus 1 \\
k_4 =& k_{44} \oplus k_{45} \oplus k_{46} \oplus k_{49} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{58} \oplus k_{59} \oplus k_{61} \oplus k_{62} \oplus v_1 \oplus v_2 \oplus \\
& v_4 \oplus v_7 \oplus v_{11} \oplus v_{12} \oplus v_{13} \oplus v_{16} \oplus v_{17} \oplus v_{18} \oplus 1 \\
k_5 =& k_{45} \oplus k_{46} \oplus k_{47} \oplus k_{52} \oplus k_{53} \oplus k_{54} \oplus k_{60} \oplus k_{62} \oplus v_3 \oplus v_4 \oplus v_7 \oplus v_9 \oplus v_{10} \oplus v_{11} \oplus \\
& v_{15} \oplus v_{17} \oplus v_{18} \oplus v_{19} \oplus v_{20} \oplus 1 \\
k_6 =& k_{51} \oplus k_{54} \oplus k_{57} \oplus k_{60} \oplus k_{63} \oplus v_3 \oplus v_5 \oplus v_{11} \oplus v_{19} \oplus v_{20} \\
k_7 =& k_{44} \oplus k_{45} \oplus k_{46} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{58} \oplus k_{59} \oplus k_{61} \oplus k_{62} \oplus v_0 \oplus v_2 \oplus v_3 \oplus v_6 \oplus \\
& v_7 \oplus v_9 \oplus v_{11} \oplus v_{13} \oplus v_{16} \oplus 1 \\
k_8 =& k_{45} \oplus k_{48} \oplus k_{51} \oplus k_{53} \oplus k_{54} \oplus k_{56} \oplus k_{57} \oplus k_{59} \oplus k_{61} \oplus k_{62} \oplus k_{63} \oplus v_1 \oplus v_7 \oplus v_{14} \oplus \\
& v_{16} \oplus v_{17} \oplus v_{18} \oplus v_{21} \\
k_9 =& k_{44} \oplus k_{45} \oplus k_{49} \oplus k_{53} \oplus k_{55} \oplus k_{57} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus k_{63} \oplus v_0 \oplus v_3 \oplus v_4 \oplus v_7 \oplus \\
& v_8 \oplus v_9 \oplus v_{11} \oplus v_{12} \oplus v_{13} \oplus v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{18} \oplus v_{19} \oplus v_{20} \oplus v_{21} \oplus 1 \\
k_{10} =& k_{45} \oplus k_{46} \oplus k_{52} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{58} \oplus k_{60} \oplus k_{61} \oplus k_{63} \oplus v_0 \oplus v_1 \oplus v_2 \oplus v_7 \oplus \\
& v_{11} \oplus v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{18} \oplus v_{19} \oplus v_{21} \oplus 1 \\
k_{11} =& k_{44} \oplus k_{45} \oplus k_{46} \oplus k_{48} \oplus k_{51} \oplus k_{52} \oplus k_{53} \oplus k_{56} \oplus k_{57} \oplus k_{60} \oplus v_0 \oplus v_1 \oplus v_3 \oplus v_7 \oplus \\
& v_9 \oplus v_{10} \oplus v_{11} \oplus v_{12} \oplus v_{13} \oplus v_{16} \oplus v_{17} \oplus v_{19} \oplus 1 \\
k_{12} =& k_{44} \oplus k_{47} \oplus k_{49} \oplus k_{50} \oplus k_{52} \oplus k_{53} \oplus k_{55} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{61} \oplus k_{62} \oplus k_{63} \oplus \\
& v_1 \oplus v_2 \oplus v_3 \oplus v_4 \oplus v_5 \oplus v_{11} \oplus v_{12} \oplus v_{13} \oplus v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{18} \oplus v_{20} \oplus v_{21} \\
k_{13} =& k_{44} \oplus k_{47} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus k_{61} \oplus v_0 \oplus v_2 \oplus v_3 \oplus v_6 \oplus v_{13} \oplus v_{15} \oplus v_{16} \oplus v_{17} \oplus \\
& v_{19} \oplus v_{21} \\
k_{14} =& k_{45} \oplus k_{46} \oplus k_{47} \oplus k_{50} \oplus k_{51} \oplus k_{52} \oplus k_{56} \oplus k_{57} \oplus k_{62} \oplus k_{63} \oplus v_1 \oplus v_2 \oplus v_4 \oplus \\
& v_5 \oplus v_9 \oplus v_{11} \oplus v_{13} \oplus v_{15} \oplus v_{17} \oplus v_{21} \oplus 1 \\
k_{15} =& k_{52} \oplus k_{55} \oplus k_{58} \oplus k_{60} \oplus k_{61} \oplus k_{63} \oplus v_0 \oplus v_2 \oplus v_4 \oplus v_5 \oplus v_6 \oplus v_8 \oplus v_{14} \oplus v_{21}
\end{aligned}$$

$$k_{16} = k_{53} \oplus k_{56} \oplus k_{59} \oplus k_{61} \oplus k_{62} \oplus v_0 \oplus v_1 \oplus v_3 \oplus v_5 \oplus v_6 \oplus v_7 \oplus v_9 \oplus v_{15}$$

$$k_{17} = k_{44} \oplus k_{45} \oplus k_{46} \oplus k_{52} \oplus k_{53} \oplus k_{57} \oplus k_{59} \oplus k_{60} \oplus k_{63} \oplus v_1 \oplus v_3 \oplus v_6 \oplus v_8 \oplus v_9 \oplus v_{10} \oplus v_{11} \oplus v_{12} \oplus v_{13} \oplus v_{20} \oplus v_{21} \oplus 1$$

$$k_{18} = k_{45} \oplus k_{46} \oplus k_{47} \oplus k_{50} \oplus k_{52} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus k_{61} \oplus k_{63} \oplus v_0 \oplus v_5 \oplus v_8 \oplus v_{15} \oplus v_{20} \oplus v_{21} \oplus 1$$

$$k_{19} = k_{56} \oplus k_{59} \oplus k_{62} \oplus v_0 \oplus v_1 \oplus v_3 \oplus v_4 \oplus v_6 \oplus v_8 \oplus v_9 \oplus v_{10} \oplus v_{12} \oplus v_{18}$$

$$k_{20} = k_{57} \oplus k_{60} \oplus k_{63} \oplus v_1 \oplus v_2 \oplus v_4 \oplus v_5 \oplus v_7 \oplus v_9 \oplus v_{10} \oplus v_{11} \oplus v_{13} \oplus v_{19}$$

$$k_{21} = k_{44} \oplus k_{45} \oplus k_{46} \oplus k_{52} \oplus k_{53} \oplus k_{54} \oplus k_{58} \oplus k_{59} \oplus k_{61} \oplus k_{62} \oplus v_0 \oplus v_4 \oplus v_5 \oplus v_6 \oplus v_7 \oplus v_8 \oplus v_9 \oplus v_{10} \oplus v_{13} \oplus v_{14} \oplus v_{16} \oplus v_{21} \oplus 1$$

$$k_{22} = k_{45} \oplus k_{46} \oplus k_{47} \oplus k_{50} \oplus k_{52} \oplus k_{54} \oplus k_{56} \oplus k_{60} \oplus k_{62} \oplus v_1 \oplus v_2 \oplus v_4 \oplus v_6 \oplus v_{12} \oplus v_{13} \oplus v_{17} \oplus v_{20} \oplus 1$$

$$k_{23} = k_{44} \oplus k_{45} \oplus k_{47} \oplus k_{48} \oplus k_{50} \oplus k_{51} \oplus k_{53} \oplus k_{54} \oplus k_{56} \oplus k_{57} \oplus k_{61} \oplus k_{62} \oplus v_2 \oplus v_7 \oplus v_8 \oplus v_{10} \oplus v_{13} \oplus v_{15} \oplus v_{16} \oplus v_{18}$$

$$k_{24} = k_{45} \oplus k_{47} \oplus k_{49} \oplus k_{50} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus k_{61} \oplus k_{62} \oplus v_2 \oplus v_5 \oplus v_7 \oplus v_8 \oplus v_{13} \oplus v_{15} \oplus v_{17} \oplus v_{18} \oplus v_{19} \oplus v_{20} \oplus v_{21} \oplus 1$$

$$k_{25} = k_{47} \oplus k_{50} \oplus k_{53} \oplus k_{56} \oplus k_{59} \oplus k_{62} \oplus k_{63} \oplus v_4 \oplus v_6 \oplus v_9 \oplus v_{10} \oplus v_{12} \oplus v_{14} \oplus v_{19} \oplus v_{20}$$

$$k_{26} = k_{48} \oplus k_{51} \oplus k_{54} \oplus k_{57} \oplus k_{60} \oplus k_{63} \oplus v_0 \oplus v_5 \oplus v_7 \oplus v_{10} \oplus v_{11} \oplus v_{13} \oplus v_{15} \oplus v_{20} \oplus v_{21}$$

$$k_{27} = k_{44} \oplus k_{45} \oplus k_{46} \oplus k_{49} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{58} \oplus k_{59} \oplus k_{61} \oplus k_{62} \oplus v_0 \oplus v_1 \oplus v_2 \oplus v_3 \oplus v_4 \oplus v_6 \oplus v_7 \oplus v_8 \oplus v_9 \oplus v_{13} \oplus v_{14} \oplus v_{20} \oplus 1$$

$$k_{28} = k_{45} \oplus k_{46} \oplus k_{47} \oplus k_{52} \oplus k_{53} \oplus k_{54} \oplus k_{60} \oplus k_{62} \oplus v_1 \oplus v_3 \oplus v_{11} \oplus v_{12} \oplus v_{13} \oplus v_{20} \oplus v_{21} \oplus 1$$

$$k_{29} = k_{46} \oplus k_{47} \oplus k_{48} \oplus k_{50} \oplus k_{52} \oplus k_{54} \oplus k_{56} \oplus k_{59} \oplus k_{61} \oplus v_5 \oplus v_7 \oplus v_8 \oplus v_9 \oplus v_{10} \oplus v_{11} \oplus v_{15} \oplus v_{20} \oplus v_{21} \oplus 1$$

$$k_{30} = k_{47} \oplus k_{48} \oplus k_{49} \oplus k_{50} \oplus k_{51} \oplus k_{52} \oplus k_{56} \oplus k_{57} \oplus k_{59} \oplus k_{60} \oplus k_{62} \oplus k_{63} \oplus v_2 \oplus v_4 \oplus v_5 \oplus v_6 \oplus v_7 \oplus v_{13} \oplus v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{20} \oplus v_{21} \oplus 1$$

$$k_{31} = k_{46} \oplus k_{47} \oplus k_{49} \oplus k_{50} \oplus k_{52} \oplus k_{55} \oplus k_{58} \oplus k_{63} \oplus v_0 \oplus v_4 \oplus v_5 \oplus v_6 \oplus v_7 \oplus v_{10} \oplus v_{12} \oplus v_{14} \oplus v_{15} \oplus v_{17} \oplus v_{18}$$

$$
\begin{aligned}
k_{32} =& k_{47} \oplus k_{48} \oplus k_{50} \oplus k_{51} \oplus k_{53} \oplus k_{56} \oplus k_{59} \oplus v_0 \oplus v_1 \oplus v_5 \oplus v_6 \oplus v_7 \oplus v_8 \\
& \oplus v_{11} \oplus v_{13} \oplus v_{15} \oplus v_{16} \oplus v_{18} \oplus v_{19} \\
k_{33} =& k_{44} \oplus k_{45} \oplus k_{46} \oplus k_{48} \oplus k_{49} \oplus k_{51} \oplus k_{53} \oplus k_{57} \oplus k_{59} \oplus k_{60} \oplus k_{62} \oplus v_1 \oplus v_3 \\
& \oplus v_4 \oplus v_6 \oplus v_8 \oplus v_{11} \oplus v_{13} \oplus v_{14} \oplus v_{17} \oplus v_{19} \oplus v_{21} \oplus 1 \\
k_{34} =& k_{44} \oplus k_{47} \oplus k_{49} \oplus k_{52} \oplus k_{55} \oplus k_{56} \oplus k_{58} \oplus k_{60} \oplus k_{61} \oplus k_{62} \oplus v_2 \oplus v_3 \oplus v_4 \\
& \oplus v_7 \oplus v_8 \oplus v_9 \oplus v_{10} \oplus v_{12} \oplus v_{16} \oplus v_{18} \oplus v_{20} \oplus v_{21} \\
k_{35} =& k_{45} \oplus k_{46} \oplus k_{47} \oplus k_{51} \oplus k_{52} \oplus k_{53} \oplus k_{60} \oplus k_{62} \oplus v_2 \oplus v_4 \oplus v_7 \oplus v_8 \oplus v_{10} \\
& \oplus v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{18} \oplus v_{19} \oplus v_{20} \oplus 1 \\
k_{36} =& k_{44} \oplus k_{45} \oplus k_{46} \oplus k_{51} \oplus k_{53} \oplus k_{55} \oplus k_{57} \oplus k_{59} \oplus k_{60} \oplus k_{62} \oplus k_{63} \oplus v_2 \oplus v_3 \\
& \oplus v_5 \oplus v_7 \oplus v_{10} \oplus v_{13} \oplus v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{19} \oplus v_{21} \oplus 1 \\
k_{37} =& k_{45} \oplus k_{46} \oplus k_{47} \oplus k_{50} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus k_{61} \oplus v_0 \oplus v_2 \\
& \oplus v_3 \oplus v_5 \oplus v_6 \oplus v_7 \oplus v_9 \oplus v_{10} \oplus v_{12} \oplus v_{13} \oplus v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{18} \oplus 1 \\
k_{38} =& k_{53} \oplus k_{54} \oplus k_{56} \oplus k_{57} \oplus k_{59} \oplus k_{62} \oplus v_1 \oplus v_6 \oplus v_7 \oplus v_{11} \oplus v_{12} \oplus v_{13} \oplus v_{14} \oplus \\
& v_{17} \oplus v_{19} \oplus v_{21} \\
k_{39} =& k_{44} \oplus k_{45} \oplus k_{46} \oplus k_{52} \oplus k_{53} \oplus k_{55} \oplus k_{57} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus k_{62} \oplus k_{63} \oplus v_3 \\
& \oplus v_4 \oplus v_8 \oplus v_9 \oplus v_{11} \oplus v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{18} \oplus v_{21} \oplus 1 \\
k_{40} =& k_{45} \oplus k_{46} \oplus k_{47} \oplus k_{50} \oplus k_{52} \oplus k_{54} \oplus k_{55} \oplus k_{58} \oplus k_{60} \oplus k_{61} \oplus v_0 \oplus v_2 \oplus v_7 \\
& \oplus v_8 \oplus v_{11} \oplus v_{13} \oplus v_{14} \oplus v_{16} \oplus v_{17} \oplus v_{19} \oplus v_{20} \oplus 1 \\
k_{41} =& k_{44} \oplus k_{45} \oplus k_{46} \oplus k_{52} \oplus k_{53} \oplus k_{54} \oplus k_{56} \oplus k_{57} \oplus k_{60} \oplus v_1 \oplus v_2 \oplus v_3 \oplus v_7 \\
& \oplus v_{10} \oplus v_{11} \oplus v_{12} \oplus v_{13} \oplus v_{14} \oplus v_{15} \oplus v_{17} \oplus v_{21} \oplus 1 \\
k_{42} =& k_{44} \oplus k_{46} \oplus k_{48} \oplus k_{51} \oplus k_{52} \oplus k_{53} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus k_{62} \oplus v_3 \oplus v_4 \oplus v_7 \\
& \oplus v_9 \oplus v_{10} \oplus v_{12} \oplus v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{20} \oplus 1 \\
k_{43} =& k_{44} \oplus k_{46} \oplus k_{47} \oplus k_{49} \oplus k_{50} \oplus k_{52} \oplus k_{53} \oplus k_{55} \oplus k_{56} \oplus k_{59} \oplus k_{60} \oplus k_{61} \oplus k_{62} \\
& \oplus v_3 \oplus v_4 \oplus v_{11} \oplus v_{13} \oplus v_{14} \oplus v_{17}
\end{aligned}
$$

A.2 The Key and IV Relationship for Weak Key-IV

The weak Key-IV combinations as described in Section 3.4 are expressed in the three scenarios of the operation of A5/1 as follows.

First scenario: The relationship between key and IV bits

To freeze the three registers A, B and C, the conditions for the system of equations that are required to result in registers A, B and C containing all-zero values can be described as follows:

$$k_0 = v_0 \oplus v_1 \oplus v_5 \oplus v_6 \oplus v_8 \oplus v_{11} \oplus v_{18}$$
$$k_1 = v_1 \oplus v_2 \oplus v_6 \oplus v_7 \oplus v_9 \oplus v_{12} \oplus v_{19}$$
$$k_2 = v_2 \oplus v_3 \oplus v_7 \oplus v_8 \oplus v_{10} \oplus v_{13} \oplus v_{20}$$
$$k_3 = v_3 \oplus v_4 \oplus v_8 \oplus v_9 \oplus v_{11} \oplus v_{14} \oplus v_{21}$$
$$k_4 = v_4 \oplus v_5 \oplus v_9 \oplus v_{10} \oplus v_{12} \oplus v_{15}$$
$$k_5 = v_5 \oplus v_6 \oplus v_{10} \oplus v_{11} \oplus v_{13} \oplus v_{16}$$
$$k_6 = v_6 \oplus v_7 \oplus v_{11} \oplus v_{12} \oplus v_{14} \oplus v_{17}$$
$$k_7 = v_7 \oplus v_8 \oplus v_{12} \oplus v_{13} \oplus v_{15} \oplus v_{18}$$
$$k_8 = v_0 \oplus v_1 \oplus v_5 \oplus v_6 \oplus v_9 \oplus v_{11} \oplus v_{13} \oplus v_{14} \oplus v_{16} \oplus v_{18} \oplus v_{19}$$
$$k_9 = v_1 \oplus v_2 \oplus v_6 \oplus v_7 \oplus v_{10} \oplus v_{12} \oplus v_{14} \oplus v_{15} \oplus v_{17} \oplus v_{19} \oplus v_{20}$$
$$k_{10} = v_2 \oplus v_3 \oplus v_7 \oplus v_8 \oplus v_{11} \oplus v_{13} \oplus v_{15} \oplus v_{16} \oplus v_{18} \oplus v_{20} \oplus v_{21}$$
$$k_{11} = v_3 \oplus v_4 \oplus v_8 \oplus v_9 \oplus v_{12} \oplus v_{14} \oplus v_{16} \oplus v_{17} \oplus v_{19} \oplus v_{21}$$
$$k_{12} = v_4 \oplus v_5 \oplus v_9 \oplus v_{10} \oplus v_{13} \oplus v_{15} \oplus v_{17} \oplus v_{18} \oplus v_{20}$$
$$k_{13} = v_5 \oplus v_6 \oplus v_{10} \oplus v_{11} \oplus v_{14} \oplus v_{16} \oplus v_{18} \oplus v_{19} \oplus v_{21}$$
$$k_{14} = v_0 \oplus v_1 \oplus v_5 \oplus v_7 \oplus v_8 \oplus v_{12} \oplus v_{15} \oplus v_{17} \oplus v_{18} \oplus v_{19} \oplus v_{20}$$
$$k_{15} = v_1 \oplus v_2 \oplus v_6 \oplus v_8 \oplus v_9 \oplus v_{13} \oplus v_{16} \oplus v_{18} \oplus v_{19} \oplus v_{20} \oplus v_{21}$$
$$k_{16} = v_2 \oplus v_3 \oplus v_7 \oplus v_9 \oplus v_{10} \oplus v_{14} \oplus v_{17} \oplus v_{19} \oplus v_{20} \oplus v_{21}$$
$$k_{17} = v_0 \oplus v_1 \oplus v_3 \oplus v_4 \oplus v_5 \oplus v_6 \oplus v_{10} \oplus v_{15} \oplus v_{20} \oplus v_{21}$$
$$k_{18} = v_0 \oplus v_2 \oplus v_4 \oplus v_7 \oplus v_8 \oplus v_{16} \oplus v_{18} \oplus v_{21}$$

$$k_{19} = v_0 \oplus v_3 \oplus v_6 \oplus v_9 \oplus v_{11} \oplus v_{17} \oplus v_{18} \oplus v_{19}$$

$$k_{20} = v_1 \oplus v_4 \oplus v_7 \oplus v_{10} \oplus v_{12} \oplus v_{18} \oplus v_{19} \oplus v_{20}$$

$$k_{21} = v_2 \oplus v_5 \oplus v_8 \oplus v_{11} \oplus v_{13} \oplus v_{19} \oplus v_{20} \oplus v_{21}$$

$$k_{22} = v_0 \oplus v_1 \oplus v_3 \oplus v_5 \oplus v_8 \oplus v_9 \oplus v_{11} \oplus v_{12} \oplus v_{14} \oplus v_{18} \oplus v_{20} \oplus v_{21}$$

$$k_{23} = v_0 \oplus v_2 \oplus v_4 \oplus v_5 \oplus v_8 \oplus v_9 \oplus v_{10} \oplus v_{11} \oplus v_{12} \oplus v_{13} \oplus v_{15} \oplus v_{18} \oplus v_{19} \oplus v_{21}$$

$$k_{24} = v_1 \oplus v_3 \oplus v_5 \oplus v_6 \oplus v_9 \oplus v_{10} \oplus v_{11} \oplus v_{12} \oplus v_{13} \oplus v_{14} \oplus v_{16} \oplus v_{19} \oplus v_{20}$$

$$k_{25} = v_0 \oplus v_1 \oplus v_2 \oplus v_4 \oplus v_5 \oplus v_7 \oplus v_8 \oplus v_{10} \oplus v_{12} \oplus v_{13} \oplus v_{14} \oplus v_{15} \oplus v_{17} \oplus v_{18} \oplus v_{20} \oplus v_{21}$$

$$k_{26} = v_0 \oplus v_2 \oplus v_3 \oplus v_9 \oplus v_{13} \oplus v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{19} \oplus v_{21}$$

$$k_{27} = v_0 \oplus v_3 \oplus v_4 \oplus v_5 \oplus v_6 \oplus v_8 \oplus v_{10} \oplus v_{11} \oplus v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{18} \oplus v_{20}$$

$$k_{28} = v_1 \oplus v_4 \oplus v_5 \oplus v_6 \oplus v_7 \oplus v_9 \oplus v_{11} \oplus v_{12} \oplus v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{18} \oplus v_{19} \oplus v_{21}$$

$$k_{29} = v_0 \oplus v_1 \oplus v_2 \oplus v_7 \oplus v_{10} \oplus v_{11} \oplus v_{12} \oplus v_{13} \oplus v_{16} \oplus v_{17} \oplus v_{19} \oplus v_{20}$$

$$k_{30} = v_0 \oplus v_2 \oplus v_3 \oplus v_5 \oplus v_6 \oplus v_{12} \oplus v_{13} \oplus v_{14} \oplus v_{17} \oplus v_{20} \oplus v_{21}$$

$$k_{31} = v_1 \oplus v_3 \oplus v_4 \oplus v_6 \oplus v_7 \oplus v_{13} \oplus v_{14} \oplus v_{15} \oplus v_{18} \oplus v_{21}$$

$$k_{32} = v_2 \oplus v_4 \oplus v_5 \oplus v_7 \oplus v_8 \oplus v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{19}$$

$$k_{33} = v_3 \oplus v_5 \oplus v_6 \oplus v_8 \oplus v_9 \oplus v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{20}$$

$$k_{34} = v_4 \oplus v_6 \oplus v_7 \oplus v_9 \oplus v_{10} \oplus v_{16} \oplus v_{17} \oplus v_{18} \oplus v_{21}$$

$$k_{35} = v_5 \oplus v_7 \oplus v_8 \oplus v_{10} \oplus v_{11} \oplus v_{17} \oplus v_{18} \oplus v_{19}$$

$$k_{36} = v_6 \oplus v_8 \oplus v_9 \oplus v_{11} \oplus v_{12} \oplus v_{18} \oplus v_{19} \oplus v_{20}$$

$$k_{37} = v_0 \oplus v_1 \oplus v_5 \oplus v_6 \oplus v_7 \oplus v_8 \oplus v_9 \oplus v_{10} \oplus v_{11} \oplus v_{12} \oplus v_{13} \oplus v_{18} \oplus v_{19} \oplus v_{20} \oplus v_{21}$$

$$k_{38} = v_1 \oplus v_2 \oplus v_6 \oplus v_7 \oplus v_8 \oplus v_9 \oplus v_{10} \oplus v_{11} \oplus v_{12} \oplus v_{13} \oplus v_{14} \oplus v_{19} \oplus v_{20} \oplus v_{21}$$

$$k_{39} = v_2 \oplus v_3 \oplus v_7 \oplus v_8 \oplus v_9 \oplus v_{10} \oplus v_{11} \oplus v_{12} \oplus v_{13} \oplus v_{14} \oplus v_{15} \oplus v_{20} \oplus v_{21}$$

$$k_{40} = v_0 \oplus v_1 \oplus v_3 \oplus v_4 \oplus v_5 \oplus v_6 \oplus v_9 \oplus v_{10} \oplus v_{12} \oplus v_{13} \oplus v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{18} \oplus v_{21}$$

$$k_{41} = v_0 \oplus v_2 \oplus v_4 \oplus v_7 \oplus v_8 \oplus v_{10} \oplus v_{13} \oplus v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{18} \oplus v_{19}$$

$$k_{42} = v_1 \oplus v_3 \oplus v_5 \oplus v_8 \oplus v_9 \oplus v_{11} \oplus v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{18} \oplus v_{19} \oplus v_{20}$$

$$k_{43} = v_0 \oplus v_1 \oplus v_2 \oplus v_4 \oplus v_5 \oplus v_8 \oplus v_9 \oplus v_{10} \oplus v_{11} \oplus v_{12} \oplus v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{19} \oplus v_{20} \oplus v_{21}$$

$$k_{44} = v_0 \oplus v_2 \oplus v_3 \oplus v_8 \oplus v_9 \oplus v_{10} \oplus v_{12} \oplus v_{13} \oplus v_{16} \oplus v_{17} \oplus v_{20} \oplus v_{21}$$

$$
\begin{aligned}
k_{45} =& v_0 \oplus v_3 \oplus v_4 \oplus v_5 \oplus v_6 \oplus v_8 \oplus v_9 \oplus v_{10} \oplus v_{13} \oplus v_{14} \oplus v_{17} \oplus v_{21}\\
k_{46} =& v_0 \oplus v_4 \oplus v_7 \oplus v_8 \oplus v_9 \oplus v_{10} \oplus v_{14} \oplus v_{15}\\
k_{47} =& v_1 \oplus v_5 \oplus v_8 \oplus v_9 \oplus v_{10} \oplus v_{11} \oplus v_{15} \oplus v_{16}\\
k_{48} =& v_2 \oplus v_6 \oplus v_9 \oplus v_{10} \oplus v_{11} \oplus v_{12} \oplus v_{16} \oplus v_{17}\\
k_{49} =& v_0 \oplus v_1 \oplus v_3 \oplus v_5 \oplus v_6 \oplus v_7 \oplus v_8 \oplus v_{10} \oplus v_{12} \oplus v_{13} \oplus v_{17}\\
k_{50} =& v_1 \oplus v_2 \oplus v_4 \oplus v_6 \oplus v_7 \oplus v_8 \oplus v_9 \oplus v_{11} \oplus v_{13} \oplus v_{14} \oplus v_{18}\\
k_{51} =& v_2 \oplus v_3 \oplus v_5 \oplus v_7 \oplus v_8 \oplus v_9 \oplus v_{10} \oplus v_{12} \oplus v_{14} \oplus v_{15} \oplus v_{19}\\
k_{52} =& v_3 \oplus v_4 \oplus v_6 \oplus v_8 \oplus v_9 \oplus v_{10} \oplus v_{11} \oplus v_{13} \oplus v_{15} \oplus v_{16} \oplus v_{20}\\
k_{53} =& v_4 \oplus v_5 \oplus v_7 \oplus v_9 \oplus v_{10} \oplus v_{11} \oplus v_{12} \oplus v_{14} \oplus v_{16} \oplus v_{17} \oplus v_{21}\\
k_{54} =& v_5 \oplus v_6 \oplus v_8 \oplus v_{10} \oplus v_{11} \oplus v_{12} \oplus v_{13} \oplus v_{15} \oplus v_{17} \oplus v_{18}\\
k_{55} =& v_6 \oplus v_7 \oplus v_9 \oplus v_{11} \oplus v_{12} \oplus v_{13} \oplus v_{14} \oplus v_{16} \oplus v_{18} \oplus v_{19}\\
k_{56} =& v_0 \oplus v_1 \oplus v_5 \oplus v_6 \oplus v_7 \oplus v_{10} \oplus v_{11} \oplus v_{12} \oplus v_{13} \oplus v_{14} \oplus v_{15} \oplus v_{17} \oplus v_{18} \oplus\\
& v_{19} \oplus v_{20}\\
k_{57} =& v_1 \oplus v_2 \oplus v_6 \oplus v_7 \oplus v_8 \oplus v_{11} \oplus v_{12} \oplus v_{13} \oplus v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{18} \oplus v_{19} \oplus\\
& v_{20} \oplus v_{21}\\
k_{58} =& v_2 \oplus v_3 \oplus v_7 \oplus v_8 \oplus v_9 \oplus v_{12} \oplus v_{13} \oplus v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{19} \oplus v_{20} \oplus\\
& v_{21}\\
k_{59} =& v_3 \oplus v_4 \oplus v_8 \oplus v_9 \oplus v_{10} \oplus v_{13} \oplus v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{18} \oplus v_{20} \oplus v_{21}\\
k_{60} =& v_0 \oplus v_1 \oplus v_4 \oplus v_6 \oplus v_8 \oplus v_9 \oplus v_{10} \oplus v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{19} \oplus v_{21}\\
k_{61} =& v_0 \oplus v_2 \oplus v_6 \oplus v_7 \oplus v_8 \oplus v_9 \oplus v_{10} \oplus v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{20}\\
k_{62} =& v_0 \oplus v_3 \oplus v_5 \oplus v_6 \oplus v_7 \oplus v_9 \oplus v_{10} \oplus v_{16} \oplus v_{17} \oplus v_{21}\\
k_{63} =& v_0 \oplus v_4 \oplus v_5 \oplus v_7 \oplus v_{10} \oplus v_{17}
\end{aligned}
$$

Second scenario: the relationship between key and IV bits

Case 1: to freeze two registers A and B, the required conditions to the system of equations that result in these two registers A and B containing all-zero values at the end of loading phase can be described as follows:

$$\begin{aligned}
k_0 =& k_{41} \oplus k_{44} \oplus k_{46} \oplus k_{50} \oplus k_{51} \oplus k_{52} \oplus k_{53} \oplus k_{54} \oplus k_{56} \oplus k_{57} \oplus k_{59} \oplus k_{60} \oplus k_{62} \oplus v_0 \oplus \\
& v_1 \oplus v_2 \oplus v_3 \oplus v_4 \oplus v_8 \oplus v_9 \oplus v_{10} \oplus v_{11} \oplus v_{14} \oplus v_{15} \oplus v_{18} \oplus v_{19} \oplus v_{20} \\
k_1 =& k_{42} \oplus k_{45} \oplus k_{47} \oplus k_{51} \oplus k_{52} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{57} \oplus k_{58} \oplus k_{60} \oplus k_{61} \oplus k_{63} \oplus v_1 \oplus \\
& v_2 \oplus v_3 \oplus v_4 \oplus v_5 \oplus v_9 \oplus v_{10} \oplus v_{11} \oplus v_{12} \oplus v_{15} \oplus v_{16} \oplus v_{19} \oplus v_{20} \oplus v_{21} \\
k_2 =& k_{43} \oplus k_{46} \oplus k_{48} \oplus k_{52} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{58} \oplus k_{59} \oplus k_{61} \oplus k_{62} \oplus v_0 \oplus v_2 \oplus \\
& v_3 \oplus v_4 \oplus v_5 \oplus v_6 \oplus v_{10} \oplus v_{11} \oplus v_{12} \oplus v_{13} \oplus v_{16} \oplus v_{17} \oplus v_{20} \oplus v_{21} \\
k_3 =& k_{44} \oplus k_{47} \oplus k_{49} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{57} \oplus k_{59} \oplus k_{60} \oplus k_{62} \oplus k_{63} \oplus v_1 \oplus v_3 \oplus \\
& v_4 \oplus v_5 \oplus v_6 \oplus v_7 \oplus v_{11} \oplus v_{12} \oplus v_{13} \oplus v_{14} \oplus v_{17} \oplus v_{18} \oplus v_{21} \\
k_4 =& k_{45} \oplus k_{48} \oplus k_{50} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{60} \oplus k_{61} \oplus k_{63} \oplus v_0 \oplus v_2 \oplus v_4 \oplus \\
& v_5 \oplus v_6 \oplus v_7 \oplus v_8 \oplus v_{12} \oplus v_{13} \oplus v_{14} \oplus v_{15} \oplus v_{18} \oplus v_{19} \\
k_5 =& k_{46} \oplus k_{49} \oplus k_{51} \oplus k_{55} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{59} \oplus k_{61} \oplus k_{62} \oplus v_0 \oplus v_1 \oplus v_3 \oplus v_5 \oplus \\
& v_6 \oplus v_7 \oplus v_8 \oplus v_9 \oplus v_{13} \oplus v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{19} \oplus v_{20} \\
k_6 =& k_{47} \oplus k_{50} \oplus k_{52} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus k_{62} \oplus k_{63} \oplus v_1 \oplus v_2 \oplus v_4 \oplus v_6 \oplus \\
& v_7 \oplus v_8 \oplus v_9 \oplus v_{10} \oplus v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{20} \oplus v_{21} \\
k_7 =& k_{48} \oplus k_{51} \oplus k_{53} \oplus k_{57} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus k_{61} \oplus k_{63} \oplus v_0 \oplus v_2 \oplus v_3 \oplus v_5 \oplus v_7 \oplus \\
& v_8 \oplus v_9 \oplus v_{10} \oplus v_{11} \oplus v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{18} \oplus v_{21} \\
k_8 =& k_{49} \oplus k_{52} \oplus k_{54} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus k_{61} \oplus k_{62} \oplus v_0 \oplus v_1 \oplus v_3 \oplus v_4 \oplus v_6 \oplus v_8 \oplus \\
& v_9 \oplus v_{10} \oplus v_{11} \oplus v_{12} \oplus v_{16} \oplus v_{17} \oplus v_{18} \oplus v_{19} \\
k_9 =& k_{50} \oplus k_{53} \oplus k_{55} \oplus k_{59} \oplus k_{60} \oplus k_{61} \oplus k_{62} \oplus k_{63} \oplus v_1 \oplus v_2 \oplus v_4 \oplus v_5 \oplus v_7 \oplus v_9 \oplus \\
& v_{10} \oplus v_{11} \oplus v_{12} \oplus v_{13} \oplus v_{17} \oplus v_{18} \oplus v_{19} \oplus v_{20} \\
k_{10} =& k_{51} \oplus k_{54} \oplus k_{56} \oplus k_{60} \oplus k_{61} \oplus k_{62} \oplus k_{63} \oplus v_0 \oplus v_2 \oplus v_3 \oplus v_5 \oplus v_6 \oplus v_8 \oplus v_{10} \oplus \\
& v_{11} \oplus v_{12} \oplus v_{13} \oplus v_{14} \oplus v_{18} \oplus v_{19} \oplus v_{20} \oplus v_{21} \\
k_{11} =& k_{52} \oplus k_{55} \oplus k_{57} \oplus k_{61} \oplus k_{62} \oplus k_{63} \oplus v_0 \oplus v_1 \oplus v_3 \oplus v_4 \oplus v_6 \oplus v_7 \oplus v_9 \oplus v_{11} \oplus \\
& v_{12} \oplus v_{13} \oplus v_{14} \oplus v_{15} \oplus v_{19} \oplus v_{20} \oplus v_{21} \\
k_{12} =& k_{53} \oplus k_{56} \oplus k_{58} \oplus k_{62} \oplus k_{63} \oplus v_0 \oplus v_1 \oplus v_2 \oplus v_4 \oplus v_5 \oplus v_7 \oplus v_8 \oplus v_{10} \oplus v_{12} \oplus \\
& v_{13} \oplus v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{20} \oplus v_{21}
\end{aligned}$$

$$\begin{aligned}
k_{13} =& k_{54} \oplus k_{57} \oplus k_{59} \oplus k_{63} \oplus v_0 \oplus v_1 \oplus v_2 \oplus v_3 \oplus v_5 \oplus v_6 \oplus v_8 \oplus v_9 \oplus v_{11} \oplus v_{13} \oplus v_{14} \oplus \\
& v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{21} \\
k_{14} =& k_{41} \oplus k_{44} \oplus k_{46} \oplus k_{50} \oplus k_{51} \oplus k_{52} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{59} \oplus \\
& k_{62} \oplus v_6 \oplus v_7 \oplus v_8 \oplus v_{11} \oplus v_{12} \oplus v_{16} \oplus v_{17} \oplus v_{19} \oplus v_{20} \\
k_{15} =& k_{42} \oplus k_{45} \oplus k_{47} \oplus k_{51} \oplus k_{52} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus \\
& k_{63} \oplus v_7 \oplus v_8 \oplus v_9 \oplus v_{12} \oplus v_{13} \oplus v_{17} \oplus v_{18} \oplus v_{20} \oplus v_{21} \\
k_{16} =& k_{43} \oplus k_{46} \oplus k_{48} \oplus k_{52} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus k_{61} \oplus v_0 \oplus \\
& v_8 \oplus v_9 \oplus v_{10} \oplus v_{13} \oplus v_{14} \oplus v_{18} \oplus v_{19} \oplus v_{21} \\
k_{17} =& k_{41} \oplus k_{46} \oplus k_{47} \oplus k_{49} \oplus k_{50} \oplus k_{51} \oplus k_{52} \oplus k_{55} \oplus k_{58} \oplus k_{61} \oplus v_0 \oplus v_2 \oplus v_3 \oplus v_4 \oplus \\
& v_8 \oplus v_{18} \\
k_{18} =& k_{41} \oplus k_{42} \oplus k_{44} \oplus k_{46} \oplus k_{47} \oplus k_{48} \oplus k_{54} \oplus k_{57} \oplus k_{60} \oplus v_0 \oplus v_2 \oplus v_5 \oplus v_8 \oplus v_{10} \oplus \\
& v_{11} \oplus v_{14} \oplus v_{15} \oplus v_{18} \oplus v_{20} \\
k_{19} =& k_{41} \oplus k_{42} \oplus k_{43} \oplus k_{44} \oplus k_{45} \oplus k_{46} \oplus k_{47} \oplus k_{48} \oplus k_{49} \oplus k_{50} \oplus k_{51} \oplus k_{52} \oplus k_{53} \oplus \\
& k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus k_{61} \oplus k_{62} \oplus v_0 \oplus v_2 \oplus v_4 \oplus v_6 \oplus v_8 \oplus \\
& v_{10} \oplus v_{12} \oplus v_{14} \oplus v_{16} \oplus v_{18} \oplus v_{20} \oplus v_{21} \\
k_{20} =& k_{42} \oplus k_{43} \oplus k_{44} \oplus k_{45} \oplus k_{46} \oplus k_{47} \oplus k_{48} \oplus k_{49} \oplus k_{50} \oplus k_{51} \oplus k_{52} \oplus k_{53} \oplus k_{54} \oplus \\
& k_{55} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus k_{61} \oplus k_{62} \oplus k_{63} \oplus v_1 \oplus v_3 \oplus v_5 \oplus v_7 \oplus v_9 \oplus \\
& v_{11} \oplus v_{13} \oplus v_{15} \oplus v_{17} \oplus v_{19} \oplus v_{21} \\
k_{21} =& k_{41} \oplus k_{43} \oplus k_{45} \oplus k_{47} \oplus k_{48} \oplus k_{49} \oplus k_{55} \oplus k_{58} \oplus k_{61} \oplus k_{63} \oplus v_1 \oplus v_3 \oplus v_6 \oplus v_9 \oplus \\
& v_{11} \oplus v_{12} \oplus v_{15} \oplus v_{16} \oplus v_{19} \\
k_{22} =& k_{41} \oplus k_{42} \oplus k_{48} \oplus k_{49} \oplus k_{51} \oplus k_{52} \oplus k_{53} \oplus k_{54} \oplus k_{57} \oplus k_{60} \oplus v_1 \oplus v_3 \oplus v_7 \oplus v_8 \oplus \\
& v_9 \oplus v_{11} \oplus v_{12} \oplus v_{13} \oplus v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{18} \oplus v_{19} \\
k_{23} =& k_{42} \oplus k_{43} \oplus k_{49} \oplus k_{50} \oplus k_{52} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{58} \oplus k_{61} \oplus v_2 \oplus v_4 \oplus v_8 \oplus v_9 \oplus \\
& v_{10} \oplus v_{12} \oplus v_{13} \oplus v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{18} \oplus v_{19} \oplus v_{20} \\
k_{24} =& k_{43} \oplus k_{44} \oplus k_{50} \oplus k_{51} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{59} \oplus k_{62} \oplus v_3 \oplus v_5 \oplus v_9 \oplus v_{10} \oplus \\
& v_{11} \oplus v_{13} \oplus v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{18} \oplus v_{19} \oplus v_{20} \oplus v_{21} \\
k_{25} =& k_{44} \oplus k_{45} \oplus k_{51} \oplus k_{52} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{57} \oplus k_{60} \oplus k_{63} \oplus v_4 \oplus v_6 \oplus v_{10} \oplus v_{11} \oplus \\
& v_{12} \oplus v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{18} \oplus v_{19} \oplus v_{20} \oplus v_{21}
\end{aligned}$$

$$
\begin{aligned}
k_{26} =& k_{45} \oplus k_{46} \oplus k_{52} \oplus k_{53} \oplus k_{55} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{61} \oplus v_0 \oplus v_5 \oplus v_7 \oplus v_{11} \oplus v_{12} \oplus \\
& v_{13} \oplus v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{18} \oplus v_{19} \oplus v_{20} \oplus v_{21} \\
k_{27} =& k_{46} \oplus k_{47} \oplus k_{53} \oplus k_{54} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{59} \oplus k_{62} \oplus v_1 \oplus v_6 \oplus v_8 \oplus v_{12} \oplus v_{13} \oplus \\
& v_{14} \oplus v_{16} \oplus v_{17} \oplus v_{18} \oplus v_{19} \oplus v_{20} \oplus v_{21} \\
k_{28} =& k_{47} \oplus k_{48} \oplus k_{54} \oplus k_{55} \oplus k_{57} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus k_{63} \oplus v_2 \oplus v_7 \oplus v_9 \oplus v_{13} \oplus v_{14} \oplus \\
& v_{15} \oplus v_{17} \oplus v_{18} \oplus v_{19} \oplus v_{20} \oplus v_{21} \\
k_{29} =& k_{48} \oplus k_{49} \oplus k_{55} \oplus k_{56} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus k_{61} \oplus v_0 \oplus v_3 \oplus v_8 \oplus v_{10} \oplus v_{14} \oplus v_{15} \oplus \\
& v_{16} \oplus v_{18} \oplus v_{19} \oplus v_{20} \oplus v_{21} \\
k_{30} =& k_{49} \oplus k_{50} \oplus k_{56} \oplus k_{57} \oplus k_{59} \oplus k_{60} \oplus k_{61} \oplus k_{62} \oplus v_1 \oplus v_4 \oplus v_9 \oplus v_{11} \oplus v_{15} \oplus v_{16} \oplus \\
& v_{17} \oplus v_{19} \oplus v_{20} \oplus v_{21} \\
k_{31} =& k_{50} \oplus k_{51} \oplus k_{57} \oplus k_{58} \oplus k_{60} \oplus k_{61} \oplus k_{62} \oplus k_{63} \oplus v_2 \oplus v_5 \oplus v_{10} \oplus v_{12} \oplus v_{16} \oplus v_{17} \oplus \\
& v_{18} \oplus v_{20} \oplus v_{21} \\
k_{32} =& k_{51} \oplus k_{52} \oplus k_{58} \oplus k_{59} \oplus k_{61} \oplus k_{62} \oplus k_{63} \oplus v_0 \oplus v_3 \oplus v_6 \oplus v_{11} \oplus v_{13} \oplus v_{17} \oplus v_{18} \oplus \\
& v_{19} \oplus v_{21} \\
k_{33} =& k_{52} \oplus k_{53} \oplus k_{59} \oplus k_{60} \oplus k_{62} \oplus k_{63} \oplus v_0 \oplus v_1 \oplus v_4 \oplus v_7 \oplus v_{12} \oplus v_{14} \oplus v_{18} \oplus v_{19} \oplus v_{20} \\
k_{34} =& k_{53} \oplus k_{54} \oplus k_{60} \oplus k_{61} \oplus k_{63} \oplus v_0 \oplus v_1 \oplus v_2 \oplus v_5 \oplus v_8 \oplus v_{13} \oplus v_{15} \oplus v_{19} \oplus v_{20} \oplus v_{21} \\
k_{35} =& k_{41} \oplus k_{44} \oplus k_{46} \oplus k_{50} \oplus k_{51} \oplus k_{52} \oplus k_{53} \oplus k_{55} \oplus k_{56} \oplus k_{57} \oplus k_{59} \oplus k_{60} \oplus k_{61} \oplus \\
& v_4 \oplus v_6 \oplus v_8 \oplus v_{10} \oplus v_{11} \oplus v_{15} \oplus v_{16} \oplus v_{18} \oplus v_{19} \oplus v_{21} \\
k_{36} =& k_{41} \oplus k_{42} \oplus k_{44} \oplus k_{45} \oplus k_{46} \oplus k_{47} \oplus k_{50} \oplus k_{58} \oplus k_{59} \oplus k_{61} \oplus v_0 \oplus v_1 \oplus v_2 \oplus v_3 \oplus \\
& v_4 \oplus v_5 \oplus v_7 \oplus v_8 \oplus v_{10} \oplus v_{12} \oplus v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{18} \\
k_{37} =& k_{42} \oplus k_{43} \oplus k_{45} \oplus k_{46} \oplus k_{47} \oplus k_{48} \oplus k_{51} \oplus k_{59} \oplus k_{60} \oplus k_{62} \oplus v_1 \oplus v_2 \oplus v_3 \oplus v_4 \oplus \\
& v_5 \oplus v_6 \oplus v_8 \oplus v_9 \oplus v_{11} \oplus v_{13} \oplus v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{18} \oplus v_{19} \\
k_{38} =& k_{41} \oplus k_{43} \oplus k_{47} \oplus k_{48} \oplus k_{49} \oplus k_{50} \oplus k_{51} \oplus k_{53} \oplus k_{54} \oplus k_{56} \oplus k_{57} \oplus k_{59} \oplus k_{61} \oplus \\
& k_{62} \oplus k_{63} \oplus v_0 \oplus v_1 \oplus v_5 \oplus v_6 \oplus v_7 \oplus v_8 \oplus v_{11} \oplus v_{12} \oplus v_{15} \oplus v_{16} \oplus v_{17} \\
k_{39} =& k_{42} \oplus k_{44} \oplus k_{48} \oplus k_{49} \oplus k_{50} \oplus k_{51} \oplus k_{52} \oplus k_{54} \oplus k_{55} \oplus k_{57} \oplus k_{58} \oplus k_{60} \oplus k_{62} \oplus \\
& k_{63} \oplus v_0 \oplus v_1 \oplus v_2 \oplus v_6 \oplus v_7 \oplus v_8 \oplus v_9 \oplus v_{12} \oplus v_{13} \oplus v_{16} \oplus v_{17} \oplus v_{18} \\
k_{40} =& k_{43} \oplus k_{45} \oplus k_{49} \oplus k_{50} \oplus k_{51} \oplus k_{52} \oplus k_{53} \oplus k_{55} \oplus k_{56} \oplus k_{58} \oplus k_{59} \oplus k_{61} \oplus k_{63} \oplus \\
& v_0 \oplus v_1 \oplus v_2 \oplus v_3 \oplus v_7 \oplus v_8 \oplus v_9 \oplus v_{10} \oplus v_{13} \oplus v_{14} \oplus v_{17} \oplus v_{18} \oplus v_{19}
\end{aligned}
$$

Case 2: to freeze two registers A and C, the conditions for the system of equations that are required to result in these two registers A and C containing all-zero values after completing loading phase can be expressed as follows:

$$\begin{aligned}
k_0 =&k_{42} \oplus k_{44} \oplus k_{47} \oplus k_{49} \oplus k_{50} \oplus k_{51} \oplus k_{53} \oplus k_{54} \oplus k_{60} \oplus k_{61} \oplus v_0 \oplus v_4 \oplus v_5 \oplus v_7 \oplus \\
&v_{12} \oplus v_{14} \oplus v_{17} \oplus v_{19} \oplus v_{20} \oplus v_{21} \\
k_1 =&k_{43} \oplus k_{45} \oplus k_{48} \oplus k_{50} \oplus k_{51} \oplus k_{52} \oplus k_{54} \oplus k_{55} \oplus k_{61} \oplus k_{62} \oplus v_1 \oplus v_5 \oplus v_6 \oplus v_8 \oplus \\
&v_{13} \oplus v_{15} \oplus v_{18} \oplus v_{20} \oplus v_{21} \\
k_2 =&k_{44} \oplus k_{46} \oplus k_{49} \oplus k_{51} \oplus k_{52} \oplus k_{53} \oplus k_{55} \oplus k_{56} \oplus k_{62} \oplus k_{63} \oplus v_2 \oplus v_6 \oplus v_7 \oplus v_9 \oplus \\
&v_{14} \oplus v_{16} \oplus v_{19} \oplus v_{21} \\
k_3 =&k_{45} \oplus k_{47} \oplus k_{50} \oplus k_{52} \oplus k_{53} \oplus k_{54} \oplus k_{56} \oplus k_{57} \oplus k_{63} \oplus v_0 \oplus v_3 \oplus v_7 \oplus v_8 \oplus v_{10} \oplus \\
&v_{15} \oplus v_{17} \oplus v_{20} \\
k_4 =&k_{46} \oplus k_{48} \oplus k_{51} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{57} \oplus k_{58} \oplus v_0 \oplus v_1 \oplus v_4 \oplus v_8 \oplus v_9 \oplus v_{11} \oplus \\
&v_{16} \oplus v_{18} \oplus v_{21} \\
k_5 =&k_{47} \oplus k_{49} \oplus k_{52} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{58} \oplus k_{59} \oplus v_1 \oplus v_2 \oplus v_5 \oplus v_9 \oplus v_{10} \oplus v_{12} \oplus \\
&v_{17} \oplus v_{19} \\
k_6 =&k_{48} \oplus k_{50} \oplus k_{53} \oplus k_{55} \oplus k_{56} \oplus k_{57} \oplus k_{59} \oplus k_{60} \oplus v_2 \oplus v_3 \oplus v_6 \oplus v_{10} \oplus v_{11} \oplus v_{13} \oplus \\
&v_{18} \oplus v_{20} \\
k_7 =&k_{49} \oplus k_{51} \oplus k_{54} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{60} \oplus k_{61} \oplus v_3 \oplus v_4 \oplus v_7 \oplus v_{11} \oplus v_{12} \oplus v_{14} \oplus \\
&v_{19} \oplus v_{21} \\
k_8 =&k_{42} \oplus k_{44} \oplus k_{47} \oplus k_{49} \oplus k_{51} \oplus k_{52} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{57} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus \\
&k_{62} \oplus v_0 \oplus v_7 \oplus v_8 \oplus v_{13} \oplus v_{14} \oplus v_{15} \oplus v_{17} \oplus v_{19} \oplus v_{21} \\
k_9 =&k_{43} \oplus k_{45} \oplus k_{48} \oplus k_{50} \oplus k_{52} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus k_{61} \oplus \\
&k_{63} \oplus v_1 \oplus v_8 \oplus v_9 \oplus v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{18} \oplus v_{20} \\
k_{10} =&k_{44} \oplus k_{46} \oplus k_{49} \oplus k_{51} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{57} \oplus k_{59} \oplus k_{60} \oplus k_{61} \oplus k_{62} \oplus \\
&v_0 \oplus v_2 \oplus v_9 \oplus v_{10} \oplus v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{19} \oplus v_{21} \\
k_{11} =&k_{45} \oplus k_{47} \oplus k_{50} \oplus k_{52} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{60} \oplus k_{61} \oplus k_{62} \oplus k_{63} \oplus \\
&v_1 \oplus v_3 \oplus v_{10} \oplus v_{11} \oplus v_{16} \oplus v_{17} \oplus v_{18} \oplus v_{20} \\
k_{12} =&k_{46} \oplus k_{48} \oplus k_{51} \oplus k_{53} \oplus k_{55} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{59} \oplus k_{61} \oplus k_{62} \oplus k_{63} \oplus v_0 \oplus \\
&v_2 \oplus v_4 \oplus v_{11} \oplus v_{12} \oplus v_{17} \oplus v_{18} \oplus v_{19} \oplus v_{21} \\
k_{13} =&k_{47} \oplus k_{49} \oplus k_{52} \oplus k_{54} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus k_{62} \oplus k_{63} \oplus v_0 \oplus v_1 \oplus \\
&v_3 \oplus v_5 \oplus v_{12} \oplus v_{13} \oplus v_{18} \oplus v_{19} \oplus v_{20}
\end{aligned}$$

$$\begin{aligned}
k_{14} =&k_{42} \oplus k_{44} \oplus k_{47} \oplus k_{48} \oplus k_{49} \oplus k_{51} \oplus k_{54} \oplus k_{55} \oplus k_{57} \oplus k_{58} \oplus k_{59} \oplus k_{63} \oplus v_1 \oplus \\
&v_2 \oplus v_5 \oplus v_6 \oplus v_7 \oplus v_{12} \oplus v_{13} \oplus v_{17} \\
k_{15} =&k_{43} \oplus k_{45} \oplus k_{48} \oplus k_{49} \oplus k_{50} \oplus k_{52} \oplus k_{55} \oplus k_{56} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus v_0 \oplus v_2 \oplus v_3 \oplus \\
&v_6 \oplus v_7 \oplus v_8 \oplus v_{13} \oplus v_{14} \oplus v_{18} \\
k_{16} =&k_{44} \oplus k_{46} \oplus k_{49} \oplus k_{50} \oplus k_{51} \oplus k_{53} \oplus k_{56} \oplus k_{57} \oplus k_{59} \oplus k_{60} \oplus k_{61} \oplus v_1 \oplus v_3 \oplus v_4 \oplus \\
&v_7 \oplus v_8 \oplus v_9 \oplus v_{14} \oplus v_{15} \oplus v_{19} \\
k_{17} =&k_{42} \oplus k_{44} \oplus k_{45} \oplus k_{49} \oplus k_{52} \oplus k_{53} \oplus k_{57} \oplus k_{58} \oplus k_{62} \oplus v_0 \oplus v_2 \oplus v_7 \oplus v_8 \oplus v_9 \oplus \\
&v_{10} \oplus v_{12} \oplus v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{19} \oplus v_{21} \\
k_{18} =&k_{42} \oplus k_{43} \oplus k_{44} \oplus k_{45} \oplus k_{46} \oplus k_{47} \oplus k_{49} \oplus k_{51} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus k_{61} \oplus k_{63} \oplus \\
&v_0 \oplus v_1 \oplus v_3 \oplus v_4 \oplus v_5 \oplus v_7 \oplus v_8 \oplus v_9 \oplus v_{10} \oplus v_{11} \oplus v_{12} \oplus v_{13} \oplus v_{14} \oplus v_{15} \oplus v_{16} \oplus \\
&v_{18} \oplus v_{19} \oplus v_{21} \\
k_{19} =&k_{42} \oplus k_{43} \oplus k_{45} \oplus k_{46} \oplus k_{48} \oplus k_{49} \oplus k_{51} \oplus k_{52} \oplus k_{53} \oplus k_{54} \oplus k_{59} \oplus k_{62} \oplus v_1 \oplus v_2 \oplus \\
&v_6 \oplus v_7 \oplus v_8 \oplus v_9 \oplus v_{10} \oplus v_{11} \oplus v_{13} \oplus v_{15} \oplus v_{16} \oplus v_{21} \\
k_{20} =&k_{43} \oplus k_{44} \oplus k_{46} \oplus k_{47} \oplus k_{49} \oplus k_{50} \oplus k_{52} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{60} \oplus k_{63} \oplus v_2 \oplus v_3 \oplus \\
&v_7 \oplus v_8 \oplus v_9 \oplus v_{10} \oplus v_{11} \oplus v_{12} \oplus v_{14} \oplus v_{16} \oplus v_{17} \\
k_{21} =&k_{42} \oplus k_{45} \oplus k_{48} \oplus k_{49} \oplus k_{55} \oplus k_{56} \oplus k_{60} \oplus v_3 \oplus v_5 \oplus v_7 \oplus v_8 \oplus v_9 \oplus v_{10} \oplus v_{11} \oplus \\
&v_{13} \oplus v_{14} \oplus v_{15} \oplus v_{18} \oplus v_{19} \oplus v_{20} \oplus v_{21} \\
k_{22} =&k_{43} \oplus k_{46} \oplus k_{49} \oplus k_{50} \oplus k_{56} \oplus k_{57} \oplus k_{61} \oplus v_4 \oplus v_6 \oplus v_8 \oplus v_9 \oplus v_{10} \oplus v_{11} \oplus v_{12} \oplus \\
&v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{19} \oplus v_{20} \oplus v_{21} \\
k_{23} =&k_{42} \oplus k_{49} \oplus k_{53} \oplus k_{54} \oplus k_{57} \oplus k_{58} \oplus k_{60} \oplus k_{61} \oplus k_{62} \oplus v_0 \oplus v_4 \oplus v_9 \oplus v_{10} \oplus v_{11} \oplus \\
&v_{13} \oplus v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{19} \\
k_{24} =&k_{43} \oplus k_{50} \oplus k_{54} \oplus k_{55} \oplus k_{58} \oplus k_{59} \oplus k_{61} \oplus k_{62} \oplus k_{63} \oplus v_1 \oplus v_5 \oplus v_{10} \oplus v_{11} \oplus v_{12} \oplus \\
&v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{20} \\
k_{25} =&k_{42} \oplus k_{47} \oplus k_{49} \oplus k_{50} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{59} \oplus k_{61} \oplus k_{62} \oplus k_{63} \oplus v_2 \oplus v_4 \oplus \\
&v_5 \oplus v_6 \oplus v_7 \oplus v_{11} \oplus v_{13} \oplus v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{18} \oplus v_{19} \oplus v_{20} \\
k_{26} =&k_{42} \oplus k_{43} \oplus k_{44} \oplus k_{47} \oplus k_{48} \oplus k_{49} \oplus k_{53} \oplus k_{55} \oplus k_{56} \oplus k_{57} \oplus k_{61} \oplus k_{62} \oplus k_{63} \oplus v_3 \oplus \\
&v_4 \oplus v_6 \oplus v_8 \oplus v_{15} \oplus v_{16} \\
k_{27} =&k_{42} \oplus k_{43} \oplus k_{45} \oplus k_{47} \oplus k_{48} \oplus k_{51} \oplus k_{53} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{60} \oplus k_{61} \oplus k_{62} \oplus k_{63} \oplus \\
&v_9 \oplus v_{12} \oplus v_{14} \oplus v_{16} \oplus v_{19} \oplus v_{20} \oplus v_{21}
\end{aligned}$$

$$k_{28} = k_{43} \oplus k_{44} \oplus k_{46} \oplus k_{48} \oplus k_{49} \oplus k_{52} \oplus k_{54} \oplus k_{57} \oplus k_{58} \oplus k_{59} \oplus k_{61} \oplus k_{62} \oplus k_{63} \oplus v_0 \oplus v_{10} \oplus v_{13} \oplus v_{15} \oplus v_{17} \oplus v_{20} \oplus v_{21}$$

$$k_{29} = k_{44} \oplus k_{45} \oplus k_{47} \oplus k_{49} \oplus k_{50} \oplus k_{53} \oplus k_{55} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus k_{62} \oplus k_{63} \oplus v_0 \oplus v_1 \oplus v_{11} \oplus v_{14} \oplus v_{16} \oplus v_{18} \oplus v_{21}$$

$$k_{30} = k_{45} \oplus k_{46} \oplus k_{48} \oplus k_{50} \oplus k_{51} \oplus k_{54} \oplus k_{56} \oplus k_{59} \oplus k_{60} \oplus k_{61} \oplus k_{63} \oplus v_0 \oplus v_1 \oplus v_2 \oplus v_{12} \oplus v_{15} \oplus v_{17} \oplus v_{19}$$

$$k_{31} = k_{46} \oplus k_{47} \oplus k_{49} \oplus k_{51} \oplus k_{52} \oplus k_{55} \oplus k_{57} \oplus k_{60} \oplus k_{61} \oplus k_{62} \oplus v_0 \oplus v_1 \oplus v_2 \oplus v_3 \oplus v_{13} \oplus v_{16} \oplus v_{18} \oplus v_{20}$$

$$k_{32} = k_{47} \oplus k_{48} \oplus k_{50} \oplus k_{52} \oplus k_{53} \oplus k_{56} \oplus k_{58} \oplus k_{61} \oplus k_{62} \oplus k_{63} \oplus v_1 \oplus v_2 \oplus v_3 \oplus v_4 \oplus v_{14} \oplus v_{17} \oplus v_{19} \oplus v_{21}$$

$$k_{33} = k_{48} \oplus k_{49} \oplus k_{51} \oplus k_{53} \oplus k_{54} \oplus k_{57} \oplus k_{59} \oplus k_{62} \oplus k_{63} \oplus v_0 \oplus v_2 \oplus v_3 \oplus v_4 \oplus v_5 \oplus v_{15} \oplus v_{18} \oplus v_{20}$$

$$k_{34} = k_{49} \oplus k_{50} \oplus k_{52} \oplus k_{54} \oplus k_{55} \oplus k_{58} \oplus k_{60} \oplus k_{63} \oplus v_0 \oplus v_1 \oplus v_3 \oplus v_4 \oplus v_5 \oplus v_6 \oplus v_{16} \oplus v_{19} \oplus v_{21}$$

$$k_{35} = k_{42} \oplus k_{44} \oplus k_{47} \oplus k_{49} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{59} \oplus k_{60} \oplus v_1 \oplus v_2 \oplus v_6 \oplus v_{12} \oplus v_{14} \oplus v_{19} \oplus v_{21}$$

$$k_{36} = k_{42} \oplus k_{43} \oplus k_{44} \oplus k_{45} \oplus k_{47} \oplus k_{48} \oplus k_{49} \oplus k_{51} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{57} \oplus v_0 \oplus v_2 \oplus v_3 \oplus v_4 \oplus v_5 \oplus v_{12} \oplus v_{13} \oplus v_{14} \oplus v_{15} \oplus v_{17} \oplus v_{19} \oplus v_{21}$$

$$k_{37} = k_{42} \oplus k_{43} \oplus k_{45} \oplus k_{46} \oplus k_{47} \oplus k_{48} \oplus k_{51} \oplus k_{52} \oplus k_{53} \oplus k_{55} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{60} \oplus k_{61} \oplus v_0 \oplus v_1 \oplus v_3 \oplus v_6 \oplus v_7 \oplus v_{12} \oplus v_{13} \oplus v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{18} \oplus v_{19} \oplus v_{21}$$

$$k_{38} = k_{42} \oplus k_{43} \oplus k_{46} \oplus k_{48} \oplus k_{50} \oplus k_{51} \oplus k_{52} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus k_{62} \oplus v_0 \oplus v_1 \oplus v_2 \oplus v_5 \oplus v_8 \oplus v_{12} \oplus v_{13} \oplus v_{16} \oplus v_{18} \oplus v_{21}$$

$$k_{39} = k_{43} \oplus k_{44} \oplus k_{47} \oplus k_{49} \oplus k_{51} \oplus k_{52} \oplus k_{53} \oplus k_{57} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus k_{61} \oplus k_{63} \oplus v_1 \oplus v_2 \oplus v_3 \oplus v_6 \oplus v_9 \oplus v_{13} \oplus v_{14} \oplus v_{17} \oplus v_{19}$$

$$k_{40} = k_{42} \oplus k_{45} \oplus k_{47} \oplus k_{48} \oplus k_{49} \oplus k_{51} \oplus k_{52} \oplus k_{58} \oplus k_{59} \oplus k_{62} \oplus v_2 \oplus v_3 \oplus v_5 \oplus v_{10} \oplus v_{12} \oplus v_{15} \oplus v_{17} \oplus v_{18} \oplus v_{19} \oplus v_{21}$$

$$k_{41} = k_{43} \oplus k_{46} \oplus k_{48} \oplus k_{49} \oplus k_{50} \oplus k_{52} \oplus k_{53} \oplus k_{59} \oplus k_{60} \oplus k_{63} \oplus v_3 \oplus v_4 \oplus v_6 \oplus v_{11} \oplus v_{13} \oplus v_{16} \oplus v_{18} \oplus v_{19} \oplus v_{20}$$

Case 3: to freeze two registers B and C, the system of equations that is required to result in these two registers B and C containing all-zero values requires some constraints as follow:

$$\begin{aligned}
k_0 =& k_{45} \oplus k_{48} \oplus k_{51} \oplus k_{54} \oplus k_{57} \oplus k_{61} \oplus k_{63} \oplus v_5 \oplus v_{13} \oplus v_{14} \oplus v_{16} \oplus v_{17} \oplus v_{18} \\
k_1 =& k_{46} \oplus k_{49} \oplus k_{52} \oplus k_{55} \oplus k_{58} \oplus k_{62} \oplus v_0 \oplus v_6 \oplus v_{14} \oplus v_{15} \oplus v_{17} \oplus v_{18} \oplus v_{19} \\
k_2 =& k_{47} \oplus k_{50} \oplus k_{53} \oplus k_{56} \oplus k_{59} \oplus k_{63} \oplus v_1 \oplus v_7 \oplus v_{15} \oplus v_{16} \oplus v_{18} \oplus v_{19} \oplus v_{20} \\
k_3 =& k_{48} \oplus k_{51} \oplus k_{54} \oplus k_{57} \oplus k_{60} \oplus v_0 \oplus v_2 \oplus v_8 \oplus v_{16} \oplus v_{17} \oplus v_{19} \oplus v_{20} \oplus v_{21} \\
k_4 =& k_{49} \oplus k_{52} \oplus k_{55} \oplus k_{58} \oplus k_{61} \oplus v_1 \oplus v_3 \oplus v_9 \oplus v_{17} \oplus v_{18} \oplus v_{20} \oplus v_{21} \\
k_5 =& k_{50} \oplus k_{53} \oplus k_{56} \oplus k_{59} \oplus k_{62} \oplus v_2 \oplus v_4 \oplus v_{10} \oplus v_{18} \oplus v_{19} \oplus v_{21} \\
k_6 =& k_{51} \oplus k_{54} \oplus k_{57} \oplus k_{60} \oplus k_{63} \oplus v_3 \oplus v_5 \oplus v_{11} \oplus v_{19} \oplus v_{20} \\
k_7 =& k_{52} \oplus k_{55} \oplus k_{58} \oplus k_{61} \oplus v_0 \oplus v_4 \oplus v_6 \oplus v_{12} \oplus v_{20} \oplus v_{21} \\
k_8 =& k_{45} \oplus k_{48} \oplus k_{51} \oplus k_{53} \oplus k_{54} \oplus k_{56} \oplus k_{57} \oplus k_{59} \oplus k_{61} \oplus k_{62} \oplus k_{63} \oplus v_1 \oplus v_7 \oplus v_{14} \oplus \\
& v_{16} \oplus v_{17} \oplus v_{18} \oplus v_{21} \\
k_9 =& k_{46} \oplus k_{49} \oplus k_{52} \oplus k_{54} \oplus k_{55} \oplus k_{57} \oplus k_{58} \oplus k_{60} \oplus k_{62} \oplus k_{63} \oplus v_0 \oplus v_2 \oplus v_8 \oplus v_{15} \oplus \\
& v_{17} \oplus v_{18} \oplus v_{19} \\
k_{10} =& k_{47} \oplus k_{50} \oplus k_{53} \oplus k_{55} \oplus k_{56} \oplus k_{58} \oplus k_{59} \oplus k_{61} \oplus k_{63} \oplus v_0 \oplus v_1 \oplus v_3 \oplus v_9 \oplus v_{16} \oplus \\
& v_{18} \oplus v_{19} \oplus v_{20} \\
k_{11} =& k_{48} \oplus k_{51} \oplus k_{54} \oplus k_{56} \oplus k_{57} \oplus k_{59} \oplus k_{60} \oplus k_{62} \oplus v_0 \oplus v_1 \oplus v_2 \oplus v_4 \oplus v_{10} \oplus v_{17} \oplus \\
& v_{19} \oplus v_{20} \oplus v_{21} \\
k_{12} =& k_{49} \oplus k_{52} \oplus k_{55} \oplus k_{57} \oplus k_{58} \oplus k_{60} \oplus k_{61} \oplus k_{63} \oplus v_1 \oplus v_2 \oplus v_3 \oplus v_5 \oplus v_{11} \oplus v_{18} \oplus \\
& v_{20} \oplus v_{21} \\
k_{13} =& k_{50} \oplus k_{53} \oplus k_{56} \oplus k_{58} \oplus k_{59} \oplus k_{61} \oplus k_{62} \oplus v_0 \oplus v_2 \oplus v_3 \oplus v_4 \oplus v_6 \oplus v_{12} \oplus v_{19} \oplus v_{21} \\
k_{14} =& k_{51} \oplus k_{54} \oplus k_{57} \oplus k_{59} \oplus k_{60} \oplus k_{62} \oplus k_{63} \oplus v_1 \oplus v_3 \oplus v_4 \oplus v_5 \oplus v_7 \oplus v_{13} \oplus v_{20} \\
k_{15} =& k_{52} \oplus k_{55} \oplus k_{58} \oplus k_{60} \oplus k_{61} \oplus k_{63} \oplus v_0 \oplus v_2 \oplus v_4 \oplus v_5 \oplus v_6 \oplus v_8 \oplus v_{14} \oplus v_{21} \\
k_{16} =& k_{53} \oplus k_{56} \oplus k_{59} \oplus k_{61} \oplus k_{62} \oplus v_0 \oplus v_1 \oplus v_3 \oplus v_5 \oplus v_6 \oplus v_7 \oplus v_9 \oplus v_{15} \\
k_{17} =& k_{54} \oplus k_{57} \oplus k_{60} \oplus k_{62} \oplus k_{63} \oplus v_1 \oplus v_2 \oplus v_4 \oplus v_6 \oplus v_7 \oplus v_8 \oplus v_{10} \oplus v_{16} \\
k_{18} =& k_{55} \oplus k_{58} \oplus k_{61} \oplus k_{63} \oplus v_0 \oplus v_2 \oplus v_3 \oplus v_5 \oplus v_7 \oplus v_8 \oplus v_9 \oplus v_{11} \oplus v_{17} \\
k_{19} =& k_{56} \oplus k_{59} \oplus k_{62} \oplus v_0 \oplus v_1 \oplus v_3 \oplus v_4 \oplus v_6 \oplus v_8 \oplus v_9 \oplus v_{10} \oplus v_{12} \oplus v_{18} \\
k_{20} =& k_{57} \oplus k_{60} \oplus k_{63} \oplus v_1 \oplus v_2 \oplus v_4 \oplus v_5 \oplus v_7 \oplus v_9 \oplus v_{10} \oplus v_{11} \oplus v_{13} \oplus v_{19}
\end{aligned}$$

$$k_{21} = k_{58} \oplus k_{61} \oplus v_0 \oplus v_2 \oplus v_3 \oplus v_5 \oplus v_6 \oplus v_8 \oplus v_{10} \oplus v_{11} \oplus v_{12} \oplus v_{14} \oplus v_{20}$$

$$k_{22} = k_{59} \oplus k_{62} \oplus v_1 \oplus v_3 \oplus v_4 \oplus v_6 \oplus v_7 \oplus v_9 \oplus v_{11} \oplus v_{12} \oplus v_{13} \oplus v_{15} \oplus v_{21}$$

$$k_{23} = k_{45} \oplus k_{48} \oplus k_{51} \oplus k_{54} \oplus k_{57} \oplus k_{60} \oplus k_{61} \oplus v_2 \oplus v_4 \oplus v_7 \oplus v_8 \oplus v_{10} \oplus v_{12} \oplus v_{17} \oplus v_{18}$$

$$k_{24} = k_{46} \oplus k_{49} \oplus k_{52} \oplus k_{55} \oplus k_{58} \oplus k_{61} \oplus k_{62} \oplus v_3 \oplus v_5 \oplus v_8 \oplus v_9 \oplus v_{11} \oplus v_{13} \oplus v_{18} \oplus v_{19}$$

$$k_{25} = k_{47} \oplus k_{50} \oplus k_{53} \oplus k_{56} \oplus k_{59} \oplus k_{62} \oplus k_{63} \oplus v_4 \oplus v_6 \oplus v_9 \oplus v_{10} \oplus v_{12} \oplus v_{14} \oplus v_{19} \oplus v_{20}$$

$$k_{26} = k_{48} \oplus k_{51} \oplus k_{54} \oplus k_{57} \oplus k_{60} \oplus k_{63} \oplus v_0 \oplus v_5 \oplus v_7 \oplus v_{10} \oplus v_{11} \oplus v_{13} \oplus v_{15} \oplus v_{20} \oplus v_{21}$$

$$k_{27} = k_{49} \oplus k_{52} \oplus k_{55} \oplus k_{58} \oplus k_{61} \oplus v_0 \oplus v_1 \oplus v_6 \oplus v_8 \oplus v_{11} \oplus v_{12} \oplus v_{14} \oplus v_{16} \oplus v_{21}$$

$$k_{28} = k_{50} \oplus k_{53} \oplus k_{56} \oplus k_{59} \oplus k_{62} \oplus v_1 \oplus v_2 \oplus v_7 \oplus v_9 \oplus v_{12} \oplus v_{13} \oplus v_{15} \oplus v_{17}$$

$$k_{29} = k_{45} \oplus k_{48} \oplus k_{60} \oplus k_{61} \oplus v_2 \oplus v_3 \oplus v_5 \oplus v_8 \oplus v_{10} \oplus v_{17}$$

$$\begin{aligned} k_{30} =& k_{45} \oplus k_{46} \oplus k_{48} \oplus k_{49} \oplus k_{51} \oplus k_{54} \oplus k_{57} \oplus k_{62} \oplus k_{63} \oplus v_3 \oplus v_4 \oplus v_5 \oplus v_6 \oplus v_9 \oplus \\ & v_{11} \oplus v_{13} \oplus v_{14} \oplus v_{16} \oplus v_{17} \end{aligned}$$

$$\begin{aligned} k_{31} =& k_{46} \oplus k_{47} \oplus k_{49} \oplus k_{50} \oplus k_{52} \oplus k_{55} \oplus k_{58} \oplus k_{63} \oplus v_0 \oplus v_4 \oplus v_5 \oplus v_6 \oplus v_7 \oplus v_{10} \oplus \\ & v_{12} \oplus v_{14} \oplus v_{15} \oplus v_{17} \oplus v_{18} \end{aligned}$$

$$\begin{aligned} k_{32} =& k_{47} \oplus k_{48} \oplus k_{50} \oplus k_{51} \oplus k_{53} \oplus k_{56} \oplus k_{59} \oplus v_0 \oplus v_1 \oplus v_5 \oplus v_6 \oplus v_7 \oplus v_8 \oplus v_{11} \oplus \\ & v_{13} \oplus v_{15} \oplus v_{16} \oplus v_{18} \oplus v_{19} \end{aligned}$$

$$\begin{aligned} k_{33} =& k_{48} \oplus k_{49} \oplus k_{51} \oplus k_{52} \oplus k_{54} \oplus k_{57} \oplus k_{60} \oplus v_1 \oplus v_2 \oplus v_6 \oplus v_7 \oplus v_8 \oplus v_9 \oplus v_{12} \oplus \\ & v_{14} \oplus v_{16} \oplus v_{17} \oplus v_{19} \oplus v_{20} \end{aligned}$$

$$\begin{aligned} k_{34} =& k_{49} \oplus k_{50} \oplus k_{52} \oplus k_{53} \oplus k_{55} \oplus k_{58} \oplus k_{61} \oplus v_2 \oplus v_3 \oplus v_7 \oplus v_8 \oplus v_9 \oplus v_{10} \oplus v_{13} \oplus \\ & v_{15} \oplus v_{17} \oplus v_{18} \oplus v_{20} \oplus v_{21} \end{aligned}$$

$$\begin{aligned} k_{35} =& k_{50} \oplus k_{51} \oplus k_{53} \oplus k_{54} \oplus k_{56} \oplus k_{59} \oplus k_{62} \oplus v_3 \oplus v_4 \oplus v_8 \oplus v_9 \oplus v_{10} \oplus v_{11} \oplus v_{14} \oplus \\ & v_{16} \oplus v_{18} \oplus v_{19} \oplus v_{21} \end{aligned}$$

$$\begin{aligned} k_{36} =& k_{51} \oplus k_{52} \oplus k_{54} \oplus k_{55} \oplus k_{57} \oplus k_{60} \oplus k_{63} \oplus v_4 \oplus v_5 \oplus v_9 \oplus v_{10} \oplus v_{11} \oplus v_{12} \oplus v_{15} \oplus \\ & v_{17} \oplus v_{19} \oplus v_{20} \end{aligned}$$

$$\begin{aligned} k_{37} =& k_{52} \oplus k_{53} \oplus k_{55} \oplus k_{56} \oplus k_{58} \oplus k_{61} \oplus v_0 \oplus v_5 \oplus v_6 \oplus v_{10} \oplus v_{11} \oplus v_{12} \oplus v_{13} \oplus v_{16} \oplus \\ & v_{18} \oplus v_{20} \oplus v_{21} \end{aligned}$$

$$\begin{aligned} k_{38} =& k_{53} \oplus k_{54} \oplus k_{56} \oplus k_{57} \oplus k_{59} \oplus k_{62} \oplus v_1 \oplus v_6 \oplus v_7 \oplus v_{11} \oplus v_{12} \oplus v_{13} \oplus v_{14} \oplus v_{17} \oplus \\ & v_{19} \oplus v_{21} \end{aligned}$$

$$k_{39} = k_{54} \oplus k_{55} \oplus k_{57} \oplus k_{58} \oplus k_{60} \oplus k_{63} \oplus v_2 \oplus v_7 \oplus v_8 \oplus v_{12} \oplus v_{13} \oplus v_{14} \oplus v_{15} \oplus v_{18} \oplus v_{20}$$

$$k_{40} = k_{55} \oplus k_{56} \oplus k_{58} \oplus k_{59} \oplus k_{61} \oplus v_0 \oplus v_3 \oplus v_8 \oplus v_9 \oplus v_{13} \oplus v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{19} \oplus v_{21}$$

$$k_{41} = k_{56} \oplus k_{57} \oplus k_{59} \oplus k_{60} \oplus k_{62} \oplus v_1 \oplus v_4 \oplus v_9 \oplus v_{10} \oplus v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{20}$$

$$k_{42} = k_{45} \oplus k_{48} \oplus k_{51} \oplus k_{54} \oplus k_{58} \oplus k_{60} \oplus v_2 \oplus v_{10} \oplus v_{11} \oplus v_{13} \oplus v_{14} \oplus v_{15} \oplus v_{21}$$

$$k_{43} = k_{46} \oplus k_{49} \oplus k_{52} \oplus k_{55} \oplus k_{59} \oplus k_{61} \oplus v_3 \oplus v_{11} \oplus v_{12} \oplus v_{14} \oplus v_{15} \oplus v_{16}$$

$$k_{44} = k_{47} \oplus k_{50} \oplus k_{53} \oplus k_{56} \oplus k_{60} \oplus k_{62} \oplus v_4 \oplus v_{12} \oplus v_{13} \oplus v_{15} \oplus v_{16} \oplus v_{17}$$

Third scenario: the relationship between key and IV bits

Case 4: to freeze register A, there are some constraints for the system of equations that are required to result in register A containing all-zero values at the end of the loading phase as follows:

$$\begin{aligned}
k_0 =& k_{19} \oplus k_{20} \oplus k_{22} \oplus k_{23} \oplus k_{24} \oplus k_{25} \oplus k_{28} \oplus k_{33} \oplus k_{34} \oplus k_{36} \oplus k_{37} \oplus k_{39} \oplus k_{40} \oplus k_{42} \oplus \\
& k_{44} \oplus k_{46} \oplus k_{47} \oplus k_{51} \oplus k_{52} \oplus k_{53} \oplus k_{57} \oplus k_{58} \oplus k_{59} \oplus k_{62} \oplus k_{63} \oplus v_0 \oplus v_1 \oplus v_2 \oplus \\
& v_3 \oplus v_4 \oplus v_5 \oplus v_7 \oplus v_8 \oplus v_9 \oplus v_{10} \oplus v_{12} \oplus v_{13} \oplus v_{19}
\end{aligned}$$

$$\begin{aligned}
k_1 =& k_{20} \oplus k_{21} \oplus k_{23} \oplus k_{24} \oplus k_{25} \oplus k_{26} \oplus k_{29} \oplus k_{34} \oplus k_{35} \oplus k_{37} \oplus k_{38} \oplus k_{40} \oplus k_{41} \oplus k_{43} \oplus \\
& k_{45} \oplus k_{47} \oplus k_{48} \oplus k_{52} \oplus k_{53} \oplus k_{54} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus k_{63} \oplus v_0 \oplus v_1 \oplus v_2 \oplus v_3 \oplus v_4 \oplus \\
& v_5 \oplus v_6 \oplus v_8 \oplus v_9 \oplus v_{10} \oplus v_{11} \oplus v_{13} \oplus v_{14} \oplus v_{20}
\end{aligned}$$

$$\begin{aligned}
k_2 =& k_{21} \oplus k_{22} \oplus k_{24} \oplus k_{25} \oplus k_{26} \oplus k_{27} \oplus k_{30} \oplus k_{35} \oplus k_{36} \oplus k_{38} \oplus k_{39} \oplus k_{41} \oplus k_{42} \oplus k_{44} \oplus \\
& k_{46} \oplus k_{48} \oplus k_{49} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{59} \oplus k_{60} \oplus k_{61} \oplus v_0 \oplus v_1 \oplus v_2 \oplus v_3 \oplus v_4 \oplus v_5 \oplus \\
& v_6 \oplus v_7 \oplus v_9 \oplus v_{10} \oplus v_{11} \oplus v_{12} \oplus v_{14} \oplus v_{15} \oplus v_{21}
\end{aligned}$$

$$\begin{aligned}
k_3 =& k_{22} \oplus k_{23} \oplus k_{25} \oplus k_{26} \oplus k_{27} \oplus k_{28} \oplus k_{31} \oplus k_{36} \oplus k_{37} \oplus k_{39} \oplus k_{40} \oplus k_{42} \oplus k_{43} \oplus k_{45} \oplus \\
& k_{47} \oplus k_{49} \oplus k_{50} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{60} \oplus k_{61} \oplus k_{62} \oplus v_1 \oplus v_2 \oplus v_3 \oplus v_4 \oplus v_5 \oplus v_6 \oplus \\
& v_7 \oplus v_8 \oplus v_{10} \oplus v_{11} \oplus v_{12} \oplus v_{13} \oplus v_{15} \oplus v_{16}
\end{aligned}$$

$$\begin{aligned}
k_4 =& k_{23} \oplus k_{24} \oplus k_{26} \oplus k_{27} \oplus k_{28} \oplus k_{29} \oplus k_{32} \oplus k_{37} \oplus k_{38} \oplus k_{40} \oplus k_{41} \oplus k_{43} \oplus k_{44} \oplus k_{46} \oplus \\
& k_{48} \oplus k_{50} \oplus k_{51} \oplus k_{55} \oplus k_{56} \oplus k_{57} \oplus k_{61} \oplus k_{62} \oplus k_{63} \oplus v_2 \oplus v_3 \oplus v_4 \oplus v_5 \oplus v_6 \oplus v_7 \oplus \\
& v_8 \oplus v_9 \oplus v_{11} \oplus v_{12} \oplus v_{13} \oplus v_{14} \oplus v_{16} \oplus v_{17}
\end{aligned}$$

$$\begin{aligned}
k_5 =& k_{24} \oplus k_{25} \oplus k_{27} \oplus k_{28} \oplus k_{29} \oplus k_{30} \oplus k_{33} \oplus k_{38} \oplus k_{39} \oplus k_{41} \oplus k_{42} \oplus k_{44} \oplus k_{45} \oplus k_{47} \oplus \\
& k_{49} \oplus k_{51} \oplus k_{52} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{62} \oplus k_{63} \oplus v_0 \oplus v_3 \oplus v_4 \oplus v_5 \oplus v_6 \oplus v_7 \oplus v_8 \oplus \\
& v_9 \oplus v_{10} \oplus v_{12} \oplus v_{13} \oplus v_{14} \oplus v_{15} \oplus v_{17} \oplus v_{18}
\end{aligned}$$

$$\begin{aligned}
k_6 =& k_{25} \oplus k_{26} \oplus k_{28} \oplus k_{29} \oplus k_{30} \oplus k_{31} \oplus k_{34} \oplus k_{39} \oplus k_{40} \oplus k_{42} \oplus k_{43} \oplus k_{45} \oplus k_{46} \oplus k_{48} \oplus \\
& k_{50} \oplus k_{52} \oplus k_{53} \oplus k_{57} \oplus k_{58} \oplus k_{59} \oplus k_{63} \oplus v_0 \oplus v_1 \oplus v_4 \oplus v_5 \oplus v_6 \oplus v_7 \oplus v_8 \oplus v_9 \oplus \\
& v_{10} \oplus v_{11} \oplus v_{13} \oplus v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{18} \oplus v_{19}
\end{aligned}$$

$$\begin{aligned}
k_7 =& k_{26} \oplus k_{27} \oplus k_{29} \oplus k_{30} \oplus k_{31} \oplus k_{32} \oplus k_{35} \oplus k_{40} \oplus k_{41} \oplus k_{43} \oplus k_{44} \oplus k_{46} \oplus k_{47} \oplus k_{49} \oplus \\
& k_{51} \oplus k_{53} \oplus k_{54} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus v_0 \oplus v_1 \oplus v_2 \oplus v_5 \oplus v_6 \oplus v_7 \oplus v_8 \oplus v_9 \oplus v_{10} \oplus \\
& v_{11} \oplus v_{12} \oplus v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{19} \oplus v_{20}
\end{aligned}$$

$$\begin{aligned}
k_8 =& k_{27} \oplus k_{28} \oplus k_{30} \oplus k_{31} \oplus k_{32} \oplus k_{33} \oplus k_{36} \oplus k_{41} \oplus k_{42} \oplus k_{44} \oplus k_{45} \oplus k_{47} \oplus k_{48} \oplus k_{50} \oplus \\
& k_{52} \oplus k_{54} \oplus k_{55} \oplus k_{59} \oplus k_{60} \oplus k_{61} \oplus v_1 \oplus v_2 \oplus v_3 \oplus v_6 \oplus v_7 \oplus v_8 \oplus v_9 \oplus v_{10} \oplus v_{11} \oplus \\
& v_{12} \oplus v_{13} \oplus v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{18} \oplus v_{20} \oplus v_{21}
\end{aligned}$$

$$\begin{aligned}
k_9 =& k_{28} \oplus k_{29} \oplus k_{31} \oplus k_{32} \oplus k_{33} \oplus k_{34} \oplus k_{37} \oplus k_{42} \oplus k_{43} \oplus k_{45} \oplus k_{46} \oplus k_{48} \oplus k_{49} \oplus k_{51} \oplus \\
& k_{53} \oplus k_{55} \oplus k_{56} \oplus k_{60} \oplus k_{61} \oplus k_{62} \oplus v_2 \oplus v_3 \oplus v_4 \oplus v_7 \oplus v_8 \oplus v_9 \oplus v_{10} \oplus v_{11} \oplus v_{12} \oplus \\
& v_{13} \oplus v_{14} \oplus v_{16} \oplus v_{17} \oplus v_{18} \oplus v_{19} \oplus v_{21} \\
k_{10} =& k_{29} \oplus k_{30} \oplus k_{32} \oplus k_{33} \oplus k_{34} \oplus k_{35} \oplus k_{38} \oplus k_{43} \oplus k_{44} \oplus k_{46} \oplus k_{47} \oplus k_{49} \oplus k_{50} \oplus \\
& k_{52} \oplus k_{54} \oplus k_{56} \oplus k_{57} \oplus k_{61} \oplus k_{62} \oplus k_{63} \oplus v_3 \oplus v_4 \oplus v_5 \oplus v_8 \oplus v_9 \oplus v_{10} \oplus v_{11} \oplus \\
& v_{12} \oplus v_{13} \oplus v_{14} \oplus v_{15} \oplus v_{17} \oplus v_{18} \oplus v_{19} \oplus v_{20} \\
k_{11} =& k_{30} \oplus k_{31} \oplus k_{33} \oplus k_{34} \oplus k_{35} \oplus k_{36} \oplus k_{39} \oplus k_{44} \oplus k_{45} \oplus k_{47} \oplus k_{48} \oplus k_{50} \oplus k_{51} \oplus \\
& k_{53} \oplus k_{55} \oplus k_{57} \oplus k_{58} \oplus k_{62} \oplus k_{63} \oplus v_0 \oplus v_4 \oplus v_5 \oplus v_6 \oplus v_9 \oplus v_{10} \oplus v_{11} \oplus v_{12} \oplus \\
& v_{13} \oplus v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{18} \oplus v_{19} \oplus v_{20} \oplus v_{21} \\
k_{12} =& k_{31} \oplus k_{32} \oplus k_{34} \oplus k_{35} \oplus k_{36} \oplus k_{37} \oplus k_{40} \oplus k_{45} \oplus k_{46} \oplus k_{48} \oplus k_{49} \oplus k_{51} \oplus k_{52} \oplus \\
& k_{54} \oplus k_{56} \oplus k_{58} \oplus k_{59} \oplus k_{63} \oplus v_0 \oplus v_1 \oplus v_5 \oplus v_6 \oplus v_7 \oplus v_{10} \oplus v_{11} \oplus v_{12} \oplus v_{13} \oplus \\
& v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{19} \oplus v_{20} \oplus v_{21} \\
k_{13} =& k_{32} \oplus k_{33} \oplus k_{35} \oplus k_{36} \oplus k_{37} \oplus k_{38} \oplus k_{41} \oplus k_{46} \oplus k_{47} \oplus k_{49} \oplus k_{50} \oplus k_{52} \oplus k_{53} \oplus \\
& k_{55} \oplus k_{57} \oplus k_{59} \oplus k_{60} \oplus v_0 \oplus v_1 \oplus v_2 \oplus v_6 \oplus v_7 \oplus v_8 \oplus v_{11} \oplus v_{12} \oplus v_{13} \oplus v_{14} \oplus \\
& v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{18} \oplus v_{20} \oplus v_{21} \\
k_{14} =& k_{19} \oplus k_{20} \oplus k_{22} \oplus k_{23} \oplus k_{24} \oplus k_{25} \oplus k_{28} \oplus k_{38} \oplus k_{40} \oplus k_{44} \oplus k_{46} \oplus k_{48} \oplus k_{50} \oplus \\
& k_{52} \oplus k_{54} \oplus k_{56} \oplus k_{57} \oplus k_{59} \oplus k_{60} \oplus k_{61} \oplus k_{62} \oplus k_{63} \oplus v_0 \oplus v_4 \oplus v_5 \oplus v_{10} \oplus v_{14} \oplus \\
& v_{15} \oplus v_{16} \oplus v_{17} \oplus v_{18} \oplus v_{21} \\
k_{15} =& k_{20} \oplus k_{21} \oplus k_{23} \oplus k_{24} \oplus k_{25} \oplus k_{26} \oplus k_{29} \oplus k_{39} \oplus k_{41} \oplus k_{45} \oplus k_{47} \oplus k_{49} \oplus k_{51} \oplus \\
& k_{53} \oplus k_{55} \oplus k_{57} \oplus k_{58} \oplus k_{60} \oplus k_{61} \oplus k_{62} \oplus k_{63} \oplus v_0 \oplus v_1 \oplus v_5 \oplus v_6 \oplus v_{11} \oplus v_{15} \oplus \\
& v_{16} \oplus v_{17} \oplus v_{18} \oplus v_{19} \\
k_{16} =& k_{21} \oplus k_{22} \oplus k_{24} \oplus k_{25} \oplus k_{26} \oplus k_{27} \oplus k_{30} \oplus k_{40} \oplus k_{42} \oplus k_{46} \oplus k_{48} \oplus k_{50} \oplus k_{52} \oplus \\
& k_{54} \oplus k_{56} \oplus k_{58} \oplus k_{59} \oplus k_{61} \oplus k_{62} \oplus k_{63} \oplus v_0 \oplus v_1 \oplus v_2 \oplus v_6 \oplus v_7 \oplus v_{12} \oplus v_{16} \oplus \\
& v_{17} \oplus v_{18} \oplus v_{19} \oplus v_{20} \\
k_{17} =& k_{19} \oplus k_{20} \oplus k_{24} \oplus k_{26} \oplus k_{27} \oplus k_{31} \oplus k_{33} \oplus k_{34} \oplus k_{36} \oplus k_{37} \oplus k_{39} \oplus k_{40} \oplus k_{41} \oplus \\
& k_{42} \oplus k_{43} \oplus k_{44} \oplus k_{46} \oplus k_{49} \oplus k_{52} \oplus k_{55} \oplus k_{58} \oplus k_{60} \oplus v_4 \oplus v_5 \oplus v_9 \oplus v_{10} \oplus v_{12} \oplus \\
& v_{17} \oplus v_{18} \oplus v_{20} \oplus v_{21} \\
k_{18} =& k_{19} \oplus k_{21} \oplus k_{22} \oplus k_{23} \oplus k_{24} \oplus k_{27} \oplus k_{32} \oplus k_{33} \oplus k_{35} \oplus k_{36} \oplus k_{38} \oplus k_{39} \oplus k_{41} \oplus \\
& k_{43} \oplus k_{45} \oplus k_{46} \oplus k_{50} \oplus k_{51} \oplus k_{52} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{61} \oplus k_{62} \oplus k_{63} \oplus v_0 \oplus \\
& v_1 \oplus v_2 \oplus v_3 \oplus v_4 \oplus v_6 \oplus v_7 \oplus v_8 \oplus v_9 \oplus v_{11} \oplus v_{12} \oplus v_{18} \oplus v_{21}
\end{aligned}$$

Case 5: to freeze register B, there are some constraints for the system of equations that are required to result in register B containing all-zero values at the end of loading phase, as follows:

$$\begin{aligned}
k_0 =& k_{22} \oplus k_{23} \oplus k_{24} \oplus k_{25} \oplus k_{26} \oplus k_{27} \oplus k_{28} \oplus k_{29} \oplus k_{30} \oplus k_{31} \oplus k_{32} \oplus k_{33} \oplus k_{34} \oplus k_{35} \oplus \\
& k_{36} \oplus k_{37} \oplus k_{38} \oplus k_{39} \oplus k_{40} \oplus k_{41} \oplus k_{42} \oplus k_{43} \oplus k_{45} \oplus k_{47} \oplus k_{49} \oplus k_{51} \oplus k_{53} \oplus k_{55} \oplus \\
& k_{57} \oplus k_{59} \oplus k_{61} \oplus k_{63} \oplus v_1 \oplus v_2 \oplus v_5 \oplus v_6 \oplus v_9 \oplus v_{10} \oplus v_{13} \oplus v_{14} \oplus v_{17} \oplus v_{18} \oplus v_{21}
\end{aligned}$$

$$\begin{aligned}
k_1 =& k_{23} \oplus k_{24} \oplus k_{25} \oplus k_{26} \oplus k_{27} \oplus k_{28} \oplus k_{29} \oplus k_{30} \oplus k_{31} \oplus k_{32} \oplus k_{33} \oplus k_{34} \oplus k_{35} \oplus k_{36} \oplus \\
& k_{37} \oplus k_{38} \oplus k_{39} \oplus k_{40} \oplus k_{41} \oplus k_{42} \oplus k_{43} \oplus k_{44} \oplus k_{46} \oplus k_{48} \oplus k_{50} \oplus k_{52} \oplus k_{54} \oplus k_{56} \oplus \\
& k_{58} \oplus k_{60} \oplus k_{62} \oplus v_0 \oplus v_2 \oplus v_3 \oplus v_6 \oplus v_7 \oplus v_{10} \oplus v_{11} \oplus v_{14} \oplus v_{15} \oplus v_{18} \oplus v_{19}
\end{aligned}$$

$$\begin{aligned}
k_2 =& k_{24} \oplus k_{25} \oplus k_{26} \oplus k_{27} \oplus k_{28} \oplus k_{29} \oplus k_{30} \oplus k_{31} \oplus k_{32} \oplus k_{33} \oplus k_{34} \oplus k_{35} \oplus k_{36} \oplus k_{37} \oplus \\
& k_{38} \oplus k_{39} \oplus k_{40} \oplus k_{41} \oplus k_{42} \oplus k_{43} \oplus k_{44} \oplus k_{45} \oplus k_{47} \oplus k_{49} \oplus k_{51} \oplus k_{53} \oplus k_{55} \oplus k_{57} \oplus \\
& k_{59} \oplus k_{61} \oplus k_{63} \oplus v_1 \oplus v_3 \oplus v_4 \oplus v_7 \oplus v_8 \oplus v_{11} \oplus v_{12} \oplus v_{15} \oplus v_{16} \oplus v_{19} \oplus v_{20}
\end{aligned}$$

$$\begin{aligned}
k_3 =& k_{25} \oplus k_{26} \oplus k_{27} \oplus k_{28} \oplus k_{29} \oplus k_{30} \oplus k_{31} \oplus k_{32} \oplus k_{33} \oplus k_{34} \oplus k_{35} \oplus k_{36} \oplus k_{37} \oplus k_{38} \oplus \\
& k_{39} \oplus k_{40} \oplus k_{41} \oplus k_{42} \oplus k_{43} \oplus k_{44} \oplus k_{45} \oplus k_{46} \oplus k_{48} \oplus k_{50} \oplus k_{52} \oplus k_{54} \oplus k_{56} \oplus k_{58} \oplus \\
& k_{60} \oplus k_{62} \oplus v_0 \oplus v_2 \oplus v_4 \oplus v_5 \oplus v_8 \oplus v_9 \oplus v_{12} \oplus v_{13} \oplus v_{16} \oplus v_{17} \oplus v_{20} \oplus v_{21}
\end{aligned}$$

$$\begin{aligned}
k_4 =& k_{26} \oplus k_{27} \oplus k_{28} \oplus k_{29} \oplus k_{30} \oplus k_{31} \oplus k_{32} \oplus k_{33} \oplus k_{34} \oplus k_{35} \oplus k_{36} \oplus k_{37} \oplus k_{38} \oplus k_{39} \oplus \\
& k_{40} \oplus k_{41} \oplus k_{42} \oplus k_{43} \oplus k_{44} \oplus k_{45} \oplus k_{46} \oplus k_{47} \oplus k_{49} \oplus k_{51} \oplus k_{53} \oplus k_{55} \oplus k_{57} \oplus k_{59} \oplus \\
& k_{61} \oplus k_{63} \oplus v_1 \oplus v_3 \oplus v_5 \oplus v_6 \oplus v_9 \oplus v_{10} \oplus v_{13} \oplus v_{14} \oplus v_{17} \oplus v_{18} \oplus v_{21}
\end{aligned}$$

$$\begin{aligned}
k_5 =& k_{27} \oplus k_{28} \oplus k_{29} \oplus k_{30} \oplus k_{31} \oplus k_{32} \oplus k_{33} \oplus k_{34} \oplus k_{35} \oplus k_{36} \oplus k_{37} \oplus k_{38} \oplus k_{39} \oplus k_{40} \oplus \\
& k_{41} \oplus k_{42} \oplus k_{43} \oplus k_{44} \oplus k_{45} \oplus k_{46} \oplus k_{47} \oplus k_{48} \oplus k_{50} \oplus k_{52} \oplus k_{54} \oplus k_{56} \oplus k_{58} \oplus k_{60} \oplus \\
& k_{62} \oplus v_0 \oplus v_2 \oplus v_4 \oplus v_6 \oplus v_7 \oplus v_{10} \oplus v_{11} \oplus v_{14} \oplus v_{15} \oplus v_{18} \oplus v_{19}
\end{aligned}$$

$$\begin{aligned}
k_6 =& k_{28} \oplus k_{29} \oplus k_{30} \oplus k_{31} \oplus k_{32} \oplus k_{33} \oplus k_{34} \oplus k_{35} \oplus k_{36} \oplus k_{37} \oplus k_{38} \oplus k_{39} \oplus k_{40} \oplus k_{41} \oplus \\
& k_{42} \oplus k_{43} \oplus k_{44} \oplus k_{45} \oplus k_{46} \oplus k_{47} \oplus k_{48} \oplus k_{49} \oplus k_{51} \oplus k_{53} \oplus k_{55} \oplus k_{57} \oplus k_{59} \oplus k_{61} \oplus \\
& k_{63} \oplus v_1 \oplus v_3 \oplus v_5 \oplus v_7 \oplus v_8 \oplus v_{11} \oplus v_{12} \oplus v_{15} \oplus v_{16} \oplus v_{19} \oplus v_{20}
\end{aligned}$$

$$\begin{aligned}
k_7 =& k_{29} \oplus k_{30} \oplus k_{31} \oplus k_{32} \oplus k_{33} \oplus k_{34} \oplus k_{35} \oplus k_{36} \oplus k_{37} \oplus k_{38} \oplus k_{39} \oplus k_{40} \oplus k_{41} \oplus \\
& k_{42} \oplus k_{43} \oplus k_{44} \oplus k_{45} \oplus k_{46} \oplus k_{47} \oplus k_{48} \oplus k_{49} \oplus k_{50} \oplus k_{52} \oplus k_{54} \oplus k_{56} \oplus k_{58} \oplus \\
& k_{60} \oplus k_{62} \oplus v_0 \oplus v_2 \oplus v_4 \oplus v_6 \oplus v_8 \oplus v_9 \oplus v_{12} \oplus v_{13} \oplus v_{16} \oplus v_{17} \oplus v_{20} \oplus v_{21}
\end{aligned}$$

$$\begin{aligned}
k_8 =& k_{30} \oplus k_{31} \oplus k_{32} \oplus k_{33} \oplus k_{34} \oplus k_{35} \oplus k_{36} \oplus k_{37} \oplus k_{38} \oplus k_{39} \oplus k_{40} \oplus k_{41} \oplus k_{42} \oplus \\
& k_{43} \oplus k_{44} \oplus k_{45} \oplus k_{46} \oplus k_{47} \oplus k_{48} \oplus k_{49} \oplus k_{50} \oplus k_{51} \oplus k_{53} \oplus k_{55} \oplus k_{57} \oplus k_{59} \oplus \\
& k_{61} \oplus k_{63} \oplus v_1 \oplus v_3 \oplus v_5 \oplus v_7 \oplus v_9 \oplus v_{10} \oplus v_{13} \oplus v_{14} \oplus v_{17} \oplus v_{18} \oplus v_{21}
\end{aligned}$$

$$\begin{aligned}
k_9 =& k_{31} \oplus k_{32} \oplus k_{33} \oplus k_{34} \oplus k_{35} \oplus k_{36} \oplus k_{37} \oplus k_{38} \oplus k_{39} \oplus k_{40} \oplus k_{41} \oplus k_{42} \oplus k_{43} \oplus \\
& k_{44} \oplus k_{45} \oplus k_{46} \oplus k_{47} \oplus k_{48} \oplus k_{49} \oplus k_{50} \oplus k_{51} \oplus k_{52} \oplus k_{54} \oplus k_{56} \oplus k_{58} \oplus k_{60} \oplus \\
& k_{62} \oplus v_0 \oplus v_2 \oplus v_4 \oplus v_6 \oplus v_8 \oplus v_{10} \oplus v_{11} \oplus v_{14} \oplus v_{15} \oplus v_{18} \oplus v_{19} \\
k_{10} =& k_{32} \oplus k_{33} \oplus k_{34} \oplus k_{35} \oplus k_{36} \oplus k_{37} \oplus k_{38} \oplus k_{39} \oplus k_{40} \oplus k_{41} \oplus k_{42} \oplus k_{43} \oplus k_{44} \oplus \\
& k_{45} \oplus k_{46} \oplus k_{47} \oplus k_{48} \oplus k_{49} \oplus k_{50} \oplus k_{51} \oplus k_{52} \oplus k_{53} \oplus k_{55} \oplus k_{57} \oplus k_{59} \oplus k_{61} \oplus \\
& k_{63} \oplus v_1 \oplus v_3 \oplus v_5 \oplus v_7 \oplus v_9 \oplus v_{11} \oplus v_{12} \oplus v_{15} \oplus v_{16} \oplus v_{19} \oplus v_{20} \\
k_{11} =& k_{33} \oplus k_{34} \oplus k_{35} \oplus k_{36} \oplus k_{37} \oplus k_{38} \oplus k_{39} \oplus k_{40} \oplus k_{41} \oplus k_{42} \oplus k_{43} \oplus k_{44} \oplus k_{45} \oplus \\
& k_{46} \oplus k_{47} \oplus k_{48} \oplus k_{49} \oplus k_{50} \oplus k_{51} \oplus k_{52} \oplus k_{53} \oplus k_{54} \oplus k_{56} \oplus k_{58} \oplus k_{60} \oplus k_{62} \oplus \\
& v_0 \oplus v_2 \oplus v_4 \oplus v_6 \oplus v_8 \oplus v_{10} \oplus v_{12} \oplus v_{13} \oplus v_{16} \oplus v_{17} \oplus v_{20} \oplus v_{21} \\
k_{12} =& k_{34} \oplus k_{35} \oplus k_{36} \oplus k_{37} \oplus k_{38} \oplus k_{39} \oplus k_{40} \oplus k_{41} \oplus k_{42} \oplus k_{43} \oplus k_{44} \oplus k_{45} \oplus k_{46} \oplus \\
& k_{47} \oplus k_{48} \oplus k_{49} \oplus k_{50} \oplus k_{51} \oplus k_{52} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{57} \oplus k_{59} \oplus k_{61} \oplus k_{63} \oplus \\
& v_1 \oplus v_3 \oplus v_5 \oplus v_7 \oplus v_9 \oplus v_{11} \oplus v_{13} \oplus v_{14} \oplus v_{17} \oplus v_{18} \oplus v_{21} \\
k_{13} =& k_{35} \oplus k_{36} \oplus k_{37} \oplus k_{38} \oplus k_{39} \oplus k_{40} \oplus k_{41} \oplus k_{42} \oplus k_{43} \oplus k_{44} \oplus k_{45} \oplus k_{46} \oplus k_{47} \oplus \\
& k_{48} \oplus k_{49} \oplus k_{50} \oplus k_{51} \oplus k_{52} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{58} \oplus k_{60} \oplus k_{62} \oplus v_0 \oplus \\
& v_2 \oplus v_4 \oplus v_6 \oplus v_8 \oplus v_{10} \oplus v_{12} \oplus v_{14} \oplus v_{15} \oplus v_{18} \oplus v_{19} \\
k_{14} =& k_{36} \oplus k_{37} \oplus k_{38} \oplus k_{39} \oplus k_{40} \oplus k_{41} \oplus k_{42} \oplus k_{43} \oplus k_{44} \oplus k_{45} \oplus k_{46} \oplus k_{47} \oplus k_{48} \oplus \\
& k_{49} \oplus k_{50} \oplus k_{51} \oplus k_{52} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{57} \oplus k_{59} \oplus k_{61} \oplus k_{63} \oplus v_1 \oplus \\
& v_3 \oplus v_5 \oplus v_7 \oplus v_9 \oplus v_{11} \oplus v_{13} \oplus v_{15} \oplus v_{16} \oplus v_{19} \oplus v_{20} \\
k_{15} =& k_{37} \oplus k_{38} \oplus k_{39} \oplus k_{40} \oplus k_{41} \oplus k_{42} \oplus k_{43} \oplus k_{44} \oplus k_{45} \oplus k_{46} \oplus k_{47} \oplus k_{48} \oplus k_{49} \oplus \\
& k_{50} \oplus k_{51} \oplus k_{52} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{60} \oplus k_{62} \oplus v_0 \oplus v_2 \oplus \\
& v_4 \oplus v_6 \oplus v_8 \oplus v_{10} \oplus v_{12} \oplus v_{14} \oplus v_{16} \oplus v_{17} \oplus v_{20} \oplus v_{21} \\
k_{16} =& k_{38} \oplus k_{39} \oplus k_{40} \oplus k_{41} \oplus k_{42} \oplus k_{43} \oplus k_{44} \oplus k_{45} \oplus k_{46} \oplus k_{47} \oplus k_{48} \oplus k_{49} \oplus k_{50} \oplus \\
& k_{51} \oplus k_{52} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{59} \oplus k_{61} \oplus k_{63} \oplus v_1 \oplus v_3 \oplus v_5 \oplus \\
& v_7 \oplus v_9 \oplus v_{11} \oplus v_{13} \oplus v_{15} \oplus v_{17} \oplus v_{18} \oplus v_{21} \\
k_{17} =& k_{39} \oplus k_{40} \oplus k_{41} \oplus k_{42} \oplus k_{43} \oplus k_{44} \oplus k_{45} \oplus k_{46} \oplus k_{47} \oplus k_{48} \oplus k_{49} \oplus k_{50} \oplus k_{51} \oplus \\
& k_{52} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus k_{62} \oplus v_0 \oplus v_2 \oplus v_4 \oplus v_6 \oplus \\
& v_8 \oplus v_{10} \oplus v_{12} \oplus v_{14} \oplus v_{16} \oplus v_{18} \oplus v_{19}
\end{aligned}$$

$$\begin{aligned}
k_{18} =& k_{40} \oplus k_{41} \oplus k_{42} \oplus k_{43} \oplus k_{44} \oplus k_{45} \oplus k_{46} \oplus k_{47} \oplus k_{48} \oplus k_{49} \oplus k_{50} \oplus k_{51} \oplus k_{52} \oplus \\
& k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus k_{61} \oplus k_{63} \oplus v_1 \oplus v_3 \oplus v_5 \oplus v_7 \oplus \\
& v_9 \oplus v_{11} \oplus v_{13} \oplus v_{15} \oplus v_{17} \oplus v_{19} \oplus v_{20} \\
k_{19} =& k_{41} \oplus k_{42} \oplus k_{43} \oplus k_{44} \oplus k_{45} \oplus k_{46} \oplus k_{47} \oplus k_{48} \oplus k_{49} \oplus k_{50} \oplus k_{51} \oplus k_{52} \oplus k_{53} \oplus \\
& k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus k_{61} \oplus k_{62} \oplus v_0 \oplus v_2 \oplus v_4 \oplus v_6 \oplus v_8 \oplus \\
& v_{10} \oplus v_{12} \oplus v_{14} \oplus v_{16} \oplus v_{18} \oplus v_{20} \oplus v_{21} \\
k_{20} =& k_{42} \oplus k_{43} \oplus k_{44} \oplus k_{45} \oplus k_{46} \oplus k_{47} \oplus k_{48} \oplus k_{49} \oplus k_{50} \oplus k_{51} \oplus k_{52} \oplus k_{53} \oplus k_{54} \oplus \\
& k_{55} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus k_{61} \oplus k_{62} \oplus k_{63} \oplus v_1 \oplus v_3 \oplus v_5 \oplus v_7 \oplus v_9 \oplus \\
& v_{11} \oplus v_{13} \oplus v_{15} \oplus v_{17} \oplus v_{19} \oplus v_{21} \\
k_{21} =& k_{22} \oplus k_{23} \oplus k_{24} \oplus k_{25} \oplus k_{26} \oplus k_{27} \oplus k_{28} \oplus k_{29} \oplus k_{30} \oplus k_{31} \oplus k_{32} \oplus k_{33} \oplus k_{34} \oplus \\
& k_{35} \oplus k_{36} \oplus k_{37} \oplus k_{38} \oplus k_{39} \oplus k_{40} \oplus k_{41} \oplus k_{42} \oplus k_{44} \oplus k_{46} \oplus k_{48} \oplus k_{50} \oplus k_{52} \oplus \\
& k_{54} \oplus k_{56} \oplus k_{58} \oplus k_{60} \oplus k_{62} \oplus v_0 \oplus v_1 \oplus v_4 \oplus v_5 \oplus v_8 \oplus v_9 \oplus v_{12} \oplus v_{13} \oplus v_{16} \oplus \\
& v_{17} \oplus v_{20} \oplus v_{21}
\end{aligned}$$

Case 6: to freeze register C, there are some constraints for the system of equations that are required to result in register C containing all-zero values at the end of the loading phase as follows:

$$\begin{aligned}
k_0 =& k_{23} \oplus k_{24} \oplus k_{26} \oplus k_{27} \oplus k_{29} \oplus k_{30} \oplus k_{32} \oplus k_{33} \oplus k_{35} \oplus k_{36} \oplus k_{39} \oplus k_{40} \oplus k_{41} \oplus k_{42} \oplus \\
& k_{45} \oplus k_{47} \oplus k_{51} \oplus k_{54} \oplus k_{55} \oplus k_{58} \oplus k_{63} \oplus v_2 \oplus v_3 \oplus v_4 \oplus v_5 \oplus v_7 \oplus v_8 \oplus v_9 \oplus v_{10} \oplus \\
& v_{12} \oplus v_{14} \oplus v_{15} \oplus v_{17} \oplus v_{20} \oplus v_{21} \\
k_1 =& k_{24} \oplus k_{25} \oplus k_{27} \oplus k_{28} \oplus k_{30} \oplus k_{31} \oplus k_{33} \oplus k_{34} \oplus k_{36} \oplus k_{37} \oplus k_{40} \oplus k_{41} \oplus k_{42} \oplus k_{43} \oplus \\
& k_{46} \oplus k_{48} \oplus k_{52} \oplus k_{55} \oplus k_{56} \oplus k_{59} \oplus v_0 \oplus v_3 \oplus v_4 \oplus v_5 \oplus v_6 \oplus v_8 \oplus v_9 \oplus v_{10} \oplus v_{11} \oplus \\
& v_{13} \oplus v_{15} \oplus v_{16} \oplus v_{18} \oplus v_{21} \\
k_2 =& k_{25} \oplus k_{26} \oplus k_{28} \oplus k_{29} \oplus k_{31} \oplus k_{32} \oplus k_{34} \oplus k_{35} \oplus k_{37} \oplus k_{38} \oplus k_{41} \oplus k_{42} \oplus k_{43} \oplus k_{44} \oplus \\
& k_{47} \oplus k_{49} \oplus k_{53} \oplus k_{56} \oplus k_{57} \oplus k_{60} \oplus v_1 \oplus v_4 \oplus v_5 \oplus v_6 \oplus v_7 \oplus v_9 \oplus v_{10} \oplus v_{11} \oplus v_{12} \oplus \\
& v_{14} \oplus v_{16} \oplus v_{17} \oplus v_{19} \\
k_3 =& k_{26} \oplus k_{27} \oplus k_{29} \oplus k_{30} \oplus k_{32} \oplus k_{33} \oplus k_{35} \oplus k_{36} \oplus k_{38} \oplus k_{39} \oplus k_{42} \oplus k_{43} \oplus k_{44} \oplus k_{45} \oplus \\
& k_{48} \oplus k_{50} \oplus k_{54} \oplus k_{57} \oplus k_{58} \oplus k_{61} \oplus v_2 \oplus v_5 \oplus v_6 \oplus v_7 \oplus v_8 \oplus v_{10} \oplus v_{11} \oplus v_{12} \oplus v_{13} \oplus \\
& v_{15} \oplus v_{17} \oplus v_{18} \oplus v_{20} \\
k_4 =& k_{27} \oplus k_{28} \oplus k_{30} \oplus k_{31} \oplus k_{33} \oplus k_{34} \oplus k_{36} \oplus k_{37} \oplus k_{39} \oplus k_{40} \oplus k_{43} \oplus k_{44} \oplus k_{45} \oplus k_{46} \oplus \\
& k_{49} \oplus k_{51} \oplus k_{55} \oplus k_{58} \oplus k_{59} \oplus k_{62} \oplus v_3 \oplus v_6 \oplus v_7 \oplus v_8 \oplus v_9 \oplus v_{11} \oplus v_{12} \oplus v_{13} \oplus \\
& v_{14} \oplus v_{16} \oplus v_{18} \oplus v_{19} \oplus v_{21} \\
k_5 =& k_{28} \oplus k_{29} \oplus k_{31} \oplus k_{32} \oplus k_{34} \oplus k_{35} \oplus k_{37} \oplus k_{38} \oplus k_{40} \oplus k_{41} \oplus k_{44} \oplus k_{45} \oplus k_{46} \oplus k_{47} \oplus \\
& k_{50} \oplus k_{52} \oplus k_{56} \oplus k_{59} \oplus k_{60} \oplus k_{63} \oplus v_4 \oplus v_7 \oplus v_8 \oplus v_9 \oplus v_{10} \oplus v_{12} \oplus v_{13} \oplus v_{14} \oplus \\
& v_{15} \oplus v_{17} \oplus v_{19} \oplus v_{20} \\
k_6 =& k_{29} \oplus k_{30} \oplus k_{32} \oplus k_{33} \oplus k_{35} \oplus k_{36} \oplus k_{38} \oplus k_{39} \oplus k_{41} \oplus k_{42} \oplus k_{45} \oplus k_{46} \oplus k_{47} \oplus k_{48} \oplus \\
& k_{51} \oplus k_{53} \oplus k_{57} \oplus k_{60} \oplus k_{61} \oplus v_0 \oplus v_5 \oplus v_8 \oplus v_9 \oplus v_{10} \oplus v_{11} \oplus v_{13} \oplus v_{14} \oplus v_{15} \oplus \\
& v_{16} \oplus v_{18} \oplus v_{20} \oplus v_{21} \\
k_7 =& k_{30} \oplus k_{31} \oplus k_{33} \oplus k_{34} \oplus k_{36} \oplus k_{37} \oplus k_{39} \oplus k_{40} \oplus k_{42} \oplus k_{43} \oplus k_{46} \oplus k_{47} \oplus k_{48} \oplus k_{49} \oplus \\
& k_{52} \oplus k_{54} \oplus k_{58} \oplus k_{61} \oplus k_{62} \oplus v_1 \oplus v_6 \oplus v_9 \oplus v_{10} \oplus v_{11} \oplus v_{12} \oplus v_{14} \oplus v_{15} \oplus v_{16} \oplus \\
& v_{17} \oplus v_{19} \oplus v_{21} \\
k_8 =& k_{23} \oplus k_{24} \oplus k_{26} \oplus k_{27} \oplus k_{29} \oplus k_{30} \oplus k_{31} \oplus k_{33} \oplus k_{34} \oplus k_{36} \oplus k_{37} \oplus k_{38} \oplus k_{39} \oplus k_{42} \oplus \\
& k_{43} \oplus k_{44} \oplus k_{45} \oplus k_{48} \oplus k_{49} \oplus k_{50} \oplus k_{51} \oplus k_{53} \oplus k_{54} \oplus k_{58} \oplus k_{59} \oplus k_{62} \oplus v_3 \oplus v_4 \oplus \\
& v_5 \oplus v_8 \oplus v_9 \oplus v_{11} \oplus v_{13} \oplus v_{14} \oplus v_{16} \oplus v_{18} \oplus v_{21}
\end{aligned}$$

$$
\begin{aligned}
k_9 =& k_{24} \oplus k_{25} \oplus k_{27} \oplus k_{28} \oplus k_{30} \oplus k_{31} \oplus k_{32} \oplus k_{34} \oplus k_{35} \oplus k_{37} \oplus k_{38} \oplus k_{39} \oplus k_{40} \oplus k_{43} \oplus \\
& k_{44} \oplus k_{45} \oplus k_{46} \oplus k_{49} \oplus k_{50} \oplus k_{51} \oplus k_{52} \oplus k_{54} \oplus k_{55} \oplus k_{59} \oplus k_{60} \oplus k_{63} \oplus v_4 \oplus v_5 \oplus \\
& v_6 \oplus v_9 \oplus v_{10} \oplus v_{12} \oplus v_{14} \oplus v_{15} \oplus v_{17} \oplus v_{19} \\
k_{10} =& k_{25} \oplus k_{26} \oplus k_{28} \oplus k_{29} \oplus k_{31} \oplus k_{32} \oplus k_{33} \oplus k_{35} \oplus k_{36} \oplus k_{38} \oplus k_{39} \oplus k_{40} \oplus k_{41} \oplus k_{44} \oplus \\
& k_{45} \oplus k_{46} \oplus k_{47} \oplus k_{50} \oplus k_{51} \oplus k_{52} \oplus k_{53} \oplus k_{55} \oplus k_{56} \oplus k_{60} \oplus k_{61} \oplus v_0 \oplus v_5 \oplus v_6 \oplus \\
& v_7 \oplus v_{10} \oplus v_{11} \oplus v_{13} \oplus v_{15} \oplus v_{16} \oplus v_{18} \oplus v_{20} \\
k_{11} =& k_{26} \oplus k_{27} \oplus k_{29} \oplus k_{30} \oplus k_{32} \oplus k_{33} \oplus k_{34} \oplus k_{36} \oplus k_{37} \oplus k_{39} \oplus k_{40} \oplus k_{41} \oplus k_{42} \oplus k_{45} \oplus \\
& k_{46} \oplus k_{47} \oplus k_{48} \oplus k_{51} \oplus k_{52} \oplus k_{53} \oplus k_{54} \oplus k_{56} \oplus k_{57} \oplus k_{61} \oplus k_{62} \oplus v_1 \oplus v_6 \oplus v_7 \oplus \\
& v_8 \oplus v_{11} \oplus v_{12} \oplus v_{14} \oplus v_{16} \oplus v_{17} \oplus v_{19} \oplus v_{21} \\
k_{12} =& k_{27} \oplus k_{28} \oplus k_{30} \oplus k_{31} \oplus k_{33} \oplus k_{34} \oplus k_{35} \oplus k_{37} \oplus k_{38} \oplus k_{40} \oplus k_{41} \oplus k_{42} \oplus k_{43} \oplus k_{46} \oplus \\
& k_{47} \oplus k_{48} \oplus k_{49} \oplus k_{52} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{57} \oplus k_{58} \oplus k_{62} \oplus k_{63} \oplus v_2 \oplus v_7 \oplus v_8 \oplus \\
& v_9 \oplus v_{12} \oplus v_{13} \oplus v_{15} \oplus v_{17} \oplus v_{18} \oplus v_{20} \\
k_{13} =& k_{28} \oplus k_{29} \oplus k_{31} \oplus k_{32} \oplus k_{34} \oplus k_{35} \oplus k_{36} \oplus k_{38} \oplus k_{39} \oplus k_{41} \oplus k_{42} \oplus k_{43} \oplus k_{44} \oplus k_{47} \oplus \\
& k_{48} \oplus k_{49} \oplus k_{50} \oplus k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{58} \oplus k_{59} \oplus k_{63} \oplus v_0 \oplus v_3 \oplus v_8 \oplus v_9 \oplus \\
& v_{10} \oplus v_{13} \oplus v_{14} \oplus v_{16} \oplus v_{18} \oplus v_{19} \oplus v_{21} \\
k_{14} =& k_{29} \oplus k_{30} \oplus k_{32} \oplus k_{33} \oplus k_{35} \oplus k_{36} \oplus k_{37} \oplus k_{39} \oplus k_{40} \oplus k_{42} \oplus k_{43} \oplus k_{44} \oplus k_{45} \oplus k_{48} \oplus \\
& k_{49} \oplus k_{50} \oplus k_{51} \oplus k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{57} \oplus k_{59} \oplus k_{60} \oplus v_0 \oplus v_1 \oplus v_4 \oplus v_9 \oplus v_{10} \oplus \\
& v_{11} \oplus v_{14} \oplus v_{15} \oplus v_{17} \oplus v_{19} \oplus v_{20} \\
k_{15} =& k_{30} \oplus k_{31} \oplus k_{33} \oplus k_{34} \oplus k_{36} \oplus k_{37} \oplus k_{38} \oplus k_{40} \oplus k_{41} \oplus k_{43} \oplus k_{44} \oplus k_{45} \oplus k_{46} \oplus k_{49} \oplus \\
& k_{50} \oplus k_{51} \oplus k_{52} \oplus k_{55} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{60} \oplus k_{61} \oplus v_1 \oplus v_2 \oplus v_5 \oplus v_{10} \oplus v_{11} \oplus \\
& v_{12} \oplus v_{15} \oplus v_{16} \oplus v_{18} \oplus v_{20} \oplus v_{21} \\
k_{16} =& k_{31} \oplus k_{32} \oplus k_{34} \oplus k_{35} \oplus k_{37} \oplus k_{38} \oplus k_{39} \oplus k_{41} \oplus k_{42} \oplus k_{44} \oplus k_{45} \oplus k_{46} \oplus k_{47} \oplus k_{50} \oplus \\
& k_{51} \oplus k_{52} \oplus k_{53} \oplus k_{56} \oplus k_{57} \oplus k_{58} \oplus k_{59} \oplus k_{61} \oplus k_{62} \oplus v_2 \oplus v_3 \oplus v_6 \oplus v_{11} \oplus v_{12} \oplus \\
& v_{13} \oplus v_{16} \oplus v_{17} \oplus v_{19} \oplus v_{21} \\
k_{17} =& k_{32} \oplus k_{33} \oplus k_{35} \oplus k_{36} \oplus k_{38} \oplus k_{39} \oplus k_{40} \oplus k_{42} \oplus k_{43} \oplus k_{45} \oplus k_{46} \oplus k_{47} \oplus k_{48} \oplus k_{51} \oplus \\
& k_{52} \oplus k_{53} \oplus k_{54} \oplus k_{57} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus k_{62} \oplus k_{63} \oplus v_3 \oplus v_4 \oplus v_7 \oplus v_{12} \oplus v_{13} \oplus \\
& v_{14} \oplus v_{17} \oplus v_{18} \oplus v_{20} \\
k_{18} =& k_{33} \oplus k_{34} \oplus k_{36} \oplus k_{37} \oplus k_{39} \oplus k_{40} \oplus k_{41} \oplus k_{43} \oplus k_{44} \oplus k_{46} \oplus k_{47} \oplus k_{48} \oplus k_{49} \oplus k_{52} \oplus \\
& k_{53} \oplus k_{54} \oplus k_{55} \oplus k_{58} \oplus k_{59} \oplus k_{60} \oplus k_{61} \oplus k_{63} \oplus v_0 \oplus v_4 \oplus v_5 \oplus v_8 \oplus v_{13} \oplus v_{14} \oplus \\
& v_{15} \oplus v_{18} \oplus v_{19} \oplus v_{21}
\end{aligned}
$$

$$\begin{aligned}
k_{19} =& k_{34} \oplus k_{35} \oplus k_{37} \oplus k_{38} \oplus k_{40} \oplus k_{41} \oplus k_{42} \oplus k_{44} \oplus k_{45} \oplus k_{47} \oplus k_{48} \oplus k_{49} \oplus k_{50} \oplus k_{53} \oplus \\
& k_{54} \oplus k_{55} \oplus k_{56} \oplus k_{59} \oplus k_{60} \oplus k_{61} \oplus k_{62} \oplus v_0 \oplus v_1 \oplus v_5 \oplus v_6 \oplus v_9 \oplus v_{14} \oplus v_{15} \oplus \\
& v_{16} \oplus v_{19} \oplus v_{20} \\
k_{20} =& k_{35} \oplus k_{36} \oplus k_{38} \oplus k_{39} \oplus k_{41} \oplus k_{42} \oplus k_{43} \oplus k_{45} \oplus k_{46} \oplus k_{48} \oplus k_{49} \oplus k_{50} \oplus k_{51} \oplus k_{54} \oplus \\
& k_{55} \oplus k_{56} \oplus k_{57} \oplus k_{60} \oplus k_{61} \oplus k_{62} \oplus k_{63} \oplus v_1 \oplus v_2 \oplus v_6 \oplus v_7 \oplus v_{10} \oplus v_{15} \oplus v_{16} \oplus \\
& v_{17} \oplus v_{20} \oplus v_{21} \\
k_{21} =& k_{23} \oplus k_{24} \oplus k_{26} \oplus k_{27} \oplus k_{29} \oplus k_{30} \oplus k_{32} \oplus k_{33} \oplus k_{35} \oplus k_{37} \oplus k_{41} \oplus k_{43} \oplus k_{44} \oplus k_{45} \oplus \\
& k_{46} \oplus k_{49} \oplus k_{50} \oplus k_{52} \oplus k_{54} \oplus k_{56} \oplus k_{57} \oplus k_{61} \oplus k_{62} \oplus v_0 \oplus v_4 \oplus v_5 \oplus v_9 \oplus v_{10} \oplus \\
& v_{11} \oplus v_{12} \oplus v_{14} \oplus v_{15} \oplus v_{16} \oplus v_{18} \oplus v_{20} \\
k_{22} =& k_{23} \oplus k_{25} \oplus k_{26} \oplus k_{28} \oplus k_{29} \oplus k_{31} \oplus k_{32} \oplus k_{34} \oplus k_{35} \oplus k_{38} \oplus k_{39} \oplus k_{40} \oplus k_{41} \oplus \\
& k_{44} \oplus k_{46} \oplus k_{50} \oplus k_{53} \oplus k_{54} \oplus k_{57} \oplus k_{62} \oplus v_1 \oplus v_2 \oplus v_3 \oplus v_4 \oplus v_6 \oplus v_7 \oplus v_8 \oplus \\
& v_9 \oplus v_{11} \oplus v_{13} \oplus v_{14} \oplus v_{16} \oplus v_{19} \oplus v_{20}
\end{aligned}$$

Appendix B

Common Scrambling Algorithm Stream Cipher

B.1 The Boolean Functions of the CSA-SC

The following two tables summaries 14 Boolean functions that are used in Chapter 5 for the Common Scrambling Algorithm Stream Cipher (CSA-SC) as reported in [96, 104].

Table B.1: Algebraic Normal Forms of the 14 Boolean functions

$x_0 =$	$S_1(i_0, i_1, i_2, i_3, i_4) =$	$1 + i_4 + i_3 + i_3i_4 + i_2i_4 + i_2i_3 + i_1i_4 + i_1i_3 + i_1i_2 + i_1i_2i_4 + i_1i_2i_3 + i_0 + i_0i_3i_4 + i_0i_2 + i_0i_2i_3 + i_0i_1 + i_0i_1i_3 + i_0i_1i_3i_4 + i_0i_1i_2 + i_0i_1i_2i_3$
$x_1 =$	$S_2(i_0, i_1, i_2, i_3, i_4) =$	$i_3 + i_2i_4 + i_1 + i_1i_4 + i_1i_3i_4 + i_0i_4 + i_0i_1 + i_0i_1i_3 + i_0i_1i_2 + i_0i_1i_2i_4$
$x_2 =$	$S_3(i_0, i_1, i_2, i_3, i_4) =$	$1 + i_4 + i_3 + i_2i_4 + i_2i_3 + i_2i_3i_4 + i_1 + i_0i_2i_3 + i_0i_1i_4 + i_0i_1i_3 + i_0i_1i_3i_4 + i_0i_1i_2$
$x_3 =$	$S_4(i_0, i_1, i_2, i_3, i_4) =$	$1 + i_3 + i_2 + i_2i_4 + i_1i_3i_4 + i_1i_2i_4 + i_0i_3i_4 + i_0i_2 + i_0i_1 + i_0i_1i_3i_4 + i_0i_1i_2i_4$
$y_0 =$	$S_5(i_0, i_1, i_2, i_3, i_4) =$	$1 + i_4 + i_3 + i_2i_4 + i_2i_3 + i_2i_3i_4 + i_1 + i_1i_4 + i_1i_3 + i_1i_3i_4 + i_1i_2 + i_1i_2i_3 + i_0 + i_0i_3 + i_0i_3i_4 + i_0i_2 + i_0i_2i_4 + i_0i_2i_3 + i_0i_2i_3i_4 + i_0i_1i_4 + i_0i_1i_2 + i_0i_1i_2i_3$
$y_1 =$	$S_6(i_0, i_1, i_2, i_3, i_4) =$	$i_3 + i_3i_4 + i_2i_4 + i_1 + i_0$
$y_2 =$	$S_7(i_0, i_1, i_2, i_3, i_4) =$	$1 + i_4 + i_3i_4 + i_2 + i_2i_3i_4 + i_1 + i_1i_2i_3 + i_0 + i_0i_4 + i_0i_3 + i_0i_2i_3i_4 + i_0i_1 + i_0i_1i_4 + i_0i_1i_3i_4 + i_0i_1i_2 + i_0i_1i_2i_3$
$y_3 =$	$S_8(i_0, i_1, i_2, i_3, i_4) =$	$1 + i_3 + i_3i_4 + i_2 + i_1i_4 + i_1i_3i_4 + i_1i_2 + i_0i_4 + i_0i_3 + i_0i_2i_3i_4 + i_0i_1 + i_0i_1i_4 + i_0i_1i_3i_4 + i_0i_1i_2 + i_0i_1i_2i_3$
$z_0 =$	$S_9(i_0, i_1, i_2, i_3, i_4) =$	$1 + i_4 + i_3 + i_3i_4 + i_2i_4 + i_2i_3 + i_2i_3i_4 + i_1 + i_1i_4 + i_1i_3i_4 + i_1i_2i_4 + i_1i_2i_3 + i_0i_4 + i_0i_3 + i_0i_2 + i_0i_2i_3 + i_0i_2i_3i_4 + i_0i_1i_4 + i_0i_1i_3 + i_0i_1i_2i_4 + i_0i_1i_2i_3$
$z_1 =$	$S_{10}(i_0, i_1, i_2, i_3, i_4) =$	$i_3i_4 + i_2 + i_2i_4 + i_2i_3i_4 + i_1i_4 + i_1i_3 + i_1i_2i_4 + i_0i_4 + i_0i_2 + i_0i_2i_4 + i_0i_2i_3 + i_0i_2i_3i_4 + i_0i_1 + i_0i_1i_4 + i_0i_1i_3 + i_0i_1i_3i_4$
$z_2 =$	$S_{11}(i_0, i_1, i_2, i_3, i_4) =$	$i_3 + i_2i_4 + i_1i_3i_4 + i_1i_2 + i_1i_2i_4 + i_0 + i_0i_3i_4 + i_0i_1i_4$
$z_3 =$	$S_{12}(i_0, i_1, i_2, i_3, i_4) =$	$i_4 + i_2 + i_2i_3 + i_2i_3i_4 + i_1i_3 + i_1i_2 + i_1i_2i_3 + i_0i_3i_4 + i_0i_2i_3 + i_0i_2i_3i_4 + i_0i_1i_3i_4 + i_0i_1i_2i_3$
$p =$	$S_{13}(i_0, i_1, i_2, i_3, i_4) =$	$i_4 + i_3 + i_3i_4 + i_2 + i_1 + i_1i_3i_4 + i_0i_4 + i_0i_3i_4 + i_0i_2 + i_0i_2i_3 + i_0i_2i_3i_4 + i_0i_1i_3i_4 + i_0i_1i_2i_3$
$q =$	$S_{14}(i_0, i_1, i_2, i_3, i_4) =$	$i_4 + i_3i_4 + i_2 + i_2i_3 + i_2i_3i_4 + i_1 + i_1i_2 + i_0 + i_0i_1i_3 + i_0i_1i_3i_4$

Table B.2: The truth table of the 14 Boolean functions

Input	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}
00000	1	0	1	1	1	0	1	1	1	0	0	0	0	0
00001	0	0	0	1	0	0	0	1	0	0	0	1	1	1
00010	0	1	0	0	0	1	1	0	0	0	1	0	1	0
00011	0	1	1	0	1	0	1	1	0	1	1	1	1	0
00100	1	0	1	0	1	0	0	0	1	1	0	1	1	1
00101	1	1	1	1	1	1	1	0	1	0	1	0	0	0
00110	1	1	1	1	1	1	0	1	1	1	1	0	0	0
00111	0	0	0	0	0	1	1	0	1	0	0	0	0	1
01000	1	1	0	1	0	1	0	1	0	0	0	0	1	1
01001	1	0	1	1	0	1	1	0	0	1	0	1	0	0
01010	1	0	1	0	0	0	0	0	1	1	1	1	0	1
01011	0	0	0	1	1	1	0	1	1	1	0	0	1	1
01100	0	1	0	0	1	1	1	1	0	1	1	0	0	1
01101	0	1	0	0	0	0	0	0	0	0	1	1	1	0
01110	0	0	0	1	1	0	0	0	1	0	0	1	1	0
01111	1	1	1	0	0	0	1	1	0	1	1	1	0	1
10000	0	0	1	1	0	1	0	1	1	0	1	0	0	1
10001	1	1	0	1	1	1	0	0	1	1	1	1	0	0
10010	1	1	0	0	0	0	1	1	1	0	0	0	1	1
10011	0	0	1	1	0	1	0	1	0	0	1	0	1	1
10100	1	0	1	1	1	1	1	0	0	0	1	1	0	0
10101	1	0	1	0	0	0	1	1	1	1	0	0	0	1
10110	0	1	0	0	1	0	0	0	0	1	0	1	0	1
10111	0	1	1	0	1	0	1	1	0	1	0	1	1	0
11000	1	0	0	0	1	0	0	0	0	1	1	0	1	0
11001	1	0	0	0	0	0	1	1	0	0	0	1	1	1
11010	0	0	0	1	0	1	1	0	1	1	0	1	0	1
11011	1	1	1	0	1	0	0	0	1	0	1	0	0	0
11100	0	1	1	0	0	0	0	1	1	1	0	0	1	0
11101	0	1	0	1	1	1	1	0	0	1	1	1	1	1
11110	1	1	1	1	1	1	1	0	0	0	1	1	0	0
11111	0	0	0	1	0	1	0	1	1	0	0	0	1	0

B.2 Algebraic Equations at Time $t = -31$

From Equation 5.1 in Section 5.1, during the initialisation process, the system of equations of the new 4-bit word after the first clock of the registers A can be written in term of stage bits as follows:

$$\begin{aligned}
a_{0,0}^{-31} =& a_{9,0}^{-32} \oplus S_1(a_{3,3}^{-31}, a_{1,1}^{-31}, a_{2,3}^{-31}, a_{4,2}^{-31}, a_{8,0}^{-31}) \oplus \\
& S_9(a_{2,1}^{-31}, a_{3,2}^{-31}, a_{6,3}^{-31}, a_{7,0}^{-31}, a_{9,1}^{-31}) \oplus e_0^{-32} \oplus b_{0out}^{-32} \oplus v_4 \\
a_{0,1}^{-31} =& a_{9,1}^{-32} \oplus S_2(a_{1,3}^{-31}, a_{2,0}^{-31}, a_{5,1}^{-31}, a_{5,3}^{-31}, a_{6,2}^{-31}) \oplus \\
& S_{10}(a_{4,0}^{-31}, a_{1,2}^{-31}, a_{6,1}^{-31}, a_{7,3}^{-31}, a_{9,0}^{-31}) \oplus e_1^{-32} \oplus b_{1out}^{-32} \oplus v_5 \\
a_{0,2}^{-31} =& a_{9,2}^{-32} \oplus S_3(a_{2,1}^{-31}, a_{3,2}^{-31}, a_{6,3}^{-31}, a_{7,0}^{-31}, a_{9,1}^{-31}) \oplus \\
& S_{11}(a_{3,1}^{-31}, a_{4,1}^{-31}, a_{5,0}^{-31}, a_{7,2}^{-31}, a_{9,3}^{-31}) \oplus e_2^{-32} \oplus b_{2out}^{-32} \oplus v_6 \\
a_{0,3}^{-31} =& a_{9,3}^{-32} \oplus S_4(a_{4,0}^{-31}, a_{1,2}^{-31}, a_{6,1}^{-31}, a_{7,3}^{-31}, a_{9,0}^{-31}) \oplus \\
& S_{12}(a_{5,2}^{-31}, a_{4,3}^{-31}, a_{6,0}^{-31}, a_{8,1}^{-31}, a_{9,2}^{-31}) \oplus e_3^{-32} \oplus b_{3out}^{-32} \oplus v_7
\end{aligned} \tag{B.1}$$

by substituting key and IV bits in Equation B.1, the above equations can be written as follows:

$$\begin{aligned}
a_{0,0}^{-31} = & k_5k_{11}k_{15}k_{18} \oplus k_9k_{14}k_{27}k_{28} \oplus k_5k_{11}k_{15} \oplus k_5k_{11}k_{18} \oplus k_5k_{15}k_{18} \oplus k_{11}k_{15}k_{18} \\
& \oplus k_9k_{14}k_{28} \oplus k_9k_{27}k_{28} \oplus k_{14}k_{27}k_{28} \oplus k_5k_{11} \oplus k_5k_{15} \oplus k_{11}k_{15} \oplus k_5k_{18} \oplus k_{11}k_{18} \\
& \oplus k_9k_{27} \oplus k_9k_{28} \oplus k_{27}k_{28} \oplus k_{14} \oplus k_{15} \oplus k_{18} \oplus k_{28} \oplus k_{41} \oplus k_{55} \oplus k_{60} \oplus v_4
\end{aligned}$$

$$\begin{aligned}
a_{0,1}^{-31} =& k_7k_8k_{21}k_{26} \oplus k_7k_8k_{21} \oplus k_7k_8k_{23} \oplus k_8k_{23}k_{26} \oplus k_6k_{16}k_{31} \oplus k_{16}k_{25}k_{31} \oplus k_7k_8 \\
& \oplus k_6k_{16} \oplus k_{16}k_{25} \oplus k_7k_{26} \oplus k_8k_{26} \oplus k_{21}k_{26} \oplus k_6k_{31} \oplus k_8 \oplus k_{23} \oplus k_{25} \oplus k_{44} \\
& \oplus k_{49} \oplus k_{51} \oplus k_{62} \oplus v_5
\end{aligned}$$

$$\begin{aligned}
a_{0,2}^{-31} =& k_9k_{14}k_{27} \oplus k_9k_{14}k_{28} \oplus k_9k_{27}k_{28} \oplus k_{17}k_{20} \oplus k_{27}k_{28} \oplus k_{13} \oplus k_{14} \oplus k_{28} \oplus k_{30} \\
& \oplus k_{43} \oplus k_{46} \oplus k_{52} \oplus k_{61} \oplus v_6 \oplus 1
\end{aligned}$$

$$a_{0,3}^{-31} = k_6k_{16} \oplus k_{19}k_{24} \oplus k_{16}k_{25} \oplus k_{24} \oplus k_{25} \oplus k_{31} \oplus k_{40} \oplus k_{53} \oplus k_{58} \oplus v_7 \oplus 1$$

Using Equations 5.6 and 5.7 of the initialisation process, the system of equations for the first 4-bits word of the registers B after the first clock can be described in the term of key-IV bits as follows:

$$\begin{aligned}
b_{0,0}^{-31} = {} & k_{10}k_{13}k_{17}k_{20}k_{29}k_{30} \oplus k_{10}k_{13}k_{17}k_{20}k_{29} \oplus k_{12}k_{13}k_{17}k_{20}k_{30} \oplus k_{10}k_{13}k_{20}k_{29}k_{30} \\
& \oplus k_{10}k_{17}k_{20}k_{29}k_{30} \oplus k_{13}k_{17}k_{20}k_{29}k_{30} \oplus k_{12}k_{13}k_{17}k_{20} \oplus k_{10}k_{13}k_{20}k_{29} \\
& \oplus k_{10}k_{17}k_{20}k_{29} \oplus k_{13}k_{17}k_{20}k_{29} \oplus k_{12}k_{13}k_{20}k_{30} \oplus k_{12}k_{17}k_{20}k_{30} \oplus k_{13}k_{17}k_{20}k_{30} \\
& \oplus k_{10}k_{13}k_{29}k_{30} \oplus k_{10}k_{17}k_{29}k_{30} \oplus k_{10}k_{20}k_{29}k_{30} \oplus k_{13}k_{20}k_{29}k_{30} \oplus k_{17}k_{20}k_{29}k_{30} \\
& \oplus k_{12}k_{13}k_{20} \oplus k_{12}k_{17}k_{20} \oplus k_{13}k_{17}k_{20} \oplus k_{10}k_{13}k_{29} \oplus k_{10}k_{17}k_{29} \oplus k_{10}k_{19}k_{29} \\
& \oplus k_{13}k_{20}k_{29} \oplus k_{17}k_{20}k_{29} \oplus k_{10}k_{22}k_{29} \oplus k_{12}k_{13}k_{30} \oplus k_{12}k_{17}k_{30} \oplus k_{12}k_{20}k_{30} \\
& \oplus k_{13}k_{20}k_{30} \oplus k_{17}k_{20}k_{30} \oplus k_{10}k_{29}k_{30} \oplus k_{13}k_{29}k_{30} \oplus k_{17}k_{29}k_{30} \oplus k_{20}k_{29}k_{30} \\
& \oplus k_{10}k_{29}k_{56} \oplus k_{10}k_{29}k_{57} \oplus k_{10}k_{29}v_{0} \oplus k_{10}k_{29}v_{1} \oplus k_{12}k_{13} \oplus k_{12}k_{17} \oplus k_{12}k_{19} \\
& \oplus k_{13}k_{20} \oplus k_{17}k_{20} \oplus k_{12}k_{22} \oplus k_{10}k_{29} \oplus k_{13}k_{29} \oplus k_{17}k_{29} \oplus k_{19}k_{29} \oplus k_{22}k_{29} \\
& \oplus k_{12}k_{30} \oplus k_{13}k_{30} \oplus k_{17}k_{30} \oplus k_{20}k_{30} \oplus k_{29}k_{30} \oplus k_{12}k_{56} \oplus k_{29}k_{56} \oplus k_{12}k_{57} \oplus k_{29}k_{57} \\
& \oplus k_{12}v_{0} \oplus k_{29}v_{0} \oplus k_{12}v_{1} \oplus k_{29}v_{1} \oplus k_{12} \oplus k_{13} \oplus k_{17} \oplus k_{29} \oplus k_{30} \oplus k_{56} \oplus v_{0} \oplus 1
\end{aligned}$$

$$\begin{aligned}
b_{0,1}^{-31} = {} & k_{5}k_{10}k_{11}k_{15}k_{18}k_{29} \oplus k_{5}k_{11}k_{12}k_{15}k_{18} \oplus k_{5}k_{10}k_{11}k_{15}k_{29} \oplus k_{5}k_{10}k_{11}k_{18}k_{29} \\
& \oplus k_{5}k_{11}k_{15}k_{18}k_{29} \oplus k_{5}k_{11}k_{12}k_{15} \oplus k_{5}k_{11}k_{12}k_{18} \oplus k_{5}k_{10}k_{15}k_{29} \oplus k_{5}k_{11}k_{15}k_{29} \\
& \oplus k_{5}k_{11}k_{18}k_{29} \oplus k_{10}k_{15}k_{18}k_{29} \oplus k_{5}k_{12}k_{15} \oplus k_{12}k_{15}k_{18} \oplus k_{5}k_{10}k_{29} \oplus k_{10}k_{11}k_{29} \\
& \oplus k_{5}k_{15}k_{29} \oplus k_{10}k_{15}k_{29} \oplus k_{15}k_{18}k_{29} \oplus k_{10}k_{19}k_{29} \oplus k_{10}k_{22}k_{29} \oplus k_{10}k_{29}k_{57} \\
& \oplus k_{10}k_{29}k_{58} \oplus k_{10}k_{29}v_{1} \oplus k_{10}k_{29}v_{2} \oplus k_{5}k_{12} \oplus k_{11}k_{12} \oplus k_{12}k_{15} \oplus k_{12}k_{19} \oplus k_{12}k_{22} \\
& \oplus k_{5}k_{29} \oplus k_{10}k_{29} \oplus k_{11}k_{29} \oplus k_{15}k_{29} \oplus k_{19}k_{29} \oplus k_{22}k_{29} \oplus k_{12}k_{57} \oplus k_{29}k_{57} \oplus k_{12}k_{58} \\
& \oplus k_{29}k_{58} \oplus k_{12}v_{1} \oplus k_{29}v_{1} \oplus k_{12}v_{2} \oplus k_{29}v_{2} \oplus k_{12} \oplus k_{19} \oplus k_{22} \oplus k_{29} \oplus k_{57} \oplus v_{1}
\end{aligned}$$

$$\begin{aligned}
b_{0,2}^{-31} = {} & k_{5}k_{10}k_{11}k_{15}k_{18}k_{29} \oplus k_{7}k_{8}k_{10}k_{21}k_{23}k_{29} \oplus k_{7}k_{8}k_{10}k_{23}k_{26}k_{29} \oplus k_{7}k_{10}k_{21}k_{23}k_{26}k_{29} \\
& \oplus k_{5}k_{11}k_{12}k_{15}k_{18} \oplus k_{7}k_{8}k_{12}k_{21}k_{23} \oplus k_{7}k_{8}k_{12}k_{23}k_{26} \oplus k_{7}k_{12}k_{21}k_{23}k_{26} \\
& \oplus k_{5}k_{10}k_{11}k_{15}k_{29} \oplus k_{5}k_{10}k_{11}k_{18}k_{29} \oplus k_{5}k_{11}k_{15}k_{18}k_{29} \oplus k_{7}k_{8}k_{10}k_{21}k_{29} \\
& \oplus k_{7}k_{8}k_{21}k_{23}k_{29} \oplus k_{7}k_{8}k_{10}k_{26}k_{29} \oplus k_{7}k_{8}k_{23}k_{26}k_{29} \oplus k_{8}k_{10}k_{23}k_{26}k_{29} \\
& \oplus k_{7}k_{21}k_{23}k_{26}k_{29} \oplus k_{5}k_{11}k_{12}k_{15} \oplus k_{5}k_{11}k_{12}k_{18} \oplus k_{5}k_{11}k_{15}k_{18} \oplus k_{7}k_{8}k_{12}k_{21} \\
& \oplus k_{7}k_{8}k_{12}k_{26} \oplus k_{8}k_{12}k_{23}k_{26} \oplus k_{7}k_{8}k_{10}k_{29} \oplus k_{5}k_{10}k_{15}k_{29} \oplus k_{5}k_{11}k_{15}k_{29} \\
& \oplus k_{5}k_{11}k_{18}k_{29} \oplus k_{10}k_{15}k_{18}k_{29} \oplus k_{7}k_{8}k_{21}k_{29} \oplus k_{8}k_{10}k_{21}k_{29} \oplus k_{7}k_{10}k_{23}k_{29} \\
& \oplus k_{7}k_{8}k_{26}k_{29} \oplus k_{7}k_{10}k_{26}k_{29} \oplus k_{8}k_{10}k_{26}k_{29} \oplus k_{8}k_{23}k_{26}k_{29} \oplus k_{10}k_{23}k_{26}k_{29} \\
& \oplus k_{7}k_{8}k_{12} \oplus k_{5}k_{11}k_{15} \oplus k_{5}k_{12}k_{15} \oplus k_{5}k_{11}k_{18} \oplus k_{12}k_{15}k_{18} \oplus k_{8}k_{12}k_{21} \oplus k_{7}k_{12}k_{23} \\
& \oplus k_{7}k_{12}k_{26} \oplus k_{8}k_{12}k_{26} \oplus k_{12}k_{23}k_{26} \oplus k_{7}k_{8}k_{29} \oplus k_{5}k_{10}k_{29} \oplus k_{10}k_{11}k_{29} \oplus k_{5}k_{15}k_{29} \\
& \oplus k_{10}k_{15}k_{29} \oplus k_{15}k_{18}k_{29} \oplus k_{8}k_{21}k_{29} \oplus k_{10}k_{21}k_{29} \oplus k_{7}k_{23}k_{29} \oplus k_{10}k_{23}k_{29} \\
& \oplus k_{7}k_{26}k_{29} \oplus k_{8}k_{26}k_{29} \oplus k_{23}k_{26}k_{29} \oplus k_{10}k_{29}k_{58} \oplus k_{10}k_{29}k_{59} \oplus k_{10}k_{29}v_{2} \\
& \oplus k_{10}k_{29}v_{3} \oplus k_{5}k_{12} \oplus k_{11}k_{12} \oplus k_{5}k_{15} \oplus k_{12}k_{15} \oplus k_{15}k_{18} \oplus k_{12}k_{21} \oplus k_{12}k_{23} \\
& \oplus k_{5}k_{29} \oplus k_{11}k_{29} \oplus k_{15}k_{29} \oplus k_{21}k_{29} \oplus k_{23}k_{29} \oplus k_{12}k_{58} \oplus k_{29}k_{58} \oplus k_{12}k_{59} \\
& \oplus k_{29}k_{59} \oplus k_{12}v_{2} \oplus k_{29}v_{2} \oplus k_{12}v_{3} \oplus k_{29}v_{3} \oplus k_{5} \oplus k_{11} \oplus k_{15} \oplus k_{58} \oplus v_{2} \oplus 1
\end{aligned}$$

$$\begin{aligned}
b_{0,3}^{-31} = {} & k_7k_8k_{10}k_{21}k_{23}k_{29} \oplus k_7k_8k_{10}k_{23}k_{26}k_{29} \\
& \oplus k_7k_{10}k_{21}k_{23}k_{26}k_{29} \oplus k_{10}k_{13}k_{17}k_{20}k_{29}k_{30} \\
& \oplus k_7k_8k_{12}k_{21}k_{23} \oplus k_7k_8k_{12}k_{23}k_{26} \oplus k_7k_{12}k_{21}k_{23}k_{26} \\
& \oplus k_{10}k_{13}k_{17}k_{20}k_{29} \oplus k_7k_8k_{10}k_{21}k_{29} \oplus k_7k_8k_{21}k_{23}k_{29} \\
& \oplus k_7k_8k_{10}k_{26}k_{29} \oplus k_7k_8k_{23}k_{26}k_{29} \oplus k_8k_{10}k_{23}k_{26}k_{29} \\
& \oplus k_7k_{21}k_{23}k_{26}k_{29} \oplus k_{12}k_{13}k_{17}k_{20}k_{30} \\
& \oplus k_{10}k_{13}k_{20}k_{29}k_{30} \oplus k_{10}k_{17}k_{20}k_{29}k_{30} \\
& \oplus k_{13}k_{17}k_{20}k_{29}k_{30} \oplus k_{12}k_{13}k_{17}k_{20} \oplus k_7k_8k_{12}k_{21} \\
& \oplus k_7k_8k_{21}k_{23} \oplus k_7k_8k_{12}k_{26} \oplus k_7k_8k_{23}k_{26} \\
& \oplus k_8k_{12}k_{23}k_{26} \oplus k_7k_{21}k_{23}k_{26} \oplus k_7k_8k_{10}k_{29} \\
& \oplus k_{10}k_{13}k_{20}k_{29} \oplus k_{10}k_{17}k_{20}k_{29} \oplus k_{13}k_{17}k_{20}k_{29} \\
& \oplus k_7k_8k_{21}k_{29} \oplus k_8k_{10}k_{21}k_{29} \oplus k_7k_{10}k_{23}k_{29} \\
& \oplus k_7k_8k_{26}k_{29} \oplus k_7k_{10}k_{26}k_{29} \oplus k_8k_{10}k_{26}k_{29} \\
& \oplus k_8k_{23}k_{26}k_{29} \oplus k_{10}k_{23}k_{26}k_{29} \oplus k_{12}k_{13}k_{20}k_{30} \\
& \oplus k_{12}k_{17}k_{20}k_{30} \oplus k_{10}k_{13}k_{29}k_{30} \oplus k_{10}k_{17}k_{29}k_{30} \\
& \oplus k_{10}k_{20}k_{29}k_{30} \oplus k_{13}k_{20}k_{29}k_{30} \oplus k_{17}k_{20}k_{29}k_{30} \\
& \oplus k_7k_8k_{12} \oplus k_{12}k_{13}k_{20} \oplus k_{12}k_{17}k_{20} \oplus k_7k_8k_{21} \\
& \oplus k_8k_{12}k_{21} \oplus k_7k_{12}k_{23} \oplus k_7k_8k_{26} \oplus k_7k_{12}k_{26} \\
& \oplus k_8k_{12}k_{26} \oplus k_8k_{23}k_{26} \oplus k_{12}k_{23}k_{26} \oplus k_7k_8k_{29} \\
& \oplus k_{10}k_{13}k_{29} \oplus k_{10}k_{17}k_{29} \oplus k_{13}k_{20}k_{29} \oplus k_{17}k_{20}k_{29} \\
& \oplus k_8k_{21}k_{29} \oplus k_{10}k_{21}k_{29} \oplus k_7k_{23}k_{29} \oplus k_{10}k_{23}k_{29} \\
& \oplus k_7k_{26}k_{29} \oplus k_8k_{26}k_{29} \oplus k_{23}k_{26}k_{29} \oplus k_{12}k_{13}k_{30} \\
& \oplus k_{12}k_{17}k_{30} \oplus k_{12}k_{20}k_{30} \oplus k_{10}k_{29}k_{30} \oplus k_{13}k_{29}k_{30} \\
& \oplus k_{17}k_{29}k_{30} \oplus k_{20}k_{29}k_{30} \oplus k_{10}k_{29}k_{56} \oplus k_{10}k_{29}k_{59} \\
& \oplus k_{10}k_{29}v_0 \oplus k_{10}k_{29}v_3 \oplus k_7k_8 \oplus k_{12}k_{13} \oplus k_{12}k_{17} \\
& \oplus k_8k_{21} \oplus k_{12}k_{21} \oplus k_7k_{23} \oplus k_{12}k_{23} \oplus k_7k_{26} \oplus k_8k_{26} \\
& \oplus k_{23}k_{26} \oplus k_{13}k_{29} \oplus k_{17}k_{29} \oplus k_{21}k_{29} \oplus k_{23}k_{29} \oplus k_{12}k_{30} \\
& \oplus k_{29}k_{30} \oplus k_{12}k_{56} \oplus k_{29}k_{56} \oplus k_{12}k_{59} \oplus k_{29}k_{59} \oplus k_{12}v_0 \\
& \oplus k_{29}v_0 \oplus k_{12}v_3 \oplus k_{29}v_3 \oplus k_{21} \oplus k_{23} \oplus k_{59} \oplus v_3 \oplus 1
\end{aligned}$$

Using Equation 5.11, the content of memory F after the first clock can be described in term of the key-IV bits as follows:

$$\begin{aligned}
f_0^{-31} = {} & k_9k_{12}k_{14}k_{27}k_{28}k_{29} \oplus k_9k_{10}k_{14}k_{27}k_{28} \oplus k_9k_{12}k_{14}k_{27}k_{28} \oplus k_9k_{12}k_{14}k_{28}k_{29} \\
& \oplus k_9k_{12}k_{27}k_{28}k_{29} \oplus k_9k_{14}k_{27}k_{28}k_{29} \oplus k_{12}k_{14}k_{27}k_{28}k_{29} \oplus k_9k_{10}k_{14}k_{28} \\
& \oplus k_9k_{12}k_{14}k_{28} \oplus k_9k_{10}k_{27}k_{28} \oplus k_9k_{12}k_{27}k_{28} \oplus k_{10}k_{14}k_{27}k_{28} \oplus k_{12}k_{14}k_{27}k_{28} \\
& \oplus k_9k_{12}k_{27}k_{29} \oplus k_9k_{12}k_{28}k_{29} \oplus k_9k_{14}k_{28}k_{29} \oplus k_9k_{27}k_{28}k_{29} \oplus k_{12}k_{27}k_{28}k_{29} \\
& \oplus k_{14}k_{27}k_{28}k_{29} \oplus k_9k_{10}k_{27} \oplus k_9k_{12}k_{27} \oplus k_9k_{10}k_{28} \oplus k_9k_{12}k_{28} \oplus k_{10}k_{27}k_{28} \\
& \oplus k_{12}k_{27}k_{28} \oplus k_{12}k_{14}k_{29} \oplus k_9k_{27}k_{29} \oplus k_9k_{28}k_{29} \oplus k_{12}k_{28}k_{29} \oplus k_{27}k_{28}k_{29} \oplus k_{10}k_{14} \\
& \oplus k_{12}k_{14} \oplus k_{10}k_{28} \oplus k_{12}k_{28} \oplus k_{12}k_{29} \oplus k_{14}k_{29} \oplus k_{28}k_{29} \oplus k_{10} \oplus k_{12} \oplus k_{29}
\end{aligned}$$

$$\begin{aligned}
f_1^{-31} = {} & k_6k_{12}k_{16}k_{29}k_{31} \oplus k_{12}k_{16}k_{25}k_{29}k_{31} \oplus k_6k_{12}k_{16}k_{29} \oplus k_{12}k_{16}k_{25}k_{29} \\
& \oplus k_6k_{10}k_{16}k_{31} \oplus k_6k_{12}k_{16}k_{31} \oplus k_{10}k_{16}k_{25}k_{31} \oplus k_{12}k_{16}k_{25}k_{31} \\
& \oplus k_6k_{12}k_{29}k_{31} \oplus k_6k_{16}k_{29}k_{31} \oplus k_{16}k_{25}k_{29}k_{31} \oplus k_6k_{10}k_{16} \oplus k_6k_{12}k_{16} \\
& \oplus k_{10}k_{16}k_{25} \oplus k_{12}k_{16}k_{25} \oplus k_6k_{16}k_{29} \oplus k_{12}k_{25}k_{29} \oplus k_{16}k_{25}k_{29} \\
& \oplus k_6k_{10}k_{31} \oplus k_6k_{12}k_{31} \oplus k_6k_{29}k_{31} \oplus k_{10}k_{25} \oplus k_{12}k_{25} \oplus k_{25}k_{29}
\end{aligned}$$

$$\begin{aligned} f_2^{-31} = {} & k_{12}k_{17}k_{20}k_{29} \oplus k_{10}k_{17}k_{20} \oplus k_{12}k_{17}k_{20} \oplus k_{12}k_{13}k_{29} \oplus k_{17}k_{20}k_{29} \\ & \oplus k_{12}k_{29}k_{30} \oplus k_{10}k_{13} \oplus k_{12}k_{13} \oplus k_{13}k_{29} \oplus k_{10}k_{30} \oplus k_{12}k_{30} \oplus k_{29}k_{30} \end{aligned}$$

$$\begin{aligned} f_3^{-31} = {} & k_{12}k_{19}k_{24}k_{29} \oplus k_{10}k_{19}k_{24} \oplus k_{12}k_{19}k_{24} \oplus k_{12}k_{24}k_{29} \\ & \oplus k_{19}k_{24}k_{29} \oplus k_{10}k_{24} \oplus k_{12}k_{24} \oplus k_{24}k_{29} \end{aligned}$$

Note: During keystream generation, there is no IV input to both register A and B and no feedback from the output function (labeled D) to register A, so, the complexity of the system of equations is less than that during the initialisation process.

Index

Bibliography

[1] M. Ågren, M. Hell, T. Johansson, and W. Meier. Grain-128a: a new version of Grain-128 with optional authentication. *International Journal of Wireless and Mobile Computing*, 5(1):48–59, 2011.

[2] A. Alhamdan. A Study of the Initialisation Process of the A5/1 Stream Cipher. Master's thesis, Queensland University of Technology, October 2008.

[3] F. Armknecht. Algebraic attacks on stream ciphers. *Fourth European Congress on Computational Methods in Applied Sciences and Engineering, Finland*, 2004.

[4] H. Arsham. Performance extrapolation in discrete-event systems simulation. *International Journal of Systems Science*, 27(9):863–869, 1996.

[5] S. Babbage and M. Dodd. The stream cipher MICKEY-128 (version 1). eSTREAM, ECRYPT Stream Cipher Project, Report 2005/016, 2005. www.ecrypt.eu.org/stream/papers.html.

[6] S. Babbage and M. Dodd. The stream cipher MICKEY (version 1). eSTREAM, ECRYPT Stream Cipher Project, Report 2005/015, 2005. http://www.ecrypt.eu.org/stream.

[7] S. Babbage and M. Dodd. The stream cipher MICKEY-128 2.0. eSTREAM, ECRYPT Stream Cipher Project, End of 3rd Phase, 2008. http://www.ecrypt.eu.org/stream/p3ciphers/mickey/mickey128_p3.pdf.

[8] S. Babbage and M. Dodd. The stream cipher MICKEY 2.0. eSTREAM, ECRYPT Stream Cipher Project, End of 3rd Phase, 2008. http://www.ecrypt.eu.org/stream/p3ciphers/mickey/mickey_p3.pdf.

[9] S. H. Babbage. Improved exhaustive search attacks on stream ciphers. In *European Convention on Security and Detection, 1995*, pages 161–166, 1995.

[10] S. Banik, S. Maitra, and S. Sarkar. Some Results on Related Key-IV Pairs of Grain. In A. Bogdanov and S. Sanadhya, editors, *Security, Privacy, and Applied Cryptography Engineering*, Lecture Notes in Computer Science, pages 94–110. Springer Berlin Heidelberg, 2012.

[11] S. Banik, S. Maitra, S. Sarkar, and T. Meltem Sönmez. A Chosen IV Related Key Attack on Grain-128a. In C. Boyd and L. Simpson, editors, *Information Security and Privacy*, volume 7959 of *Lecture Notes in Computer Science*, pages 13–26. Springer Berlin Heidelberg, 2013.

[12] E. Barkan, E. Biham, and N. Keller. Instant Ciphertext-Only Cryptanalysis of GSM Encrypted Communication. In D. Boneh, editor, *Advances in Cryptology - CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 600–616. Springer Berlin Heidelberg, 2003.

[13] C. Berbain, H. Gilbert, and A. Maximov. Cryptanalysis of Grain. In M. Robshaw, editor, *Fast Software Encryption*, volume 4047 of *Lecture Notes in Computer Science*, pages 15–29. Springer Berlin Heidelberg, 2006.

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

[15] S. Bewick. Descrambling DVB data according to ETSI common scrambling specification, 09 1998. UK Patent GB2322994A.

[16] S. Bewick. Descrambling DVB data according to ETSI common scrambling standard, 09 1998. UK Patent GB2322995A.

[17] S. Bewick. Digital Video Broadcasting, 06 2000. US Patent 6072873.

[18] E. Biham. New types of cryptanalytic attacks using related keys. *Journal of Cryptology*, 7(4):229–246, 1994.

[19] E. Biham and O. Dunkelman. Cryptanalysis of the A5/1 GSM Stream Cipher. In B. Roy and E. Okamoto, editors, *Progress in Cryptology - INDOCRYPT 2000*, volume 1977 of *Lecture Notes in Computer Science*, pages 43–51. Springer Berlin Heidelberg, 2000.

[20] E. Biham and O. Dunkelman. Differential cryptanalysis in stream ciphers. Cryptology ePrint Archive, Report 2007/218, 2007. http://eprint.iacr.org/.

[21] E. Biham and A. Shamir. Differential cryptanalysis of DES-like cryptosystems. *Journal of Cryptology*, 4(1):3–72, 1991.

[22] A. Biryukov. A new 128 bit key stream cipher : LEX. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/013, 2005. http://www.ecrypt.eu.org/stream.

[23] A. Biryukov and A. Shamir. Cryptanalytic Time/Memory/Data Tradeoffs for Stream Ciphers. In T. Okamoto, editor, *Advances in Cryptology - ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 1–13. Springer Berlin Heidelberg, 2000.

[24] A. Biryukov, A. Shamir, and D. Wagner. Real Time Cryptanalysis of A5/1 on a PC. In G. Goos, J. Hartmanis, J. Leeuwen, and B. Schneier, editors, *Fast Software Encryption*, volume 1978 of *Lecture Notes in Computer Science*, pages 1–18. Springer Berlin Heidelberg, 2001.

[25] A. Biryukov and D. Wagner. Slide Attacks. In L. Knudsen, editor, *Fast Software Encryption*, volume 1636 of *Lecture Notes in Computer Science*, pages 245–259. Springer Berlin Heidelberg, 1999.

[26] A. Biryukov and D. Wagner. Advanced Slide Attacks. In B. Preneel, editor, *Advances in Cryptology - EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 589–606. Springer Berlin Heidelberg, 2000.

[27] T. E. Bjørstad. Cryptanalysis of Grain using Time/Memory/Data Tradeoffs. eSTREAM, ECRYPT Stream Cipher Project, Report 2008/012, 2008. http://www.ecrypt.eu.org/stream.

[28] W. Bosma, Cannon J., and Playoust C. The Magma Algebra System I: The User Language . *Journal of Symbolic Computation* , 24(3):235 – 265, 1997.

[29] A. Braeken, J. Lano, N. Mentens, B. Preneel, and I. Verbauwhede. SFINKS: A synchronous stream cipher for restricted hardware environments. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/026, 2005. www.ecrypt.eu.org/stream/ciphers/sfinks/sfinks.ps.

[30] M. Briceno, I. Goldberg, and D. Wagner. A pedagogical implementation of A5/1, 1999. http://cryptome.org/jya/a51-pi.htm.

[31] J. Chen and A. Miyaji. A New Practical Key Recovery Attack on the Stream Cipher RC4 under Related-Key Model. In X. Lai, M. Yung, and D. Lin, editors, *Information Security and Cryptology*, volume 6584 of *Lecture Notes in Computer Science*, pages 62–76. Springer Berlin Heidelberg, 2011.

[32] Chen, K. and Henricksen, M. and Millan, W. and Fuller, J. and Simpson, L. and Dawson, E. and Lee, H.J. and Moon, S.J. Dragon: A Fast Word Based Stream Cipher. In C. Park and S. Chee, editors, *Information Security and Cryptology - ICISC 2004*, volume 3506 of *Lecture Notes in Computer Science*, pages 33–50. Springer Berlin Heidelberg, 2005.

[33] J. Y. Cho. An improved estimate of the correlation of distinguisher for Dragon. In *Workshop Record of The State of the Art of Stream Ciphers (SASC 2008)*, pages 11–20, 2008.

[34] J. Y. Cho and J. Pieprzyk. An Improved Distinguisher for Dragon. Cryptology ePrint Archive, Report 2007/108, 2007. http://eprint.iacr.org/.

[35] A. Clark, E. Dawson, J. Fuller, J. Golić, H-J. Lee, W. Millan, S-J. Moon, and L. Simpson. The LILI-II Keystream Generator. In L. Batten and J. Seberry, editors, *Information Security and Privacy*, volume 2384 of *Lecture Notes in Computer Science*, pages 25–39. Springer Berlin Heidelberg, 2002.

[36] N. Courtois. Cryptanalysis of Sfinks. In D. Won and S. Kim, editors, *Information Security and Cryptology - ICISC 2005*, volume 3935 of *Lecture Notes in Computer Science*, pages 261–269. Springer Berlin Heidelberg, 2006.

[37] J. Daemen, R. Govaerts, and J. Vandewalle. Resynchronization Weaknesses in Synchronous Stream Ciphers. In T. Helleseth, editor, *Advances*

in Cryptology - EUROCRYPT '93, volume 765 of *Lecture Notes in Computer Science*, pages 159–167. Springer Berlin Heidelberg, 1994.

[38] E. Dawson, A. Clark, J. Golić, W. Millan, L. Penna, and L. Simpson. The LILI-128 keystream generator. In *Proceedings of first NESSIE workshop*. Citeseer, 2000.

[39] E. Dawson, M. Henricksen, and L. Simpson. The Dragon Stream Cipher: Design, Analysis, and Implementation Issues. In M. Robshaw and O. Billet, editors, *New Stream Cipher Designs*, volume 4986 of *Lecture Notes in Computer Science*, pages 20–38. Springer Berlin Heidelberg, 2008.

[40] E. Dawson and L. Nielsen. Automated cryptanalysis of XOR plaintext strings. *Cryptologia*, 20(2):165–181, 1996.

[41] E. Dawson and L. Simpson. Analysis and design issues for synchronous stream ciphers. *Coding Theory and Cryptology*, pages 49–90, 2002.

[42] C. De Cannière, Ö. Kücük, and B. Preneel. Analysis of Grains Initialization Algorithm. In S. Vaudenay, editor, *Progress in Cryptology - AFRICACRYPT 2008*, volume 5023 of *Lecture Notes in Computer Science*, pages 276–289. Springer Berlin Heidelberg, 2008.

[43] C. De Cannière and B. Preneel. Trivium - A Stream Cipher Construction Inspired by Block Cipher Design Principles. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/030 and 2006/021, 2005. http://www.ecrypt.eu.org/stream.

[44] M. Diett. On the Security of Digital Video Broadcast Encryption. Master's thesis, Ruhr-Universität Bochum, October 2007.

[45] L. Ding and J. Guan. Related Key Chosen IV Attack on Grain-128a Stream Cipher. *IEEE Transactions on Information Forensics and Security*, 8(5):803–809, 2013.

[46] O. Dunkelman and N. Keller. Treatment of the initial value in Time-Memory-Data Tradeoff attacks on stream ciphers . *Information Processing Letters* , 107(5):133 – 137, 2008.

[47] Barkan E. and Biham E. Conditional Estimators: An Effective Attack on A5/1. In B. Preneel and S. Tavares, editors, *Selected Areas in Cryptography*, volume 3897 of *Lecture Notes in Computer Science*, pages 1–19. Springer Berlin / Heidelberg, 2006.

[48] P. Ekdahl and T. Johansson. A New Version of the Stream Cipher SNOW. In K. Nyberg and H. Heys, editors, *Selected Areas in Cryptography*, volume 2595 of *Lecture Notes in Computer Science*, pages 47–61. Springer Berlin Heidelberg, 2003.

[49] H. Englund and A. Maximov. Attack the Dragon. In S. Maitra, C.E. Veni Madhavan, and R. Venkatesan, editors, *Progress in Cryptology - INDOCRYPT 2005*, volume 3797 of *Lecture Notes in Computer Science*, pages 130–142. Springer Berlin Heidelberg, 2005.

[50] European Network of Excellence for Cryptology. The eSTREAM Project. `http://www.ecrypt.eu.org/stream/`.

[51] European Standards Organization - European Union. European Telecommunications Standards Institute. `http://http://www.etsi.org/`.

[52] S. Fluhrer, I. Mantin, and A. Shamir. Weaknesses in the Key Scheduling Algorithm of RC4. In S. Vaudenay and A. Youssef, editors, *Selected Areas in Cryptography*, volume 2259 of *Lecture Notes in Computer Science*, pages 1–24. Springer Berlin Heidelberg, 2001.

[53] T. Gendrullis, M. Novotnỳ, and A. Rupp. A Real-World Attack Breaking A5/1 within Hours. In E. Oswald and P. Rohatgi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2008*, volume 5154 of *Lecture Notes in Computer Science*, pages 266–282. Springer Berlin Heidelberg, 2008.

[54] J. Golić. Cryptanalysis of Alleged A5 Stream Cipher. In W. Fumy, editor, *Advances in Cryptology - EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 239–255. Springer Berlin Heidelberg, 1997.

[55] J. Golić. Cryptanalysis of three mutually clock-controlled stop/go shift registers. *IEEE Transactions on Information Theory*, 46(3):1081 –1090, May 2000.

[56] E.K. Grossman and B. Tuckerman. Analysis of a weakened Feistel-like cipher. In *International Conference on Communications*, volume 46, pages 1–46, 1978.

[57] H. Gustafson. *Statistical analysis of symmetric ciphers.* PhD thesis, Queensland University of Technology, July 1996.

[58] M. Hell, T. Johansson, and W. Meier. Grain - A Stream Cipher for Constrained Environments. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/010, 2005. http://www.ecrypt.eu.org/stream.

[59] M. Hell, T. Johansson, and W. Meier. Grain: a stream cipher for constrained environments. *International Journal of Wireless and Mobile Computing*, 2(1):86–93, 2007. http://www.ecrypt.eu.org/stream.

[60] Hell, M. and Johansson, T. and Maximov, A. and Meier, W. A stream cipher proposal: Grain-128. In *IEEE International Symposium on Information Theory, 2006*, pages 1614–1618, 2006.

[61] T. Helleseth, C.J.A. Jansen, O. Kazymyrov, and A. Kholosha. State space cryptanalysis of the MICKEY cipher. In *Information Theory and Applications Workshop (ITA), 2013*, pages 1–10. IEEE, 2013.

[62] M.E. Hellman. A cryptanalytic time-memory trade-off. *IEEE Transactions on Information Theory*, 26(4):401–406, 1980.

[63] M. Hojsík and B. Rudolf. Differential Fault Analysis of Trivium. In K. Nyberg, editor, *Fast Software Encryption*, volume 5086 of *Lecture Notes in Computer Science*, pages 158–172. Springer Berlin Heidelberg, 2008.

[64] M. Hojsík and B. Rudolf. Floating Fault Analysis of Trivium. In D. Chowdhury, V. Rijmen, and A. Das, editors, *Progress in Cryptology - INDOCRYPT 2008*, volume 5365 of *Lecture Notes in Computer Science*, pages 239–250. Springer Berlin Heidelberg, 2008.

[65] J. Hong and W. Kim. TMD-Tradeoff and State Entropy Loss Considerations of Stream Cipher MICKEY. In S. Maitra, C.E. Veni Madhavan, and R. Venkatesan, editors, *Progress in Cryptology - INDOCRYPT 2005*, volume 3797 of *Lecture Notes in Computer Science*, pages 169–182. Springer Berlin Heidelberg, 2005.

[66] J. Hong and P. Sarkar. New Applications of Time Memory Data Tradeoffs. In B. Roy, editor, *Advances in Cryptology - ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 353–372. Springer Berlin Heidelberg, 2005.

[67] J. Hong and P. Sarkar. Rediscovery of Time Memory Tradeoffs. Cryptology ePrint Archive, Report 2005/090, 2005. http://eprint.iacr.org/.

[68] Information Society Technologies (IST) Programme. NESSIE. http://www.cosic.esat.kuleuven.be/nessie/. accessed May 20, 2010.

[69] A. Kerckhoffs. La cryptographie militaire. *Journal des sciences militaires*, IX. pp. 5–38, Jan. 1883, pp. 161–191, Feb. 1883.

[70] S. Khazaei, M. Hassanzadeh, and M. Kiaei. Distinguishing Attack on Grain. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/071, 2005. www.ecrypt.eu.org/stream/papers.html.

[71] S.A. Kiselev and N.N. Tokareva. Reduction of the key space of the cipher A5/1 and invertibility of the next-state function for a stream generator. *Journal of Applied and Industrial Mathematics*, 6(2):194–202, 2012.

[72] A. Klein. Attacks on the RC4 stream cipher. *Designs, Codes and Cryptography*, 48(3):269–286, 2008.

[73] Ö. Küçük. Slide resynchronization attack on the initialization of Grain 1.0. eSTREAM, ECRYPT Stream Cipher Project, Report 2006/044, 2006. http://www.ecrypt.eu.org/stream.

[74] Wei L. Security Analysis of DVB Common Scrambling Algorithm. In *The First International Symposium on Data, Privacy, and E-Commerce - ISDPE 2007*, pages 271–273, 2007.

[75] J. Lano. Sfinks stream cipher source code. eSTREAM, ECRYPT Stream Cipher Project, 2005. http://www.ecrypt.eu.org/stream/sfinks.html.

[76] Y. Lee, K. Jeong, J. Sung, and S. Hong. Related-Key Chosen IV Attacks on Grain-v1 and Grain-128. In Y. Mu, W. Susilo, and J. Seberry, editors, *Information Security and Privacy*, volume 5107 of *Lecture Notes in Computer Science*, pages 321–335. Springer Berlin Heidelberg, 2008.

[77] R. Lidl and H. Niederreiter. *Finite fields*, volume 20. Cambridge University Press, 1997.

[78] I. Mantin. A Practical Attack on the Fixed RC4 in the WEP Mode. In B. Roy, editor, *Advances in Cryptology - ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 395–411. Springer Berlin Heidelberg, 2005.

[79] I. Mantin and A. Shamir. A Practical Attack on Broadcast RC4. In M. Matsui, editor, *Fast Software Encryption*, volume 2355 of *Lecture Notes in Computer Science*, pages 152–164. Springer Berlin Heidelberg, 2002.

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

[81] A. Maximov. Cryptanalysis of the Grain family of stream ciphers. In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, ASIACCS '06, pages 283–288, New York, NY, USA, 2006. ACM.

[82] A. Maximov, T. Johansson, and S. Babbage. An Improved Correlation Attack on A5/1. In H. Handschuh and M Hasan, editors, *Selected Areas in Cryptography*, volume 3357 of *Lecture Notes in Computer Science*, pages 1–18. Springer Berlin Heidelberg, 2005.

[83] A. Menezes, P. Van Oorschot, and S. Vanstone. *Handbook of applied cryptography.* CRC press, 1996.

[84] N. Molavi and X Zhao. A Security Study of Digital TV Distribution Systems. Master's thesis, Department of Computer and Systems Sciences, Royal Institute of Technology, June 2005.

[85] F. Muller. Differential attacks and stream ciphers. In *The State of the Art of Stream Ciphers, Workshop Record, ECRYPT Network of Excellence in Cryptology*, pages 133–146, 2004.

[86] D. Priemuth-Schmid and A. Biryukov. Slid Pairs in Salsa20 and Trivium. In D. Chowdhury, V. Rijmen, and A. Das, editors, *Progress in Cryptology - INDOCRYPT 2008*, volume 5365 of *Lecture Notes in Computer Science*, pages 1–14. Springer Berlin / Heidelberg, 2008.

[87] G. Rose and P. Hawkes. On the Applicability of Distinguishing Attacks Against Stream Ciphers. Cryptology ePrint Archive, Report 2002/142, 2002. http://eprint.iacr.org/.

[88] G. G. Rose and P. Hawkes. Turing: A Fast Stream Cipher. In T. Johansson, editor, *Fast Software Encryption*, volume 2887 of *Lecture Notes in Computer Science*, pages 290–306. Springer Berlin Heidelberg, 2003.

[89] A. Rueppel. *Analysis and design of stream ciphers.* Springer-Verlag New York, Inc., New York, NY, USA, 1986.

[90] A Rukhin, J. Soto, J Nechvatal, M Smid, E. Barker, S. Leigh, M. Levenson, J. Vangel, D. Banks, A. Heckert, J. Dray, and S. Vo. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. NIST, National Institute of Standards and Technology, Computer Security Division, 2010. http://csrc.nist.gov/groups/ST/toolkit/rng/documentation_software.html.

[91] B. Schneier. *Applied cryptography: protocols, algorithms, and source code in C.* John Wiley and Sons, 1996.

[92] C. Shannon. Communication theory of secrecy systems. *Bell system technical journal*, 28(4):656–715, 1949.

[93] Z. Shi, X. Feng, D. Feng, and C. Wu. A Real-Time Key Recovery Attack on the Lightweight Stream Cipher A2U2. In J. Pieprzyk, A. Sadeghi, and M. Manulis, editors, *Cryptology and Network Security*, volume 7712 of *Lecture Notes in Computer Science*, pages 12–22. Springer Berlin Heidelberg, 2012.

[94] A. Sidorenko and B. Schoenmakers. State recovery attacks on pseudorandom generators. In *Lectures Notes in Informatics (LNI) Western European Workshop on Research in Cryptology*, volume 74, pages 53–63. Citeseer, 2005.

[95] L. Simpson, E. Dawson, J. Golić, and W. Millan. LILI Keystream Generator. In D.. Stinson and S. Tavares, editors, *Selected Areas in Cryptography*, volume 2012 of *Lecture Notes in Computer Science*, pages 248–261. Springer Berlin Heidelberg, 2001.

[96] L. Simpson, M. Henricksen, and W. Yap. Improved Cryptanalysis of the Common Scrambling Algorithm Stream Cipher. In C. Boyd and J. Gonzalez Nieto, editors, *Information Security and Privacy*, volume 5594 of *Lecture Notes in Computer Science*, pages 108–121. Springer Berlin Heidelberg, 2009.

[97] W. Stallings. *Cryptography and Network Security, 4th Edition.* Pearson Prentice Hall, NJ., 2006.

[98] M. Stamp. *Information security: principles and practice.* Wiley, NJ, 2011.

[99] A. Stubblefield, J. Ioannidis, and A.D. Rubin. Using the Fluhrer, Mantin, and Shamir attack to break WEP. In *Network and Distributed Systems Security Symposium (NDSS)*, volume 1722. Citeseer, 2002.

[100] S. Teo. *Analysis of Nonlinear Sequences and Stream Ciphers.* PhD thesis, Queensland University of Technology, March 2013.

[101] E. Tews, J. Wälde, and M. Weiner. Breaking DVB-CSA. In F. Armknecht and S. Lucks, editors, *Research in Cryptology*, volume 7242 of *Lecture Notes in Computer Science*, pages 45–61. Springer Berlin Heidelberg, 2012.

[102] G. S. VERNAM. Secret signaling system, July 22 1919. US Patent 1,310,719.

[103] G. S. Vernam. Cipher printing telegraph systems for secret wire and radio telegraphic communications. *Transactions of the American Institute of Electrical Engineers*, 45:295–301, 1926.

[104] R. Weinmann and K. Wirt. Analysis of the DVB Common Scrambling Algorithm. In D. Chadwick and B. Preneel, editors, *Communications and Multimedia Security*, volume 175 of *IFIP - The International Federation for Information Processing*, pages 195–207. Springer US, 2005.

[105] K. Wirt. Fault Attack on the DVB Common Scrambling Algorithm. In O. Gervasi, M. Gavrilova, V. Kumar, A. Laganà, H. Lee, Y. Mun, D. Taniar, and C. Tan, editors, *Computational Science and Its Applications – ICCSA 2005*, volume 3481 of *Lecture Notes in Computer Science*, pages 577–584. Springer Berlin Heidelberg, 2005.

[106] H. Wu and B. Preneel. Resynchronization Attacks on WG and LEX. In M. Robshaw, editor, *Fast Software Encryption*, volume 4047 of *Lecture Notes in Computer Science*, pages 422–432. Springer Berlin Heidelberg, 2006.

[107] B. Zhang, H. Wu, D. Feng, and F. Bao. Chosen Ciphertext Attack on a New Class of Self-Synchronizing Stream Ciphers. In A. Canteaut and K. Viswanathan, editors, *Progress in Cryptology - INDOCRYPT 2004*, volume 3348 of *Lecture Notes in Computer Science*, pages 73–83. Springer Berlin Heidelberg, 2005.

[108] H. Zhang and X. Wang. Cryptanalysis of Stream Cipher Grain Family. Cryptology ePrint Archive, Report 2009/109, 2009. http://eprint.iacr.org/.