

# **Lightweight Synchronous Stream Ciphers: From Mathematical and Statistical Analysis to Proposed Secure Protocols for Mobile Cloud Computing and RFID Applications**

Ahmed Mohammed Alamer

B. Math, GradDip. Math, M. Math

A thesis submitted in total fulfilment  
of the requirements for the degree of  
Doctor of Philosophy

School of Engineering and Mathematical Sciences  
College of Science, Health and Engineering

La Trobe University

Victoria, Australia

May 2020

*Dedicated to  
my mother and memory of my father*

# Table of Contents

Table of Contents .....	III
List of Figures .....	IX
List of Tables .....	XI
Abbreviations .....	XIII
Abstract .....	XV
Statement of Authorship .....	XVI
Acknowledgments .....	XVII
List of Publications .....	XIX
Chapter 1: Introduction .....	1
1.0 Chapter overview .....	1
1.1 Introduction .....	1
1.2 Background .....	2
1.2.1 Cryptographic methods .....	3
1.3 Cryptosystems targeted in this research .....	4
1.4 Challenges and motivation .....	4
1.5 Research objective and aims .....	5
1.5.1 Thesis approach and research questions .....	6
1.5.2 Components of the thesis .....	8
1.5.2 Significance of the research and implications .....	9
1.5.3 Methodology .....	10
1.6 Thesis contributions .....	11
1.6.1 Statistical tests on SG and SSG .....	11
1.6.2 MICKEY 2.0 reduced variant (MICKEY 2.0.85) .....	11
1.6.3 Mobile cloud computing and FEATHER protocol .....	11
1.6.4 eHealth proposed cryptosystem .....	12
1.7 Thesis structure .....	12
1.7.1 Link between Chapters 3 and 4 .....	15
1.7.2 Links between Chapters 3, 4, 5 and 6 .....	15
1.7.3 Links between Chapters 5, 6 and 7 .....	16
1.8 Summary .....	16
Chapter 2: Literature review and background .....	18
2.0 Chapter overview .....	18
2.1 Introduction .....	18
2.2 Encryption types .....	19

2.3 Mathematical foundation for encryption methods.....	20
2.3.1 Boolean function.....	20
2.3.2 Pseudo-random number generators .....	21
2.4 Cryptanalysis methods and some common attacks .....	24
2.4.1 Internal state and initialisation vector (IV) .....	26
2.4.2 Statistical tests on stream ciphers .....	27
2.4.3 Statistical-based attacks on stream ciphers.....	27
2.4.3 Black-box attacks .....	29
2.5 Lightweight cryptography .....	30
2.6 Neural network predicting models .....	32
2.7 Cloud computing.....	32
2.8 Mobile cloud computing: applications, security and current challenges .....	34
2.9 RFID technology .....	36
2.10 Summary .....	36
Chapter 3: Randomness tests on synchronous lightweight stream ciphers.....	39
3.0 Chapter overview .....	39
3.1 Introduction .....	39
3.2 Mathematical basis .....	40
3.2.1 Finite field .....	40
3.2.2 Algebraic normal form .....	40
3.2.3 Truth table.....	41
3.2.4 Möbius Transform.....	41
3.2.5 Hypothesis testing.....	41
3.2.6 Chi-square test .....	42
3.2.7 Balance theory .....	43
3.2.8 Solomon Golomb for MLS.....	43
3.2.9 Linear complexity and nonlinear complexity.....	44
3.2.10 Maximum order complexity.....	44
3.3 Synchronous stream ciphers.....	44
3.4 Self-synchronous stream ciphers.....	46
3.5 IV-less stream ciphers .....	47
3.6 Linear Feedback Shift Register (LFSR) .....	48
3.6.1 Primitive polynomials and <i>LFSR</i> .....	48
3.6.2 General attack on <i>LFSR</i> based stream ciphers.....	50
3.7 Unique Window Size ( <i>UWS</i> ).....	50
3.8 The d-monomial test.....	52
3.8.1 Running d-monomial tests.....	53

3.9 The shrinking generator .....	54
3.9.1 Shrinking generator period .....	55
3.9.2 Linear complexity .....	55
3.9.3 Attacks on shrinking generator .....	55
3.9.4 <i>UWS</i> on shrinking generator .....	56
3.9.6 <i>d</i> -monomial test results for SG .....	58
3.10 The self-shrinking generator .....	58
3.10.1 The period .....	59
3.10.2 Self-shrinking generator linear complexity .....	59
3.10.3 Attacks.....	60
3.10.4 Statistical tests on SSG .....	61
3.10.5 <i>UWS</i> test on SSG .....	61
3.10.6 The <i>d</i> -monomial test on SSG.....	66
3.11 Comparison between Shrinking Generator and Self-Shrinking Generator results.....	67
3.12 Other <i>d</i> -monomial based tests and results.....	68
3.13 Data distribution .....	69
3.13.1 Statistical modelling: Predicting <i>UWS</i> .....	70
3.13.2 Sensitivity analysis .....	75
3.14 Conclusion.....	76
Chapter 4: Proposed neural network-based prediction models .....	77
4.0 Chapter overview .....	77
4.1 Introduction .....	77
4.2 Background .....	79
4.3 Related work on neural network and security.....	80
4.4 The importance of the neural networks for the <i>UWS</i> .....	82
4.5 Implementation .....	83
4.5.1 Model specifications .....	83
4.5.2 Terminology .....	83
4.5.3 Variables.....	84
4.6 Mathematical background.....	84
4.6.1 Mean square error (MSE).....	85
4.6.2 Rectified linear unit (ReLU) activation function.....	85
4.6.3 Mathematical representation for the neural network models .....	86
4.7 Python for model building .....	88
4.8 Results analysis .....	91
4.8.1 Shrinking generator (SG) results .....	91
4.8.2 Self-shrinking generator (SSG) results .....	92

4.8.3 Comparison of the shrinking generator (SG) and self-shrinking generator (SSG) results .....	93
4.8.4 Influences of model features .....	93
4.8.5 Comparison of the neural network models and linear regression results in Chapter 3 .....	94
4.9 Conclusion .....	95
Chapter 5: Proposed lighter and faster MICKEY 2.0 reduced variant for low cost implementations .....	97
5.0 Chapter overview .....	97
5.1 Introduction .....	97
5.2 The proposed cipher and the design optimisation methodology.....	99
5.3 NIST randomness test .....	100
5.4 Power consumption .....	101
5.5 MICKEY 2.0 internal design .....	101
5.6 Reduction process.....	106
5.6.1 MICKEY 2.0 and MICKEY 2.0.85 algorithms.....	106
5.7 Results of NIST tests.....	111
5.7.1 NIST test results for the keystream.....	111
5.7.2 NIST test results for the ciphertext .....	114
5.8 MICKEY 2.0.85 performance tests .....	116
5.9 Power consumption testing .....	117
5.10 Cryptanalysis .....	118
5.10.1 Many Time Pad Attack .....	118
5.10.2 Cosine similarity attack (cryptanalysis).....	119
5.11 Discussion of results and analysis .....	121
5.12 Conclusion.....	122
Chapter 6: Mobile cloud computing and FEATHER, a proposed lightweight security protocol .	123
6.0 Chapter overview .....	123
6.1 Introduction .....	123
6.1.1 Cloud computing .....	125
6.2 Background for mobile cloud computing.....	125
6.2.1 The advantage of using stream ciphers in small devices .....	126
6.2.3 Using lightweight stream ciphers in cloud computing and mobile cloud computing	127
6.2.5 AES and CLOAK protocol .....	128
6.2.6 Motivation and challenges.....	128
6.3 The lightweight protocol FEATHER .....	129
6.3.1 FEATHER protocol design principles .....	130
6.4 Protocol implementation .....	134

6.5 Results and analysis .....	140
6.5.1 FEATHER speed performance .....	140
6.5.2 Power consumption .....	142
6.5.3 FEATHER vs CLOAK .....	142
6.6 Attack analysis.....	143
6.7 Discussion.....	144
6.8 Conclusion.....	145
Chapter 7: Proposed security cryptosystem with proposed device for security application in eHealth without internet connectivity: Near Field Secure Data Extractor .....	146
7.0 Chapter overview .....	146
7.1 Introduction .....	146
7.2 Background .....	149
7.2.1 Lightweight cryptosystem.....	151
7.2.2 Physically Unclonable Functions.....	152
7.3 eHealth as case study and illustrative scenario .....	153
7.3.1 eHealth scenario description .....	153
7.3.2 eHealth scenario process .....	154
7.3.4 Proposed cryptosystem as solution for eHealth setting.....	157
7.4 Major processes of eHealth scenario setting and practical low-power ciphers applications .....	159
7.5 NFSDE device components .....	160
7.5.1 NFSDE device components .....	161
7.6 Tag creation and implementation process .....	163
7.6.1 Creation of the medical provider tag.....	163
7.6.2 Creation of the patient tag.....	166
7.7 Key generation, storage and distribution .....	169
7.8 Device processes.....	170
7.8.1 Device activation (unlock).....	170
7.8.2 Procedure for reading the patient medical record .....	172
7.9 The emulation processes for NFSDE and testing .....	173
7.9.1 The emulation processes for NFSDE .....	173
7.9.2 Testing and running the emulator .....	174
7.10 Attacks analysis.....	175
7.10.1 Known plaintext attacks.....	176
7.10.2 Brute force attack .....	177
7.10.3 Chosen IV attack.....	177
7.10.4 Two-time pad/reused key.....	178
7.10.5 Denial-of-service attack .....	178

7.10.6 Insider attack.....	179
7.10.7 Impersonation attack.....	179
7.10.8 Man in the middle attack.....	179
7.10.9 Side channel attacks.....	180
7.11 Overall analysis and discussion.....	181
7.11.1 Providing multi-factor authentication without connectivity.....	181
7.11.2 Providing privacy by design.....	182
7.11.3 Providing key distribution without connectivity.....	182
7.11.4 Other applications.....	184
7.12 Conclusion.....	185
Chapter 8: Discussion and Conclusion.....	186
8.0 Chapter overview.....	186
8.1 Introduction.....	186
8.2 Discussion of thesis rationale and overview of outcomes.....	187
8.3 Using unique window size and d-monomial tests as randomness tests.....	188
8.4 Developing proposed novel neural network-based prediction models.....	188
8.5 Developing and testing the proposed MICKEY 2.0.85 cipher.....	189
8.6 Developing new lightweight encryption method: FEATHER lightweight security protocol.....	190
8.7 Developing proposed lightweight cryptosystem with NFSDE prototype device.....	192
8.8 Thesis findings and contribution to the literature.....	193
8.9 Conclusion.....	197
8.10 Possible future research directions.....	198
Appendices.....	199
Appendices for Chapter 3.....	200
Appendices for Chapter 5.....	218
Appendix for Chapter 6.....	222
Appendices for Chapter 7.....	226
References.....	230

# List of Figures

Figure 1. 1 Research stages.....	6
Figure 1. 2 Thesis main blocks and overall focus.....	8
Figure 1. 3 Relationship between thesis themes and chapters .....	15
Figure 2. 1 Black-box principle .....	29
Figure 2. 2 Cloud computing structure and layers .....	33
Figure 3. 1 Synchronous stream ciphers general design .....	45
Figure 3. 2 Self-synchronous stream ciphers general design.....	46
Figure 3. 3 IV-less stream ciphers .....	47
Figure 3. 4 Shrinking generator design.....	55
Figure 3. 5 Unique window size 19 distribution for Shrinking Generator.....	57
Figure 3. 6 Unique window size 20 distribution for Shrinking Generator.....	57
Figure 3. 7 Self-shrinking generator design.....	59
Figure 3. 8 Unique window size 23 distribution for Self-Shrinking Generator.....	64
Figure 3. 9 Self-Shrinking Generator degrees from 4 to 24 vs Unique Window Size.....	64
Figure 3. 10 Unique Window Size 19 for Self-Shrinking Generator with weight 5 and all weights.....	65
Figure 3. 11 Unique Window Size 19 for Self-Shrinking Generator with weight 5 and all weights, with smooth line .....	65
Figure 3. 12 Unique Window Size 20 different kinds of distributions for Shrinking Generator..	70
Figure 3. 13 Comparison of cumulative density functions of observed and theoretical distributions, Unique Window Size .....	71
Figure 3. 14 Unique Window Size 20 lognormal distribution for Shrinking Generator.....	71
Figure 4. 1 The output of the neuron .....	85
Figure 4. 2 ReLU activation function graph .....	86
Figure 4. 3 Python code using Keras for the neural network model for SG (for SSG, the input in Section 2 is changed to two inputs) .....	90
Figure 5. 1 MICKEY cipher family general internal design.....	102
Figure 5. 2 The core logical process for the MICKEY cipher family.....	103
Figure 5. 3 Process flow for the linear register (R).....	104
Figure 5. 4 Process flow for the nonlinear register (S) .....	105
Figure 5. 5 Comparison histogram of NIST test passing rates for MICKEY 2.0, MICKEY 2.0.85 and MICKEY 1.0.....	112
Figure 5. 6 Levenshtein similarity test for MICKEY 2.0 and MICKEY 2.0.85 interrupted messages .....	119
Figure 5. 7 Cosine similarity between two vectors $V_1$ and $V_2$ .....	120
Figure 6. 1 Communications between mobiles and external server – FEATHER protocol .....	129
Figure 7. 1 Illustration of RFID tag basic functionality.....	150
Figure 7. 2 Relationships between the individuals in eHealth scenario.....	156
Figure 7. 3 Core components of NFSDE .....	161
Figure 7. 4 Processes for provider tag creation.....	164

Figure 7. 5 Processes for patient tag creation .....	168
Figure 7. 6 NFSDE activation and unlocking processes .....	171
Figure 7. 7 Display the patient data process .....	172
Figure 7. 8 NFSDE device software emulation .....	173

## List of Tables

Table 3. 1 Number of primitive polynomials per degree (degree 4 to 35).....	49
Table 3. 2 d-monomial test implemented on Shrinking Generator with degrees 16–19.....	58
Table 3. 3 Unique Window Size 21 for Self-Shrinking Generator counts and probability, with all weights .....	63
Table 3. 4 Unique Window Size 21 for Self-Shrinking Generator counts and probability, with weight = 5 .....	63
Table 3. 5 Total count of each Unique Window Size occurrence for degrees 4 to 24 for Self-Shrinking Generator.....	66
Table 3. 6 Finding weak polynomials with degrees 6 and 7 with different initial seed (key) .....	67
Table 3. 7 Finding weak polynomials with degree 6, 7, 12 and 14 with all initial seed (key).....	67
Table 3. 8 d-monomial with degrees 7 to 15 with full keystream string for Shrinking Generator and Self-Shrinking Generator results for comparison.....	68
Table 3. 9 Shrinking Generator exhaustive testing results for maximal monomial test for combined LFSR lengths 7 to 9.....	69
Table 3. 10 Three randomness tests for Shrinking Generator with Unique Window Size 20 .....	72
Table 4. 1 Shrinking Generator unique window size 24 model summary .....	91
Table 4. 2 Shrinking generator model results for the new neural network models, including results for degrees 20, 21, 23 and 24.....	91
Table 4. 3 Unique window size 24 chosen 10 input samples for the shrinking generator .....	92
Table 4. 4 Neural network model for a self-shrinking generator, with different unique window size degrees .....	92
Table 4. 5 Unique window size 25 for the chosen self-shrinking generator (SSG) input sample	93
Table 4. 6 The importance of independent variables for the neural network model using unique window size 24 for the self-shrinking generator.....	93
Table 5. 1 MICKEY 2.0 R_mask.....	109
Table 5. 2 MICKEY 2.0.85 R_mask.....	109
Table 5. 3 MICKEY 2.0 COMP0 .....	109
Table 5. 4 MICKEY 2.0.85 COMP0 .....	109
Table 5. 5 MICKEY 2.0 COMP1 .....	110
Table 5. 6 MICKEY 2.0.85 COMP1 .....	110
Table 5. 7 MICKEY 2.0 FBO .....	110
Table 5. 8 MICKEY 2.0.85 FB0.....	110
Table 5. 9 MICKEY 2.0 FB1 .....	111
Table 5. 10 MICKEY 2.0.85 FB1 .....	111
Table 5. 11 MICKEY 2.0.85: 410 sequences, each sequence with a length of $10^6$ bits.....	113
Table 5. 12 MICKEY 2.0: 410 sequences, each sequence with a length of $10^6$ bits.....	113
Table 5. 13 MICKEY 2.0.85: 1,350 sequences, each sequence with a length of $10^6$ bits.....	114
Table 5. 14 MICKEY 2.0 NIST test results for ciphertext with 13 MB bits length .....	115
Table 5. 15 MICKEY 2.0.85 NIST test results for ciphertext with 13 MB bits length .....	115
Table 5. 16 Improvement in the encryption speed for MICKEY 2.0.85 compared to MICKEY 2.0 .....	116
Table 5. 17 Power consumption for MICKEY 2.0.85 and other ciphers.....	117
Table 5. 18 MICKEY 2.0.85 and MICKEY 2.0 results by applying cosine similarity.....	121
Table 6. 1 Specifications of five mobile devices used to test FEATHER .....	140
Table 6. 2 Running 8 MB file 60 times and taking the average time (in seconds) for five different devices.....	141

Table 6. 3 Running 1 KB to 16 KB files and calculating time (in seconds) .....	141
Table 6. 4 Running 3 KB to 512 KB files and calculating time (in seconds) .....	141
Table 6. 5 Running 1 MB to 16 MB files and calculating time (in seconds).....	142
Table 6. 6 CLOAK and FEATHER protocols: total speed time for different files sizes.....	142
Table 7. 1 Notation description.....	154
Table 7. 2 Key generation, storage and time index update .....	169
Table 7. 3 Time for NFSDE unlock.....	174
Table 7. 4 Time to read encrypted patient data.....	175

## Abbreviations

AES	Advanced Encryption Standard
ANF	Algebraic Normal Form
AWS	Amazon Web Services
CC	Cloud Computing
CPU	Central Processing Unit
DES	Data Encryption Standard
GAE	Google App Engine
GEs	Gates Equivalents
IoT	Internet of Things
IPsec	Internet protocol security
IV	Initialisation Vector
LFSR	linear-Feedback Shift Register
LWC	Lightweight Cryptography
LWE	Lightweight Encryption
MCC	Mobile Cloud Computing
MSE	Mean Square Error
NFSDE	Near Field Secure Data Extractor
NIST	National Institute of Standards and Technology
NLFSR	Nonlinear-Feedback Shift Register
NN	Neural Network
OTP	One-Time-Pad
PII	Personal Identifying Information
PISO	Parallel-In, Serial-Out
PKI	Public Key Infrastructure
PRNG	Pseudo-Random Number Generators
ReLU	Rectified linear unit
RFID	Radio-Frequency Identification
SG	Shrinking Generator
SIPO	Serial-In, Parallel-Out
SSG	Self-Shrinking Generator
STS	Station-To-Station
TLS	Transport Layer Security
TMTO	Time–Memory Trade-Off

TRNG	Truly Random Number Generator
UID	Unique Identifier
UUID	Universally Unique Identifier
UWS	Unique Window Size
XPE	Xilinx Power Estimator

## **Abstract**

Ensuring the security of sensitive information stored on small devices has become increasingly challenging due not only to technological advances but also growth in the number of IoT devices. Currently, there is a lack of efficient encryption and cryptanalysis methods for small devices, and lightweight encryption techniques including ciphers must therefore be improved to meet security standards. The growth of radio-frequency identification (RFID) applications on small devices has also provided communication solutions which require appropriate security to ensure their integrity through authentication and authorisation. To address this security gap, this thesis develops a measurement framework by establishing the maximum order complexity method of unique window size as a vital binary sequence strength measurement. A novel neural network model is implemented to predict the unique window size to evaluate targeted ciphers' pseudo-randomness. A secure and lightweight cipher based on the well-known MICKEY 2.0, called MICKEY 2.0.85, is proposed. This cipher reduces the length of both registers from the original 100 bits each to 80 bits for both. Pseudo-randomness tests from the US National Institute of Standards and Technology were used to ensure that all usual security requirements are met. Broad cryptanalysis was also performed by testing MICKEY 2.0.85 against common attacks. The findings show that MICKEY 2.0.85 has slightly more resistance, consumes less power, occupies less space and is 23% faster in encryption speed than MICKEY 2.0. This thesis also proposes a lightweight cloud computing security protocol, FEATHER, for communications between mobile devices over insecure channels, as well as the Near Field Security Data Extractor (NFSDE) system that has an encryption protocol to provide security for RFID technologies when internet connectivity is unavailable or unreliable. These proposed security solutions are developed in an eHealth context but could be adapted for other applications using IoT, RFID and mobile cloud computing and in uses with insecure channels or unstable internet.

## **Statement of Authorship**

Except where reference is made in the text of the thesis, this thesis contains no material published elsewhere or extracted in whole or in part from a thesis submitted for the award of any other degree or diploma. No other person's work has been used without due acknowledgement in the main text of the thesis. This thesis has not been submitted for the award of any degree or diploma in any other tertiary institution.

Ahmed Alamer

12 May 2020

## **Acknowledgments**

First and foremost, I want to thank the Almighty God (Allah) for his generosity and great gratitude. May he continue to guide me in this long and interesting journey. I wish to express my deepest gratitude to the University of Tabuk, Kingdom of Saudi Arabia, for awarding me a scholarship and for providing financial and moral support throughout my studies.

I sincerely thank my expert primary supervisor, Associate Professor Ben Soh, for his ethical and high-quality support and for his ongoing encouragement including when I faced difficulties. His support and direction laid a strong foundation for my achievement. His understanding, constant support and guidance improved my research effectiveness and my publications. His experience, including teaching, supervision, research and publishing, extends to all his students from many nationalities.

I also thank Dr Ahmed Al-Ahmadi, a former student of Associate Professor Ben Soh, for suggesting that I contact him as my main supervisor, given his experience in my research field and in eHealth consulting. I also thank industry practitioner Mr David Brumbaugh for providing advice in the field of programming and for ensuring the validity of the results and application in eHealth and technological fields. I also thank Dr Lito Cruz for his advice on neural network modelling, Dr Mulla Huq for statistical modelling and Dr David Jones for advice on mobile cloud computing.

I thank my assistant supervisor, Associate Professor Andriy Olenko, for his encouragement, support and follow-up with my supervisor. I also thank Dr Rhonda Daniels for her invaluable editing advice.

I wish to extend special thanks to La Trobe University for enabling me to complete my PhD. I thank the School of Engineering and Mathematical Sciences and its wonderful staff, for providing me with the best study environment and for helping me to overcome difficulties and solve them quickly.

I would also like to thank my dear mother, who has been supporting, encouraging and guiding me since the beginning of my childhood studies. She has always encouraged me to be the best version of myself. It also gives me great pleasure to thank my dear wife,

Fawziah, the companion of my path, my life, my partner in happiness and sorrows, and my closest friend. She has been a constant aid. She has been extremely patient during my constant preoccupation with research, writing and academia. During my long and arduous journey, she has provided crucial support, and also become a mother to our daughter, Warde, and our son, Bassil. Our children are a most joyful addition to my life. I also want to thank my brothers, Ibrahim and Ali, for handling my affairs in Saudi Arabia while I studied abroad. They provided financial and moral support, which enabled me to take a sabbatical for my studies.

# List of Publications

## Published papers

1. Alamer A, Soh B, Alahmadi AH, Brumbaugh DE. Prototype device with lightweight protocol for secure RFID communication without reliable connectivity. *IEEE Access*. 2019 Nov 19;7:168337-56.

This paper forms the basis of Chapter 7 and has been updated and rewritten with new content and structured to form a thesis chapter.

2. Alamer A, Soh B, Brumbaugh DE. MICKEY 2.0. 85: A Secure and Lighter MICKEY 2.0 Cipher Variant with Improved Power Consumption for Smaller Devices in the IoT. *Symmetry*. 2020 Jan;12(1):32.

This paper forms the basis of Chapter 5, and has been updated and rewritten with a thesis structure and organisation, and with additional results and explanations.

3. Alamer A, Soh B. Design and implementation of a statistical testing framework for a lightweight stream cipher. *Engineering, Technology & Applied Science Research*. 2020 Feb 3;10(1):5132-41.

This paper forms the basis for Chapter 3, and part of Chapter 4. It has been rewritten with more results and basic contents that helps to provide clear and detailed foundations for the chapter 3.

4. Alamer A, Soh B. A new neural-network-based model for measuring the strength of a pseudorandom binary sequence. *International Journal of Advanced and Applied Sciences*. 2020 7(4): 29-38.

The paper forms the basis for Chapter 4, and has been rewritten with more results and basic contents

## Submitted paper

Alamer A, Soh B. FEATHER: a proposed lightweight protocol for mobile cloud computing security. *Engineering, Technology & Applied Science Research* (under review, May 2020).

# **Chapter 1: Introduction**

## **1.0 Chapter overview**

This chapter is an introduction for the thesis, offering an overview of thesis content and providing the chapters association and organisation. Section 1.1 presents a general introduction to security, while section 1.2 offers security background on applications requiring security to protect sensitive data. Section 1.3 provides an overview on cryptosystems, and section 1.4 discusses security challenges. In addition, section 1.5 details research objectives, aims, research questions and methodologies, while section 1.6 presents an overview of the thesis contributions to current literature in lightweight security methods and applications. Finally, section 1.7 provides the thesis structure, and section 1.8 concludes the chapter.

## **1.1 Introduction**

This thesis aims to harmonise theory and practice to make an important contribution to, and advance the field of, cryptography. Given that data transfer and sharing are essential in daily life, cryptography plays a vital role in security, privacy, authentication, and integrity.

Security and privacy are an integral part of people's lives. Sensitive personal data, including finance-related information, health records and job-related data and information, must be secured to avoid misuse and ensure it is only accessible by the data owner and authorised people. Developing strong security methods such as encryption of data is essential to the field of cryptology. Cryptanalysis is the science of analysing encryption methods that need to be updated to meet the threat of information misuse. Analysis can use either statistical methods or other mathematical methods. In this era of advanced development of communication devices and network improvements in data storage, cloud computing can facilitate the transfer of data. A specific branch of cloud computing is mobile cloud computing which makes use of mobile devices. Cloud computing, including mobile cloud computing, involves the transfer of large amounts of data, which can be sensitive. However, transfer methods are unsafe, and the data must be encrypted for

transfer and storage. There has been significant growth in the use of the internet of things (IoT), which in turn has led to growth in the use of radio-frequency identification (RFID) technology for object tracking and monitoring. As a result, a secure and valid secure cryptosystem is needed.

This thesis investigates and tests various lightweight stream ciphers as a type of cryptosystem for small devices, using different analysis methods. It introduces a new method of analysis to propose MICKEY 2.0 cipher reduced variants that aim to be suitable for small devices such as those used in IoT, mobile cloud computing and RFID tags. This thesis provides new security applications and protocols for mobile cloud computing and RFID technology that have practical use in eHealth care as a real-world example, as security of patient data is critical to avoid potentially life-threatening changes to information.

## **1.2 Background**

Both individuals and organisations need data security. Individuals want data security to protect sensitive personal data such as bank account and credit card information, while organisations want security protection for employee and business data including employment records and payments. There are two main branches of security protection: data security, which focuses on information protection; and system protection, which targets information protection within devices that contain this information, as well as protecting data transfer through networks [1].

The internet of things (IoT) [2] was established to provide solutions for data computing by devices, and transfer data over networks without the need for human interaction. The IoT connects devices with the internet for data transfer and exchange. Some of the data is sensitive and needs to be protected from unauthorised people. Cryptology uses algorithms that encrypt data with secret keys or public keys that need to be protected and shared securely. With advances in technology there is a need to improve cryptography methods to meet new demands, as devices become more widespread and networks grow and expand their coverage. Growth leads to vulnerabilities and more attacks by people who can use this sensitive data.

Cloud computing is part of the IoT facilitating an external server to store and transfer data [3]. Mobile cloud computing adds mobile devices for communication and information exchange [4], [5]. Another technology, radio frequency-identification (RFID), implements tags for information protection and tracking [6].

Encryption is an effective tool for transferring information that requires security and protection. Therefore, strong encryption systems, called cryptosystems which are sets of cryptographic algorithms applied for implementing a security service [7], must be chosen by the users and their strength must be tested. Some cryptosystems act as pseudo-random number generators (PRNG), see for example [8], and their strength can be assessed by testing the randomness of the pseudo-randomness binary sequences produced by these cryptographic systems. This thesis tests the security of cryptosystems by implementing neural network models to measure the strength of specific stream ciphers, as well as other randomness test tools. The thesis also provides security optimised methods for secure applications and tests them by improved cryptanalysis to ensure efficiency and resistance against attacks by attackers who want to reveal the information.

It is important to examine the effectiveness of the application of ciphers in current uses including mobile cloud computing [9] and IoT, especially in the field of RFID tag implementation as tags are used to secure and authenticate the users [9-11]. It is important to ensure the confidentiality and protection of information.

### **1.2.1 Cryptographic methods**

Cryptology can be defined as the science of secret and hidden information, while cryptography is specifically the science of hidden text. Cryptographic methods include encryption/decryption, one way functions like hash and digest, authentication, digital signatures, entropy and randomness, and protocols [12], [13]. This thesis provides the background for most of these primitives. A stream cipher is a binary additive cipher (as the plaintext binary sequence XORed with keystream (mod 2) for encryption). The first step in the process to identify its internal state is to use an initialisation vector (IV) and key to produce a keystream for an IV-based system, while other ciphers only use a key. The next step is to XOR the keystream – symbol by symbol – with plaintext to obtain the ciphertext.

The standard model for attacking synchronous stream ciphers, where the encryption is done one step at a time, assumes that there is a known keystream. This assumption is reasonable because the known plaintext and XOR can be used with the ciphertext to obtain the known keystream. According to good practice, the design of the cipher should be made public to enable analysis and avoid ‘security by obscurity’. Researchers can analyse their own copy of the cipher—typically in software—in any mode.

Cryptographers design the stream cipher’s internal state to produce a random keystream by using IV bits and key bits. A higher security level requires that the internal state of the encryption cipher is twice the size of the key within it. In some stream ciphers, the designer’s goal is to increase the size of the internal state in order to avoid some types of attacks like time–memory trade-off (TMTO) attacks. However, cryptanalysis has found that if the key is larger than the IV, this process will not guarantee more resistance to a TMTO. Consequently, IV bits should be greater than key bits, and the internal state should be equal to (or greater than) IV + key. This result can be found in [14], [15].

### **1.3 Cryptosystems targeted in this research**

Cryptosystems include traditional cryptography and lightweight cryptography. The focus of this thesis is lightweight cryptography as this thesis is devoted to lightweight and low-cost applications, including lightweight synchronous ciphers and lightweight asynchronous ciphers. This thesis focuses on lightweight synchronous ciphers, specifically lightweight synchronous stream ciphers [14 -17]. Lightweight cryptography that does not require extensive computation resources has been attracting interest in research and application over the past 20 years. A large number of cryptographic systems are suitable for applications that have limited capacity in terms of complex processes and small memory. These systems include Present, Clefia, LED, Trivium, Grain and MICKEY 2.0 [17-19] which are a good fit for mobile cloud computing and IoT which have small devices included in their structures.

### **1.4 Challenges and motivation**

Communication technology and information circulation is widespread, and there is an urgent need to ensure confidence and confidentiality of information dissemination through different communication channels, which are not necessarily protected. The rapid

development of mobile devices makes this an important area of study in terms of information analysis and encryption systems to ensure the best applications to help overcome cloud computing and IoT challenges.

Lightweight encryption methods are cryptosystems for small devices, and there are many of them including lightweight synchronous stream ciphers. It is important to improve these systems to be more suitable for new and emerging ultra small devices. It is also important to improve current cryptanalysis methods, which requires testing and validation of the ciphers' visibility in terms of implementation and ensuring the level of their security. Existing research still has many shortcomings and gaps requiring extensive and deeper research.

## **1.5 Research objective and aims**

The research objective is to analyse flaws in lightweight synchronous stream ciphers such as Shrinking Generator (SG) and Self-Shrinking Generator (SSG) synchronous ciphers. It adapts the MICKEY 2.0 cipher in mobile cloud computing and IoT in a secure manner. It shows how lightweight stream ciphers can be more practical compared to heavy cryptosystems such as Advanced Encryption Standard (AES) by implementing this method in mobile cloud computing and IoT applications while still maintaining security. Statistical and theoretical analysis tools such as randomness tests and neural networks are needed.

This thesis research aims to evaluate the security of targeted lightweight synchronous stream ciphers. It explores analysis methods that can be used for targeted ciphers, which can be applicable for similar ciphers. Understanding pseudo-random binary sequence behaviour will help the applications that use it to identify the flaws in those applications. This thesis adds new data analysis approaches for evaluating the security of ciphers and binary sequences, such as neural networks, which is an active research area. The results of this thesis can be applied in fields such as mobile cloud computing and IoT, with a focus on applications related to RFID tags. This thesis aims to show the importance of using an IV-based cryptosystem compared to cryptosystems that do not use an IV.

This thesis has four main goals. First, it aims to achieve a better understanding of some current lightweight synchronous stream ciphers. Second, it aims to investigate the usability of these synchronous stream ciphers in real-world applications. Third, it links the first and

second goals with mobile cloud computing to improve optimal implementation. Fourth, it links the first and second goals with RFID technology application and study of its performance to determine the most beneficial applications.

### 1.5.1 Thesis approach and research questions

Building on the three blocks of security, data analysis and applications in Figure 1.1, Figure 1.2 shows the four stages of the thesis research to address these targeted gaps as illustrated in the research stages in this section.

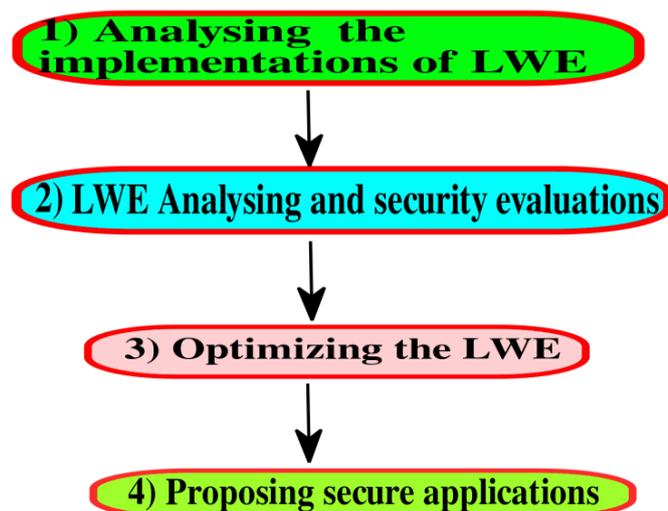


Figure 1. 1 Research stages

#### Stage 1 Analysing the implementation of lightweight ciphers

Lightweight synchronous stream ciphers provide security for devices and software with limited abilities such as computation power, energy consumption and memory limited size. Despite the evaluation of implementation of lightweight encryption methods, the research on cryptanalysis, optimising existing ciphers and proper implementation still requires more attention. The research questions are:

1. How can lightweight synchronous stream ciphers be implemented effectively to secure the transfer of data?

1.1 Based on the current gaps in proper implementation, what are the benefits of using lightweight stream ciphers?

1.2 Why is it not always efficient to use other cryptosystems such as Advanced Encryption Standard (AES)?

### **Stage 2 Analysing lightweight stream ciphers**

Identifying the weaknesses and flaws in existing lightweight synchronous stream ciphers will help optimise or replace them with more lightweight encryption methods. Although this is a very active research area, implementing new cryptanalysis methods needs further effort. The research questions are:

2. How can existing lightweight stream ciphers be analysed properly, and avoid problems in poor analysis to find weaknesses to help users decide which cryptographic algorithm should be used?

2.1 How can neural networks be applied as a prediction method for the nonlinear complexity of a binary pseudo-random sequence?

2.2. How can randomness tests be applied effectively to find weaknesses in a given stream cipher?

### **Stage 3 Providing solutions for real-world applications by optimising lightweight encryption**

Improving current lightweight encryption has not received sufficient focus, and optimisation and analysis methods, and the cost of current encryption methods, are still not optimal. There is a need to secure sensitive data, especially in small devices, at reasonable cost. The research questions are:

3. How can this study contribute to real-world applications by providing solutions to current issues in security, efficiency, cost and performance?

3.1 How can lightweight encryption ciphers be optimised, and how can lighter versions be proposed to avoid shortcomings in implementation in small devices?

3.2 How can cryptanalysis be tested to provide sufficient confidence for the proposed novel lightweight encryption cipher to ensure validity for usage?

### **Stage 4 Implementing MICKEY 2.0 and proposing secure applications**

The binary sequences generated by pseudo-random number generators need to be as pseudo-random as possible. MICKEY 2.0 is a popular lightweight encryption cipher, yet needs to be optimised for applications such as RFID and mobile cloud computing. Further

research to enhance such ciphers has benefits in security and related costs associated with implementation. As there are problems with current applications, the study provides solutions for these applications by addressing the following question:

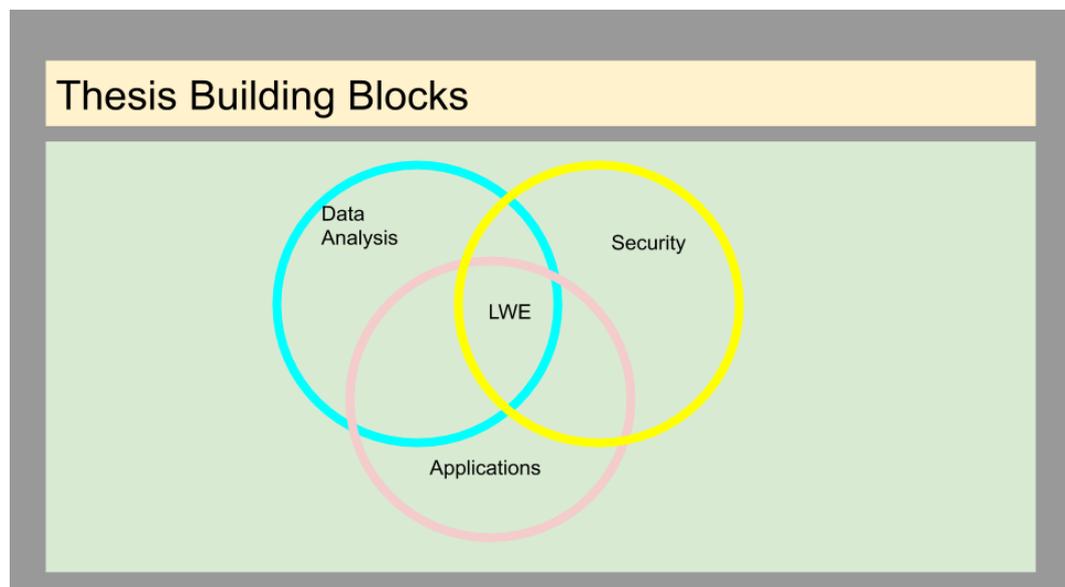
4. How can RFID, mobile cloud computing, pseudo-random binary sequence analysis and MICKEY 2.0 be connected to ensure consistency and identify contributions to multiple disciplines and different applications, and fix current implementation issues?

4.1 How can MICKEY 2.0 be implemented efficiently to secure communication between mobile devices in mobile cloud computing?

4.2 How can MICKEY 2.0 be implemented efficiently to secure communication in RFID technology devices in IoT, especially in RFID tags?

### 1.5.2 Components of the thesis

The thesis investigates lightweight synchronous stream ciphers as the main focus of general lightweight encryption methods [16] through three main blocks of security, data analysis (for cryptanalysis) and applications. The thesis assesses encryption methods and some lightweight ciphers, and optimises the best candidate and then implements them with novel security applications, as shown in Figure 1.2.



**Figure 1. 2 Thesis main blocks and overall focus**

The first block is security. The protection of information has become necessary with the proliferation of means of communication such as portable devices and other channels of

communication such as Wi-Fi, 3G, 4G and, in the near future, 5G [20], as well as the development of systems and IoT technology such as RFID, cloud computing and mobile cloud computing [21].

The second block of data analysis is important in the study and evaluation of cryptographic systems for development. This thesis uses randomness statistical tests and neural networks, a connectionist approach for computation based on the use of interconnected artificial neurons, among other methods.

The third block is application. Applications can be designed so they are highly protected in mobile cloud computing and can protect information using RFID technology in sensitive areas such as eHealth as an example.

### **1.5.2 Significance of the research and implications**

This thesis provides a better understanding of some current lightweight synchronous stream ciphers, and investigates the usability of some synchronous stream ciphers and optimises them to be suitable in real-world applications. This research helps evaluate the security of the targeted ciphers. It also explores the most effective methods that can be used for evaluation of ciphers, which can be applicable for similar ciphers. In addition, understanding pseudo-random binary sequence behaviour can help the applications that use it to identify the flaws in those applications. This research adds new data analysis approaches in terms of evaluating the security of ciphers and binary sequences, such as neural networks, which is an active research area. Lastly, the results of this research can be applied in fields such as mobile cloud computing and IoT, with a focus on applications related to RFID tags such as eHealth.

1. Lightweight encryption is important because it provides a considerable level of security that is suitable for devices with a small capacity.
2. It is important to find the flaws of the ciphers for designers to improve their cryptosystems, and for users to choose the best cryptosystem for their security needs.
3. Applying different cryptanalysis methods in this thesis helps to improve them and enables them to be applied in different cryptosystems.

4. Suitable applications need to be created using lightweight stream ciphers such as MICKEY 2.0 for implementation in mobile cloud computing and RFID tags for eHealth security.

### **1.5.3 Methodology**

This thesis uses the following analysis methods to achieve the research aims and answer the research questions:

1. The outcomes of recently published research on lightweight stream ciphers are analysed to obtain a broader understanding of the topic and extend the knowledge of IV-dependent (MICKEY and Trivium) and IV-less (SSG and SG) stream ciphers.
2. Statistical tests such as d-monomial and chi-square are applied to investigate the appearance of randomness in synchronous lightweight stream ciphers.
3. Comparison techniques are used for the statistical test results to identify the weaknesses in stream ciphers. In particular, this thesis compares SG and SSG (IV-less) and MICKEY 2.0 (IV-dependent) stream ciphers.
4. Statistical programming software such as R programming is used as a statistical tool for the chi-square test. C programming is used to design codes to interpret the algorithms for the statistical tests. Linux is used to submit the job to EC2 [21], [22] and Python programming language is used for neural network models.
5. A range of theoretical approaches are applied to cipher design. This enables the mathematical theories, formulas and propositions to be compared to provide an understanding of their relevance and applicability to the thesis aims.
6. The NIST randomness test suite is used to measure the randomness of the proposed MICKEY 2.0 cipher lighter variant, a reduced version of MICKEY 2.0 cipher, which is designed to be suitable for RFID tag applications [23].
7. The neural networks model is implemented to predict nonlinearity and complexity levels for given binary sequences.
8. Cryptanalysis and attacks are applied to show how the targeted ciphers are resistant to attacks.

## **1.6 Thesis contributions**

This section provides an overview of the results, including the results from the analysis, and the proposed security applications, to identify the thesis contributions.

### **1.6.1 Statistical tests on SG and SSG**

This thesis conducts in-depth calculations to test the effectiveness of encryption systems such as shrinking generator (SG) [24] and self-shrinking generator (SSG) [25] ciphers. It identified their strengths and weaknesses to provide insights to researchers on the protection of information that is transmitted through unsafe communication channels (see Chapter 3).

This thesis presents prediction models using neural networks [26] to measure randomised binary sequences, which are reflected in the efficiency of the cryptosystems they produce. Promising and interesting results are obtained (see Chapter 4). Implementing these models as a measuring tool can contribute to related applications in information protection systems.

### **1.6.2 MICKEY 2.0 reduced variant (MICKEY 2.0.85)**

To ensure that the encryption system is suitable for use in RFID tags, the number of gates equivalent (GE) must be as small as possible. This thesis introduces an optimised version of MICKEY 2.0 that is suitable for use in RFID tags, as well as lightweight devices. Using standard randomised tests from NIST [27] as well as other cryptanalysis methods shows the MICKEY 2.0.85 reduced variant does meet the required standards and is resistant to attacks (see Chapter 5).

### **1.6.3 Mobile cloud computing and FEATHER protocol**

This thesis introduces a protocol called FEATHER to secure information transfer between two or more mobile devices using an insecure communication channel. FEATHER depends on the following components:

- Mobile devices for communication between people, networks (e.g., Wi-Fi and 4G) and cloud servers are used to produce secure keystreams.

- The MICKEY 2.0 cipher is used to produce a secure keystream.
- A number of identification and protection parameters, including One-Time-Pad (OBP), hash function and time stamp, are added to ensure the transfer of information between mobile devices and between mobile devices and the cloud server. The MICKEY 2.0 cipher is used because it provides protection and rapid production of the keystream in the server, which enables fast communication between the server and mobile devices. This is an important feature and reduces costs. Chapter 6 presents details of the proposed FEATHER protocol design and application and the results of the tests.

#### **1.6.4 eHealth proposed cryptosystem**

RFID technology is important in healthcare [28], for example, to follow up patients and their medications and to monitor the development of patients' health. For example, patients who have state-of-health calls for permanent follow-up may be able to call an ambulance at any time. To supervise patients, patients can be given a wristband carrying an RFID tag. To follow up on their status and to verify their identity, this thesis added another dimension to protect the confidentiality of the information using the MICKEY 2.0 cipher, which was developed in this thesis and is suitable for RFID tags. This protocol can be implemented without the need for internet connectivity. This protocol secures patients' follow-up and communication using their medical records. It also ensures identity verification. In addition, to protect the confidentiality of the information, this thesis considered a reduction in the costs of use and its resistance to piracy (see Chapter 7). The MICKEY 2.0 included in the protocol was tested in a real-world application of eHealth. The protocol showed it could provide identity verification and confidentiality.

### **1.7 Thesis structure**

The thesis has eight chapters addressing the three main components of this thesis of security, data analysis and applications. Figure 1.3 shows how the eight chapters are connected as one theme, with lightweight encryption as the main focus for the thesis and the centre of these approaches.

Chapter 1 outlines the main objectives and goals of the thesis research, and presents a brief summary of the individual studies that make up the thesis. The chapter discusses the

importance of the research and its applications, as well as the motivations and challenges of the research. The chapter summarises the types of algorithms used for cryptography in this study and applications in daily life.

Chapter 2 provides the background for the thesis research in terms of the science of cryptography and its importance. It also examines previous research related to this research, as well as random tests of binary series. The chapter discusses the main and general design of the principle of cryptography from the perspective of hardware, software and theory, as well as challenges relating to the security of information and confidentiality.

Chapter 3 defines and analyses SG and SSG in terms of the principle of their operation and how to use them. Randomness based tests are used to test SG and SSG keystreams which are binary sequences, and the results are presented.

Chapter 4 continues to examine the importance of random binary sequences in the field of cryptography. It discusses the importance of the neural network model in the field of randomness prediction of this sequence, as a tool to measure and test the effective cryptographic algorithms developed. Promising and important results are achieved for the SG and SSG ciphers.

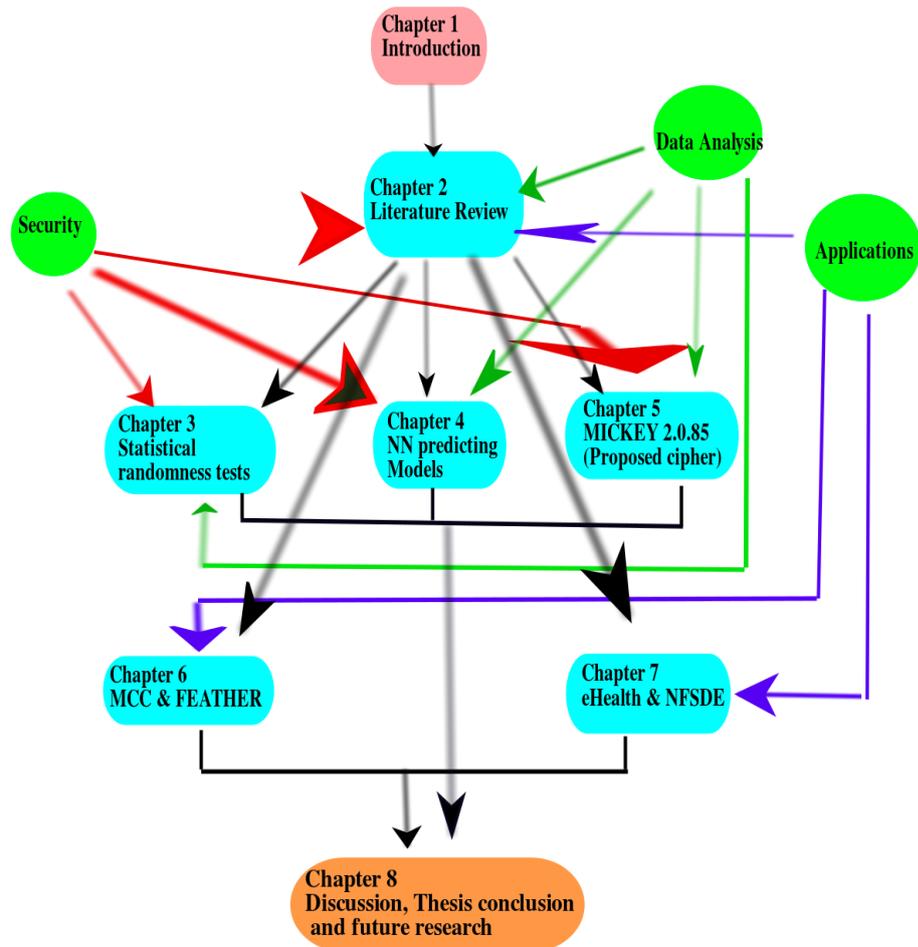
Chapter 5 introduces a secure, lighter and faster version of MICKEY 2.0, named MICKEY 2.0.85, which has 30 bits less in internal state than MICKEY 2.0. The pseudo-randomness, cryptanalysis and performance of MICKEY 2.0.85 are explained in detail.

Chapter 6 examines cloud computing, its importance and its components, with a focus on mobile cloud computing. It outlines the importance of ensuring the confidentiality of information in mobile cloud computing. The proposed FEATHER protocol includes the implementation of cipher MICKEY 2.0, and other important components are discussed. Good results are presented that can be applied to devices with limited computational capacity, memory and bandwidth.

Chapter 7 presents an important application for lightweight stream ciphers in the field of sensor networks, especially RFID tags, for which the MICKEY 2.0 cipher is applied within a secure protocol and its proposed variant to be compatible with the technology. The proposed secure protocol was implemented as a lightweight security method using a

prototype device, called Near Field Secure Data Extractor (NFSDE). The chapter presents results showing the method's effectiveness, performance and security.

Chapter 8 analyses and links the three main components – confidentiality and protection of information, analysis, and applications. It discusses the results using multiple disciplines, which represent an important contribution in the field of cryptography, and it contributes to research by examining important tools that have different applications in non-cryptographic areas. The chapter shows the thesis research may contribute to future studies by linking the science of cryptography with other fields of study and making them more interactive and productive. Thus, this research contributes to the development of the science of cryptography and security in general as well as the development of other sciences that interact with cryptography. The conclusion of the chapter summarises the results of this thesis in line with the objectives and research questions and the methods used. It outlines the contributions made by this thesis and identifies future research directions to help enhance cryptanalysis research.



**Figure 1. 3 Relationship between thesis themes and chapters**

### 1.7.1 Link between Chapters 3 and 4

Results of the statistical tests on the pseudo-randomness study of SG and SSG showed their weaknesses through testing using d-monomial tests and their derivatives, as well as a model of expectation through the multilinear regression model in Chapter 3. Further, theoretical concepts behind SG and SSG were provided in terms of how they are designed and how they work. Applying the neural network model resulted in superior predictability of pseudo-randomness, as shown in Chapter 4.

### 1.7.2 Links between Chapters 3, 4, 5 and 6

Based on the work conducted in Chapters 3 and 4, this study implements the MICKEY 2.0 cipher as an IV-based lightweight synchronous stream cipher because it is far more secure

than SG and SSG, by introducing MICKEY 2.0.85 as a lighter and secure version of MICKEY 2.0 cipher. It is implemented in a protocol called FEATHER to facilitate communication between the cloud server and mobile devices. Further, design principles and implementation details are provided, as well as the security aspect and results of the implementation.

### **1.7.3 Links between Chapters 5, 6 and 7**

This thesis uses the MICKEY 2.0 cipher and introduces a reduced variant that is suitable for RFID tag communications and implementation in Chapter 5. The ciphers' performance and designs are compared. Chapters 6 and 7 discuss enhanced applications in mobile cloud computing, IoT and wireless sensor networks.

In summary, analysing the security of lightweight stream ciphers by collecting data (e.g., about the keystream) and then analysing nonlinearity and randomness will lead to the best applications for daily life without compromising confidentiality and security. FEATHER is internet dependent and NFSDE can be used if the internet is not available.

## **1.8 Summary**

This thesis examines different analysis methods to find the flaws in lightweight synchronous stream ciphers. The findings contribute to the existing body of knowledge and enhance the field of security with a focus on lightweight encryption methods, which are suitable for small devices with limited computational power and memory.

Simulations and computations are run to measure the random appearance of the keystream of the shrinking generator and the self-shrinking generator ciphers. The thesis proposes a new approach based on designing neural network models to predict the randomness of the ciphers. This model can then be simulated for other ciphers. A security protocol called FEATHER is introduced to secure communication between two or more mobile devices with the help of a cloud server to create a secure random keystream.

The findings are applied to the eHealth security field by implementing the MICKEY 2.0 cipher and designing a reduced version that is suitable for RFID technology. This thesis adds a step to enhance secure applications for small devices to maintain their security,

performance, speed and power consumption, which are particularly important in eHealth applications.

## **Chapter 2: Literature review and background**

### **2.0 Chapter overview**

This chapter reviews the field of cryptography and identifies research gaps. Section 2.1 introduces the field, Section 2.2 presents an overview of different encryption types, Section 2.3 presents a brief mathematical foundation for cryptographic functionalities, Section 2.4 summarizes different cryptanalysis methods with types of common attacks, Section 2.5 focuses on an important branch of cryptography which is lightweight encryption, Section 2.6 introduces neural networks with proposed prediction models, Section 2.7 introduces cloud computing, Section 2.8 introduces mobile cloud computing, Section 2.9 introduces RFID technology with security, and Section 2.10 summarizes the literature review and identifies the research gaps.

### **2.1 Introduction**

Cryptography can be explained as methods for securing information from unauthorised parties and also the study and analysis of these methods. Different branches of the field include encryption, decryption, protocols and key management. Encryption is the procedure of transforming information from a clear text into a ciphertext state, and is used for encryption [8]. It is performed by a cipher that encodes messages. Information that has already been encoded can also be recovered by the cipher via a specified performance decoding method when required. Cryptographic encoding prevents unauthorised individuals from obtaining the information encrypted in the message; it therefore has significant implications for individual, national and international security [16].

The history of cryptography is reviewed in [29] and [30]. Cryptography science has expanded greatly since Julius Caesar in ancient Rome used cipher text in letters to his military officers to conceal his plans and actions. Just before World War I, in the twentieth century, the use of cryptography grew significantly, allowing across-the-board application. For the history of cryptography since before Julius Caesar till World War I please refer to [30]. Cryptographers are increasingly considering scientific ideas and concepts from software engineering and mathematics, especially when designing and using cryptographic algorithms. The strength of cryptography depends on calculations which are hard to break,

particularly due to algorithms which require considerable expense and effort to break [30], [31]. Regardless of this careful configuration, a few attempts have demonstrated that cryptographic methods can be broken. However, it is important to note that these methods have rarely been broken using functional methodologies such as analysis and, since they are exceptionally hard to break, cryptographic algorithms are considered computationally secure [17]. Changes in whole-number factorization calculations limit the generalization of algorithms' functionality needed for figuring the encryption mechanism. Therefore, this further upgraded the security offered by cryptographic methods, as quantum computing is not yet in practice [32].

The implementation of cryptographic algorithms needs to be adapted according to the targeted system, as cryptography is part of advanced hardware as a security tool, a system where signs represent discrete simple groups of logical Boolean binary operation. The cryptographic strategy empowers a hardware electronic gadget to open and close electronic gates to encode a message using binary representation. Its advanced electronic circuits have extensive clusters of logical gates, which use a Boolean rationale that works in basic gadgets. These advanced circuits have movement registers, each consisting of a course of flip-flops with a comparable clock. Every flip-flop of a movement register is joined with information and included in an arrangement chain [16]. The subsequent circuit moves by one position (bit array) where the put away bit cluster moves in the information present at its yield while the last bit in the exhibit moves out. This procedure happens at every move (shift) of clock data. The input and yield output of movement registers can be both parallel and serial and the registers are therefore arranged as SIPO (serial-in, parallel-out) and PISO (parallel-in, serial-out) [33]. SIPO and PISO therefore produce bidirectional movement from the way that they move in both bearings.

## **2.2 Encryption types**

There are two important types of cryptographic encryption design: asymmetric and symmetric cryptography [34]. Asymmetric cryptography is the encryption of a text or message using a secret key. Asymmetric encryption has two related keys, sometimes referred to as a pair of keys: a private key, in sole possession of the owner; and a second key which is public and available to whoever wishes to send private information to the owner. All messages (binary files, documents and text) are encrypted. The public key may be decoded using the same encryption technique or the equivalent private key [34], [35].

Conversely, any message encrypted using the private decryption needs the matching public key. There is minimal concern about the exchange of public keys over the internet as long as the private key is kept secret [34-36]. The main issue for asymmetrical encryption is that it is slower than symmetric encryption, thus more power is required to decrypt and encrypt messages [34-37].

In contrast, symmetric encryption uses a secret or private key consisting of a word, a string of random letters, or a number, which is used to change the content of a text or message in a specific manner [35]. This change of content may involve shifting each of the letters across a number of places. It requires the sender and recipient to know the same concealed key for encryption and decryption of the messages. The greatest threat to this type of encryption is that if the secret keys are exchanged over a large network such as the internet, the message can easily be decrypted if an attacker knows or manages to retrieve the secret key [38].

## 2.3 Mathematical foundation for encryption methods

For the encryption relay on a Boolean function  $f$  that can generate a binary sequence which is essential in cryptography as it works as keystream generated by the cipher, this sequence should look as random as possible, thus  $f$  can be considered a pseudo-random number generator (PRNG). Brief mathematical preliminaries are provided below.

### 2.3.1 Boolean function

In the binary field  $\mathbb{F}_2 = \{1, 0\}$ , let  $f$  be a Boolean function with  $n$  variables, and  $f$  mapping from  $\{1, 0\}^n$  to  $\{1, 0\}$  and can be written as:  $f : \{1, 0\}^n \rightarrow \{1, 0\}$

Let us consider the vector  $v \in \{1, 0\}^n$  if  $f(v) = 1$  then  $f$  is accepting  $v$  and the opposite when  $f(v) = 0$ .

Then  $f$  generates a sequence of binary bits. Chapter 3 explains this sequence in detail.

### 2.3.2 Pseudo-random number generators

An important component of stream ciphers used in cryptography is the generator. It produces a sequence of numbers that appear random (pseudo-random numbers) and difficult to discover which helps to encrypt a message securely [39]. This section reviews some pseudo-random number generators.

One generator relevant to cryptography is the linear feedback shift register (*LFSR*). *LFSRs* are similar to linear congruential generators, as a linear function from the previous state is used as the input bit of an *LFSR*. The exclusive-or (*XOR*) is one of the most common single bit linear functions, where the *LFSR* input is driven by some *XOR* bits of the overall shift register value. For instance, *LFSRs* can be implemented by feeding the *XOR* gates into non-sequential, different registers within. This requires placing the *XORs* inside the shift register. In *LFSR* the initial value is referred to as a ‘seed’, and the current or previous state can be used to determine the stream of values that the register produces. Because a finite number of possible states exists in the register, the *LFSR* repeats its cycle. If an *LFSR* has a well-chosen feedback function, it can produce a sequence of bits that appear randomly and have an extended cycle [39], [40].

*LFSRs* have a great number of applications including generation of pseudo-random numbers, fast digital counting, the production of pseudo-noise sequences, cryptographic use and circuit testing, among other applications. In cryptography, *LFSRs* have been used as pseudo-random number generators (PRNG) in stream ciphers [39-41]. They can be developed from simple electronic and electromechanical circuits, have long periods, and have a uniform distribution for output streams (keystream). Because of this simple structure and its linear feature, *LFSRs* are valuable targets for cryptographic attacks [42]. For instance, if the plaintext is provided with a corresponding ciphertext, an attacker can easily use cryptanalysis to recover the output of the *LFSR*. The retrieved output stream can be used by the attacker to construct a small sized *LFSR* that is capable of simulating the intended receiver. This process can be achieved by using the Berlekamp–Massey algorithm [42], [43]. The recovered output stream can be used to gather and calculate the remaining extended output stream and thus recover the whole plaintext and break the encryption.

Three methods can be used to increase the complexity associated with the *LFSR* stream cipher: using a nonlinear combination of different bits from the *LFSR* state; using a

nonlinear combination of two or more *LFSR* output bits; and performing irregular clocking of the *LFSR* as an alternating step generator. The *LFSR*-based stream cipher incorporates A5/1 and A5/2 cipher technology, which is also used in GSM cell phones. Bluetooth and shrinking generators (SG) also use the *LFSR*-based stream cipher. The A5/1 and A5/2 ciphers both have limitations [17]. For instance, there are reports of breaking the A5/2 and different studies have identified different shortcomings in the cipher [16].

Nonlinear feedback shift generators (*NLFSRs*) are also usually connected in current stream ciphers, such as smart cards and RFID. They are used in these applications because they are more impervious to cryptanalytic attacks than the *LFSRs*. For instance, the *NLFSRs* can generate  $n$ -bit at maximum length of  $2^n$  and thus can be extended to a maximum  $n$ -stage length *LFSR* resulting in a De Bruijn sequence [42], [43]. Nonlinearity can be introduced using new tools such as the evolutionary algorithm. This design reflects that the evolutionary algorithm is equipped for learning methods applying diverse operations on strings from the *LFSR*, therefore improving the viability and quality of the function [16], [17]. Other large *NLFSRs* can be established by ensuring long periods, and this is still considered to be an open problem. For example, a list of maximum-period  $n$ -bit *NLFSRs* can be generated by brute force methods for  $n < 25$  to include  $n = 25$  and  $n = 27$ . The generation of pseudo-random sequences for stream ciphers can be achieved using both *NLFSRs* and *LFSRs*. The *LFSR*-based stream cipher is an attractive target for algebraic attacks, which can be performed by secret key recovery which then allows an attacker to solve sets of chosen numbers of equations [7]. Algebraic attacks are made possible when there is misuse of multivariate relations involving key bits and output bits. This sort of attack is successful in estimating connections in low *LFSRs* polynomial degrees, which usually exist in well-known developments of stream ciphers that are resistant to all the already-known attacks. The low degree multiples of Boolean functions in algebraic attacks are the main concern in designing both the stream ciphers and block ciphers.

There is a close relationship between both the shrinking generator (SG) and the self-shrinking generator (SSG), where the SSG is a special sub-type of the SG [42-45]. As a type of pseudo-random number generator used in stream ciphers, the SG is relevant to this thesis research. The generator's system uses two *LFSRs*, which are referred to as A (output bit generating sequence), and S (the sequence that controls output). Both the A and S sequences clock so that when the S bit = 1 it implies the A bit will be in the output, and

when  $S = 0$  it means  $A$  will not be in the output. The greatest disadvantage of this system is that the rate of output varies irregularly and can therefore hint at the state of  $S$ . However, this problem is resolved through output buffering via a SSG. The SSG uses aspects of the SG and is a pseudo-random number generator. It was first introduced by Meier and Staffelbach [25], whose implementation of the generator is based on cryptography with its variants built on the *LFSR*. Because of the shrinking rule, the SSG's equipment prerequisites are very low and yet it can resist cryptanalysis. The SSG has only one *LFSR*, which has a bits length  $L$ . The *LFSR* can generate an  $m$ -sequence and its selection criteria are similar to that of the SG. For instance, the generator uses two sequences: an  $m$ -sequence ( $a_n$ ) and a controlling version of the sequence that is expressed as ( $b_n$ ). A straightforward decimal rule is also used to relate both groupings through the basic destruction standard and to set up a yield arrangement.

SG are nonlinear keystream generators consisting of two *LFSRs*. They have three main characteristics: length, their characteristic polynomial and initial state. The length of the *LFSRs* is dictated by the number of its memory cells. The characteristic polynomial of *LFSRs* is simply the feedback function. Lastly, the initial state of *LFSR* is determined by the seed or key of the cryptosystem. If the primitive polynomial acts as the characteristic polynomial of the *LFSRs* it can generate pseudo-noise sequences that have good pseudo-randomness properties, which is important for pseudo-random number generators [24]. Chapter 3 provides more discussion and analysis on both the SG and the SSG.

The computation device termed a random number generator generates a sequence of numbers or symbols without a pattern, giving the appearance of a random sequence. This randomness has many applications and has allowed the development of multiple computational methods to generate random data. Random number generators are used in gambling, statistical sampling, cryptography, randomised design, and computer simulation amongst other areas. They can be used for simulations, such as the development of Monte Carlo method simulation [12].

Random number generators can be applied in cryptography if the seed is kept secret. This allows the sender and receiver to automatically generate an identical set of numbers, in this case, the key. Random number generators can also generate pseudo-random numbers that can be applied in computer programming. Cryptography requires a high degree of randomness even though many of its operations require a low level of unpredictability [46].

For example, stronger forms of random generation have been used by physical sources like thunder noise and to generate computer-controlled adversaries associated with computer games. On the other hand, weaker forms of random generation (randomness) have been applied in hash algorithms, the creation of amortised searching, and sorting algorithms [16].

There are two main techniques for generating random numbers. The first technique measures some physical phenomena based on expected randomness and thus possible biases can be compensated for in the measuring process [47]. Some of the parameters associated with applying this technique include atmospheric noise, thermal noise, external electromagnetism and quantum phenomena. However, this technique is not practical and not implementation friendly. The second technique uses computation algorithms to produce lengthy sequences of results with a random appearance that are determined entirely by shorter initial values (seed or key) [48]. Pseudo-random generators do not rely on natural occurring entropy. However, there are cases where pseudo-random generators are periodically and naturally seeded. Due to this, their rates are not limited by external events.

Mathematical formulae can also be used to differentiate pseudo-random numbers from quasi-random numbers [49]. The basis of random number generators is deterministic computation that is not truly random. With known seed values, the output of a random number generator can be very practical. If the pseudo-random number generator is designed and implemented carefully, it can be certified for security-critical cryptographic purposes. Some practical uses of random number generators for operating systems include FreeBSD, AIX, Mac OSX and NetBSD, among others [50], [51].

## **2.4 Cryptanalysis methods and some common attacks**

Cryptanalysis, or cryptographic attack, is the analytical study of an information system to identify the hidden aspects of the system [17]. Cryptanalysis is used to breach cryptographic security systems and thus allows the analyst to access and view encrypted messages, even where the cryptographic key is not known. Cryptanalysis can also study or determine side channel attacks [17], [52]. This type of cryptanalysis focuses on the weaknesses associated with the implementation of hardware in devices and hence does not target the weaknesses in cryptographic algorithms [53]. In addition to these methods, an

important cryptographic attack introduced by Hellman [54], called time–memory–data trade-off attack (TMTO), is used in cryptanalysis. In this particular attack, the attacker attempts to create conditions related to space–time trade-off using one or multiple parameters of data, depending on the quantity of data available to the attacker in real time. The attacker balances or reduces one or multiple parameters in order to enhance other parameters of interest. The planned attack is chosen based on design failures of the ciphers and encryption to resist the established computational conditions, and is hence considered a special type of cryptanalytic attack [55]. There are two main phases of TMTO: the pre-computation (offline) phase and the actual or real time (online) phase. The pre-processing phase (pre-computation) is characterised by structural exploration of cryptosystems before computation is performed, and the findings are recorded in large tables which take considerable time to complete. Conversely, the real-time phase is characterised by cryptanalysis of real data that is obtained from a specific unknown key. The pre-computed table from the pre-processing phase is used to generate a particular specific unknown key in the shortest time possible.

The five key parameters of a TMTO attack are search space size (N), needed pre-computation phase time (P), needed real-time phase time (T), the amount of memory available to attackers (M), and the amount of real-time data available to attackers (D). TMTO attacks on block ciphers can be shown or described through the Hellman simulation [54]. For instance, where the number of potentially-employable keys is (N), the corresponding number of possible plaintexts and ciphertexts would be N. Additionally, if the data given to a block of ciphertext corresponds to a specific plaintext, then the key  $x$  for the ciphertext  $y$  would be represented as a map function of random permutation ( $f$ ) over point space (N). Where the  $f$  function is invertible, the inverse of  $f^{-1}(y) = x$  is needed. The pre-processing phase of TMTO is characterised by coverage of N and point space would be expressed by the  $m \times t$  (rectangular) matrix, which can be constructed through random iterating of function  $f$  on  $m$  starting from points N for time  $t$ . In this matrix, the leftmost columns represent the start points whereas the rightmost columns represent the endpoints. The real-time phase is characterised by the total computation that is required to determine  $f^{-1}(y)$  hence  $T = t^2$ . Since  $t$  inversion is required, one matrix can be covered by a single evaluation of  $t$  of some  $f_i$ .

Another form of time–memory–data trade-off attack is referred to as a Babbage-and-Golic [56], [57] trade-off attack, which can be successfully performed on stream ciphers. Stream

ciphers are identified by a specific number of internal states of bit generators (N) that are considered to be different from the numbers of keys, whereas the number of first pseudo-random bits (D) is produced by the generator. The attackers therefore achieve their goals by finding one bit in the internal state of the pseudo-random number generator, and then running the pseudo-random number generator to produce the remaining part of the key.

#### **2.4.1 Internal state and initialisation vector (IV)**

The internal state of a stream cipher can be identified using an initialisation vector (IV) and key to produce a keystream, then XORing the keystream symbol by symbol. The standard model for attacking synchronous ciphers assumes there is some known keystream. Cryptographers design the stream cipher's internal state to produce a random keystream, using IV and key-bits; for a higher security level, the internal state should be at least twice the size of the key within it. Cryptographers also perform statistical tests on the internal state to investigate the correlation between the IV and the key with a fixed keystream, between the keystream and the IV with a fixed key, and between a keystream and a key with fixed IV. There are a number of IV distinguisher tests such as the coverage test, the P-test and the DP-coverage test [58].

The IV in stream ciphers is used in the internal state to obtain the first output of the keystream. The IV allows the cipher to use several modes of operations with the same key but with a different IV, to produce a unique keystream. Significantly, this unique keystream removes the need to use a different key for each process. The IV also helps improve the encryption processes and provides synchronisation between the sender and receiver by producing a keystream with a randomised appearance [59].

The security of stream ciphers depends on the IV and the key initialisation, as the pseudo-random function and the pseudo-random number generator produce a keystream that has a random appearance. However, in stream ciphers without an IV, the security is provided with a pseudo-random number generator [60].

### **2.4.2 Statistical tests on stream ciphers**

Statistical tests have historically been an important tool in cryptography and are usually used to analyse the cipher. An attacker can use statistical analysis to investigate letter frequency in ciphertext written in the English alphabet (or any other alphabet) [39]. Currently, generic statistical tests, like those of the US National Institute of Standards and Technology (NIST) 800-22 suite [61], [62], are suitable for observing performance errors without considering the cipher structure and strength. Distinguishing statistical tests are more useful for analysing the internal state, especially in stream ciphers which are the focus of the thesis research.

One important statistical testing tool is hypothesis testing which is used to analyse the behaviour or character of a large population by taking a sample and using the data obtained from the sample to represent the whole population. Determining whether the sample Mean is equal to the population Mean is important in the subsequent analysis in general [63].

### **2.4.3 Statistical-based attacks on stream ciphers**

Randomness distinguishing tests are another kind of statistical test. Random sequences are also highly significant in cryptography as a pseudo-random number generator is needed to produce a random sequence. The pseudo-random number generator is easier to use for applications where the algorithm is deterministic, while in comparison the truly random number generator requires physical objects to generate the sequences, making truly random number generators hard to apply and inefficient [34]. The benefits of applying statistical tests to distinguish randomness include ease of implementation and the short time needed to run the tests.

Turan et al. [58] investigated six cipher randomness features in their study and observed statistical differences. They applied structural tests to check the relationship between the cipher key and the IV and the keystream. They identified some failures in Decim and Polar Bear which are stream ciphers, as well as failures in Frog which is a block cipher. For Decim, Turan et al. found a positive correlation between the key and keystream and a positive correlation between the IV and keystream. They also discovered some weaknesses by using the correlation between output using a fixed key and IV, similar to that already used in Decim, F-FCRS-8, Frogbit, Mag and Zk-Crypt. They also found weaknesses in diffusion in F-FCRS-8, Frogbit, Mag and Zk-Crypt. As a result of these findings, they

suggested modifying the structure of these ciphers to remove the observed weaknesses. In the studies [58] and [64] have extensions and further related statistical based analysis and results.

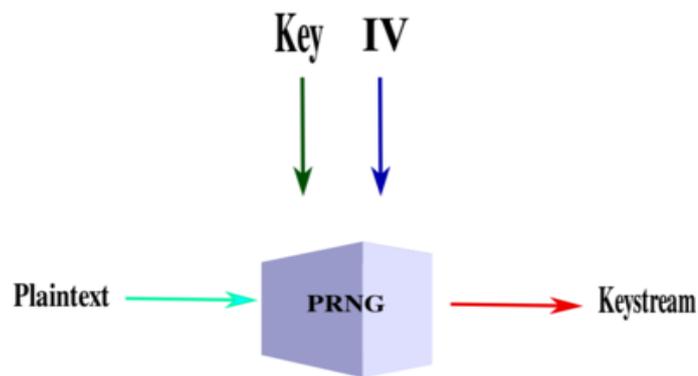
d-monomial tests as an Algebraic Normal Form-based randomness (ANF), as it combination of algebraic and statistical analysis for randomness. In addition to the distinguishing tests and hypothesis testing, the d-monomial test is another statistical tool for stream cipher analysis. It is an algebraic normal form-based test. Researchers applied the d-monomial test to some proposed stream ciphers for the eSTREAM competition [65], looking for specific parts of ciphers' output bits. These parts have bits that are less likely to have received a good mixing in the initialisation process, probably in the first or last bits of IV. By performing the experiment on eSTREAM the researchers found ciphers with insufficient mixing processes that simply failed to produce random outputs. These could be easily distinguished. Ciphers with this weakness included Mag, Frogbit and F-FCSR, all with extreme biases [43]. Englund et al. [34] used a generalised approach for the d-monomial test by using polynomial description to detect how polynomials accrue and added two new tests: the monomial distribution test and maximal degree monomial test [34].

By studying different types of TMTO cryptanalysis, suitable tests can be chosen that can be run efficiently for stream ciphers. Turan et al. presented a new suite of TMTO-based tests on selected stream ciphers using some random mapping tests with three distinguishers: coverage test, P-test and DP-coverage test. The statistical distribution of the p values varied in the Pomaranch cipher. The most significant difference (p-value of 0.05) was obtained from the encryption for Pomaranch, and thus Turan et al. repeated the test on the cipher 450 times to confirm their conclusion [58].

There has been continuous improvement in cryptanalysis methods. Randomness analysis includes Unique Window Size (UWS) and Algebraic Normal Form (ANF) based tests such as the d-monomial test, adapted to test IV-less based stream ciphers, as these tests are on SG and SSG as can be seen in Chapter 3.

### 2.4.3 Black-box attacks

Attacks have different mechanisms based on the amount of information available for an attacker. The generic distinguishing attack and the key recovery attack can be applied on stream ciphers using the internal state or taking the cipher as a black-box. In other words, the cipher as a pseudo-random number generator is treated as a black-box and then considers the element outside the black-box [58], [66]. Figure 2.1 shows the message (plaintext), the key, IV and the keystream as elements to investigate. The term “black-box” describes the cipher without considering the structure of its internal state. It is a metaphorical term, indicating that researchers are focusing on the input and output of a cipher without taking into account the internal state. A distinguishing attack aims to determine whether the sequences generated from the stream cipher appear random or not, hence identifying any flaws in the cipher. The attacker’s target in a key recovery attack is to reveal the secret key.



**Figure 2. 1 Black-box principle**

A generic distinguishing attack takes the keystream generator as a black-box, then applies a test to the keystream and studies its statistical properties, usually investigating the “random” properties of the cipher elements. The tests used in these generic applications include statistical tests and time–memory trade-off tests, amongst others. In contrast, a specific distinguishing attack studies the inner state of the stream cipher, examining the properties of the IVs and the keys used to generate the keystream [67].

A key recovery attack aims to recover the encryption key or at least some part of it. While this is a strong attack, the attacker needs more information about the cipher structure and

access to information related to input and output bits [68]. This attacker has some known plaintexts with their corresponding ciphertexts, then aims to recover the encryption key by performing some calculations on the relations between the plaintexts and ciphertexts.

In general, when applying these three types of tests which do not rely on knowing the internal structure of ciphers on a large keystream to study its random characteristics, it is possible that if the cipher fails the attackers will not recover either the key or the internal state. However, these attack methods are useful: the generic distinguishing attack can be used as a distinguisher for the keystream; the specific distinguishing attack is useful for finding the weaknesses in both the internal state (key and IV) and black-box (keystream); and the key recovery attack can help to obtain all or part of the key. Generic statistical tests can be applied on the black-box to observe weaknesses in the cipher's implementation. However, since this kind of test does not explain the cipher algorithm strength, distinguishing tests for a chosen-IV attack are needed in order to evaluate the internal state setup and allow cryptographers to determine the strength of IV initialisation [61].

## **2.5 Lightweight cryptography**

The concept of lightweight cryptography is based on enhancing the security for devices that use ciphers which are becoming smaller and smaller with technological advances. Lightweight cryptography aims to achieve the highest security levels using minimal computing power [69]. Strategies for lightweight cryptography include one-pass authenticated encryption, lightweight block ciphers, hash function and stream ciphers [61], [70]. Each of these types has features that form the basis of recent advances in security. For instance, each type of previous lightweight encryption method has a special design that enables listing of attacks and thus allows characteristic implementation of the best hardware, which makes it easy to establish and describe connections between designs. In cryptography, the reduction of the size of the device is based on design. The concept was coined by Saarinen and Engels and was expressed as lightweight primitive [12]. This expression led to the generation of algorithm measurement being referred to as "weighting a primitive" [71], which, in turn, has led to the definition of algorithmic weight. The weight of an algorithm is defined as the quantity, considering both space and time, of resources required to run it. Two distinct ways can be used to measure the weight of a primitive: the software context and the hardware context. If both contexts are considered, software lightweightness and hardware lightweightness imply vastly different things [12]. The only

connection that demonstrates any interrelation between these two contexts is power consumption.

Lightweight cryptography involves the implementation of types of cryptographic algorithms in constrained environments that include RFID tags, sensors, healthcare devices and contactless smart cards. The properties of lightweight cryptography as described in ISO/IEC 29192 are expressed according to their target platforms [70]. The hardware context of lightweight cryptography is measured and evaluated according to chip size and/or energy consumption. Conversely, the software context of lightweight cryptography is expressed according to smaller code and/or size of RAM. Lightweight primitives are better than the conventional primitives currently being used in internet security protocols such as IP security (IPsec) and Transport Layer Security (TLS) [70], [72]. Lightweight encryption (LWE) has been proven to provide an adequate level of security though it does have security–efficiency trade-offs, considering that lightweight encryption does not need much device space and, if implemented properly, will be sufficient to provide the desired security.

The key symmetric cryptographic algorithms include block ciphers, stream ciphers and hash functions [35]. Many of the block ciphers that have been recently proposed with lightweight properties are inspired by the Advanced Encryption Standard (AES). Some of these block ciphers include CLEFIA and PRESENT. In advanced studies, the block ciphers have been confirmed as having sufficient security and implementation. Stream ciphers with lightweight properties include ECRYPT II, eSTREAM (which has seven algorithms), Grain v1, MICKEY 0.2, and Trivium. The hash functions include the NIST (SHA-3). Most of the hash functions are general purpose and thus do not have lightweight properties and are considered too underdeveloped to adopt. However, theoretical concepts indicate that constructing a lightweight hash function is possible because the concepts are based on lightweight block ciphers.

There is still strong demand to create new lightweight encryption, and optimise current lightweight encryption methods to fit into new hardware, which is becoming increasingly smaller over time.

## **2.6 Neural network predicting models**

The neural network model is important for science, with rapidly increasing performance and fields of applications [73-76]. A neural network is a mathematical based tool that mimics the human brain of information processing, implementing different disciplines of mathematics such as algebra, linear algebra and calculus, as well as statistics and probability [76-80], and is thus a powerful predictive and learning tool. As neural networks are an important predictive tool and a tool which learns from the data entered, they are important for confidentiality and reliability of information [81], [82].

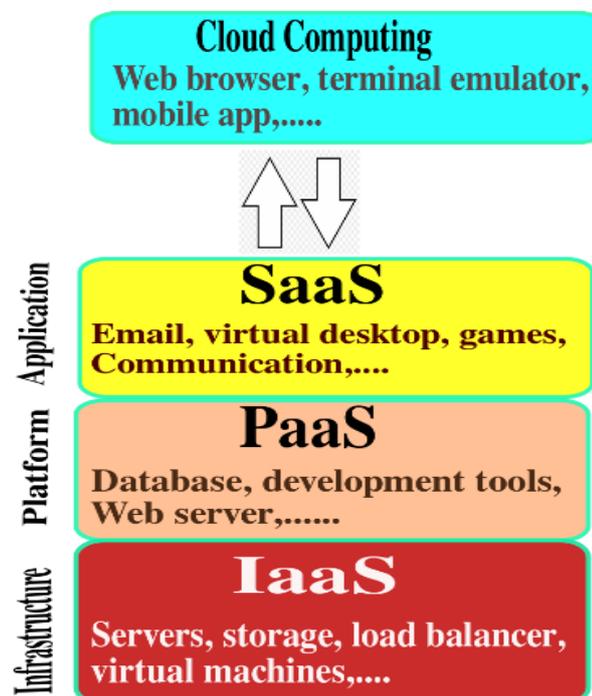
Neural networks have been used to select cryptographic keys among other fields of cryptography [83], [84]. However, use is still limited in the current literature, especially in the field of random prediction, and measuring the strength of a binary series and testing its pseudo-randomness. This thesis research provides a study on neural networks and demonstrates their effectiveness as a measurement tool, including nonlinear complexity strength for these binary sequences which result from the encryption systems to be tested. In the future, neural networks can be circulated on most encryption systems.

Pseudo-randomness tests and evaluation tools are needed for use in new methods. Thus this thesis introduces neural network based predicting models to be more effective randomness measurement tools, with a higher accuracy than the existing statistical methods.

## **2.7 Cloud computing**

Cloud computing (CC) stores and processes information in the cloud to protect information from risks to individual computers. It is becoming increasingly popular and is used when connecting a device such as a computer or smartphone through an internet provider. It also provides better information sharing. It also provides a significant amount of protection as it can only be accessed by authorised persons (who are allowed and need to guarantee access to private information), remotely from any location, hence the security needs to be in place for system and data protection [85]. Mobile cloud computing, which combines cloud computing with mobile devices and wireless channels to provide solutions for mobile devices through web apps, is discussed in the following section.

Cloud computing has several components including software, platforms and infrastructure as a service (SaaS, PaaS and IaaS). SaaS is Software as a Service such as email and a virtual desktop with software applications the user can access. PaaS is a Platform as a Service, and the environments required for users such as the operating system needed for communication, such as a web server. IaaS is Infrastructure as a Service, where the service provider has infrastructure environments such as data center storage. Cloud clients include applications used by clients through their devices such as web browsers and mobile apps. Figure 2.2 shows cloud computing components and layers [86].



**Figure 2. 2 Cloud computing structure and layers**

Cloud services can be accessed at any time and at any place. There are many cloud computing providers including EC2 by Amazon [87], Azure provided by Microsoft [88] and Google App Engine (GAE) [89].

Cloud computing can reduce costs for organisations and individuals by saving money on infrastructure and maintenance. In addition, cloud computing provides flexibility as clients use the cloud resources as needed such as storage size, memory and number of central processing units (CPUs), and can customise resources for new tasks. Thus, clients are not attached to specific hardware forever. Clients do not need to worry about support as this is the task of the cloud provider [90]. All these advantages made cloud computing an

important and increasingly popular solution for customers, as it is difficult for them to individually invest the huge amounts of money and effort required to gain the benefits provided by large cloud computing companies.

Cloud computing requires strong security. To use cloud computing resources in a secure manner, clients need to understand the cloud security parameters for better implementation. Cloud providers need to understand and address clients' needs for high security. Therefore, cloud computing presents great security challenges including authorisation and the authentication of the client [91].

Researchers [91] presented a quantitative framework for the current security challenges in practical cloud computing. They classified security concerns based on cloud architecture, and individual needs and threats that can damage the cloud components, and provided methodological solutions to mitigate these issues.

Cloud computing is an important tool for medical data storage and management, and medical data is considered to require the highest security. Study [92] provide a framework to secure medical data, by ensuring the authorisation and authentication in IoT devices used in cloud computing, which supports eHealth applications by ensuring the privacy and security for such sensitive data. As it is important that access to medical data is only granted to authorised medical practitioners, another study [93] provides a timing searchable encryption technique for securing IoT applications.

Current applications that depend heavily on security such as the blockchain can benefit from the improvement of lightweight encryption, as it needs a secure network that benefits from IoT technology including cloud computing and mobile cloud computing [94].

## **2.8 Mobile cloud computing: applications, security and current challenges**

Mobile cloud computing combines cloud computing with mobile devices and wireless channels to provide solutions for mobile devices through web apps [95]. It is a large and rapidly growing field of cloud computing, particularly with the recent increase in the use of mobile devices and forecast growth. The number of devices worldwide is expected to reach 38 billion by 2020 [96]. Improvements in communication networks such as Wi-Fi,

4G and the imminent 5G, as well as the continuous improvement in cloud computing and in IoT in general, will be reflected in improvements and increasing use of mobile cloud computing and its applications such as communications. Mobile cloud computing faces similar challenges as cloud computing such as security, and encryption, authentication and authorisation methods need to be improved and adapted for mobile cloud computing.

Mobile cloud computing offers solutions for transferring data, information and storage. It is also important to offload extensive computation to the cloud, to overcome obstacles for small devices such as battery life, to save energy, and to overcome short bandwidth and wireless connectivity problems [97].

Mobile cloud computing enables mobile device users to use more applications connected to the cloud such as social communication apps (Facebook, twitter, etc), and positioning apps (Google Maps) and makes communications and business easier and faster [95], such as banking using smartphone apps, sharing media and using devices for e-learning and e-commerce.

It is vital to address security in both cloud computing and mobile cloud computing. However, most of the current security protocols require extensive operations that consume small devices' limited resources. Hence, there is a strong need to establish a lightweight security protocol tailored to provide the security required for such small devices [98].

There are many research directions to address mobile cloud computing security, such as using well known encryption methods such as RSA, MD5 [99] and ECC [5]. However, there are new approaches to take advantage of improvements in machine learning and neural networks as in [100] as they provide two layers for authentication using a virtual machine for intrusion detection. Nevertheless, as they focus on the authentication part, neural networks can be used in other security applications such as a public key exchange [101]. Therefore, more extensive research is required on neural networks to introduce more security tools for mobile cloud computing.

To address the gap in lightweight encryption to secure mobile cloud computing applications, the thesis introduces the FEATHER protocol. FEATHER has better encryption speed and consumes less battery power than existing security protocols.

## 2.9 RFID technology

RFID technology uses radio wave frequency to transfer data, and is an essential part of IoT. RFID technology has revolutionised the way objects such as people, animals and goods can be tracked and identified through RFID tags [6]. RFID enables data to be transferred without direct contact, like contactless credit cards [6]. Ongoing improvements in designing smaller and more efficient RFID tags will help adapt this technology for more and efficient applications [102]. It is essential that RFID tags have high security [103] as authentication and authorisation are needed for most applications. From a lightweight hardware perspective, lightweight encryption is the best way to secure RFID components [104]. This thesis uses lightweight encryption within a security protocol, which works in a proposed prototype device that does not need the internet to be present.

Implementing a system and security protocol for RFID technology is challenging, as it is a very constrained hardware environment, with low computation ability. There is a large demand to have this security protocol when there is no internet connectivity, which is an even larger gap as most security protocols are internet dependent. To address these gaps, the thesis introduces a lightweight security cryptosystem that consists of a lightweight secure protocol and lightweight prototype device to implement the protocol, named Near Field Secure Data Extractor (NFSDE).

## 2.10 Summary

Cryptography has a long history of use for encryption. There are two important types of cryptographic designs: asymmetric and symmetric cryptography. The effectiveness of modern cryptography involves the design of cryptographic algorithms based on assumptions of computational hardness. Linear feedback shift registers (*LFSRs*) have been used as generators of random numbers in stream ciphers. A5/1 and A5/2 are examples of *LFSR*-based stream ciphers commonly used in GSM cell phones, whereas E0 ciphers are *LFSR*-based stream ciphers used in Bluetooth. Both the A5/1 and A5/2 have various disadvantages for cryptanalysis. However, in general, the linearity of *LFSR* facilitates easy cryptanalysis that leads to attacks. One of the most common cryptographic attacks is the time–memory–data trade-off attack, which involves an attacker attempting to create conditions related to a space–time trade-off, using one or multiple parameters of data available in real time.

The shrinking generator, a type of pseudo-random number generator, is used in stream ciphers. This system uses two *LFSRs*: the A sequence, which generates output bits; and the S sequence, which controls output. Both the A and S sequences clock so that when the S bit is at 1, the A bit is the output, and vice versa. Random number generators can be applied in cryptography if the seed is kept secret. They allow both the sender and receiver to obtain the cipher key, by automatically generating the same set of numbers. These random number generators can also generate pseudo-random numbers that can be applied in computer programming. Finally, random number generators can be used to develop simulations such as Monte Carlo method simulations.

As the pseudo-randomness of binary sequences generated by pseudo-random number generators is an essential part of any security protocol, cryptanalysis methods including mathematical and statistical analysis can be used to ensure the encryption methods are valid for implementation. Improving and analysing existing cryptanalysis methods is a critical, active and ongoing research area.

Pseudo-random number generators need to be tested in terms of design. One test method is the black-box method by taking care of binary (key/IV) input and output (keystream) without considering the internal cipher structure, giving generalisation, so the concern is the pseudo-randomness of binary sequences produced by the pseudo-random number generator which can be applied to any cipher, either IV-based or IV-less ciphers. Therefore Algebraic Normal Form (ANF) based tests and neural networks are important measurement and testing tools if implemented and designed correctly for testing.

Improving and optimising existing successful cryptosystems still suffers from a research gap, particularly for newer and growing users in mobile cloud computing and for small devices. Using lightweight cryptosystems in field mobile cloud computing requires extensive and in depth research. There is a lack of research on optimised lightweight cryptosystems in the field of RFID technology and applications such as in eHealth, and in the role of these cryptosystems in the case of unreliable internet connectivity.

This thesis recognises and addresses the current literature including the security challenges of applications such as the IoT due to the growing use and need for research in mobile

cloud computing and RFID technology to continue to enhance this field which is dramatically changing communication and information transfer.

In summary, it is necessary to implement and analyse different lightweight synchronous stream ciphers in real-life IoT applications, including RFID technology and mobile cloud computing, to enhance security in important and emerging fields such as eHealth, elearning and blockchain. Therefore, better cryptanalysis methods are required, and optimisation of lightweight cryptosystems requires in depth research. There is a need to combine data analysis, security and application with efficient lightweight protocols to overcome the research gaps in this area. In the following chapters, this thesis presents solutions for these gaps including new randomness tests in Chapter 3, proposed neural network based prediction models in Chapter 4, the proposed MICKEY 2.0.85 as a new optimised version of MICKEY 2.0 in Chapter 5, the FEATHER security protocol for mobile cloud computing in Chapter 6, and NFSDE for securing IoT based devices in eHealth in Chapter 7.

## Chapter 3: Randomness tests on synchronous lightweight stream ciphers

### 3.0 Chapter overview

This chapter offers discussion about randomness tests, which were adapted and implemented using the shrinking generator and self-shrinking generator. Section 3.2 provides the mathematical basis, Section 3.3 discusses synchronous stream ciphers, Section 3.4 discusses self-synchronous stream ciphers, Section 3.5 discusses IV-less stream ciphers, Section 3.6 discusses shift register, Section 3.7 discusses Unique Window Size, Section 3.8 discusses the  $d$ -monomial test and Section 3.9 discusses the shrinking generator, Section 3.10 discusses the self-shrinking generator, Section 3.11 provides comparison between shrinking generator and self-shrinking generator results, Section 3.12 discusses other form of  $d$ -monomial test and results, Section 3.13 discusses the data distribution, and introduces the statistical modelling for predicting UWS, and Section 3.14 concludes the chapter.

### 3.1 Introduction

The chapter provides an explanation of IV-less synchronous stream ciphers, with a focus on the shrinking generator and self-shrinking generator. This chapter presents the results found in finding the flaws with lightweight stream ciphers. It uses techniques that implement randomness with statistical tests including pseudo-randomness generators, primitive polynomials and Unique Window Size ( $UWS$ ) as a nonlinear complexity measurement tool. The Algebraic Normal Form ( $ANF$ ) based test is used for the cipher keystreams in order to find any biases. A multilinear regression model is used to predict the  $UWS$ , with investigations for the best fit distribution for the  $UWS$ . This chapter emphasises the randomness tests method as a cipher strength measurement that identifies how strong the encryption algorithms are. This enables the users of these cryptographic methods to evaluate the ciphers for themselves and avoid poor implementations.

## 3.2 Mathematical basis

This section provides mathematical preliminaries required for the foundations of the tests used for cipher evaluation in terms of their strength. The aim is to aid the explanation of the chapter content and tools used for cipher evaluations. It explains binary fields which are the basic foundation stone of the arithmetic of cryptographic algorithms, *ANF* based tests, which combine the algebraic and statistical approach as explained in this chapter, as well as statistical properties required by a binary sequence to avoid any undesirable biases, and the regression statistical based model.

### 3.2.1 Finite field

**Definition 3.1:** Finite field can be defined as a set that accepts addition, subtraction, multiplication and division.

**Theorem 3.1:** Assume  $\mathbb{F}$  is a field and  $z \bmod q$  is defined, where  $z$  are positive integer numbers, and  $q$  is a prime number.  $\mathbb{F}$  is considered finite only if the number of the elements is  $q^z$ , and  $\mathbb{F}$  is called a binary field if  $q = 2$ . The binary field is the basic element of the mathematical operation of encryption tools such as Linear Shift Register (*LFSRs*) For more information about the field and its application see [105].

### 3.2.2 Algebraic normal form

*ANF* is a Boolean representation for the function, as the keystream sequence needs to be converted to its *ANF* representation. An explanation for *ANF* is as follows:

**Definition 3.2:** If we consider  $f$  is a map from  $\mathbb{F}_2^n$  with  $n$  binary input bits into one output bit in  $\mathbb{F}_2$ , the *ANF* representation of  $f$  is:

$$f(x_1, x_2, \dots, x_n) = \sum_{I \subset \{1, \dots, n\}} a_I x^I, \quad x_I = \prod_{i \in I} x_i, \quad a_I \in \mathbb{F}_2.$$

The logical Boolean operations which can be performed in *ANF* are *XOR*, *AND* and *NOT*, but the *ANF* tests only consider the *XOR* operation.

### 3.2.3 Truth table

The truth table is a mathematical table used specifically in Boolean algebra which is suggested to calculate functional values arguments with logical expressions, each with a set of values according to the Boolean variables taken. In particular, the truth table can be used to see if the expression is given true at all input values or not and the possible values are true or false in Boolean algebra are 1s or 0s, hence creating a table that can determine if the given argument is valid.

**Definition 3.3:** Assume we have a function  $f$  and the truth table for a binary sequence  $(f(a_0), f(a_1), \dots, f(a_{2^n-1}))$ , where  $f(a_0), f(a_1), \dots$  has a binary output.

### 3.2.4 Möbius Transform

In this study Möbius Transform is used to convert Boolean function  $f$  from the truth table representation to its *ANF* representation in order to calculate the  $d$ -monomial (see the explanation for  $d$ -monomial test in Section 3.8) then to apply the  $\chi^2$ -test. For the relation between Möbius Transform and *ANF*, see [106]. If  $t$  is the characteristic function of coefficients of the function  $f$ , then  $t = \mu(f)$ , where  $\mu$  is the Möbius Transform.

### 3.2.5 Hypothesis testing

When the behaviour or character of a large population (with size  $N$ ) is the subject of study, scientists often use hypothesis testing on a subset of that population. This subset, known as a sample of the population  $n$  ( $n \subset N$ ), is chosen, and this data is taken to represent the whole population for the purpose of the analysis. Because the experimental results are used to make inferences from the sample, it is important to ensure that any findings that are made from the sample can be generalised to a high degree of validity for the population [106], [107]. For example, it could be important to determine if the sample Mean approximates the population Mean or two populations have a similar mean based on two samples from these populations. Since the hypothesis test can provide this information, it is an important tool in scientific research.

### **Null hypothesis $H_0$**

In hypothesis testing, the null hypothesis suggests that the determined population and the tested sample are similar. If there is an observed difference, this may be due to experimental error (in sample selection, or extraneous variables), or due to chance. If the null hypothesis is confirmed, there is no advantage to modifying the testing procedure.

### **Alternative hypothesis $H_1$**

In the hypothesis test, the alternative hypothesis is accepted if the null hypothesis is rejected. Therefore,  $H_1$  is the value which differs from the  $H_0$  value.

Type I error occurs when  $H_0$  is rejected while it is true, while Type II error occurs if  $H_0$  is accepted while it is false and the  $H_1$  is true.

### **P-value in hypothesis testing**

The  $p$ -value is a fundamental statistical term represented by a number that is used to evaluate statistical measurements. The  $p$ -value indicates the probability the null hypothesis is rejected if the study hypothesis is true. That is, the observed effect is due to the changes in the model, and not due to experimental error or chance alone.

$p$ -value indications can be classified as follows [106], [107]:

$p \leq 0.01$ : There is a very valid indication to reject  $H_0$ .

$0.01 \leq p \leq 0.05$ : There is a valid indication to reject  $H_0$ .

$0.05 \leq p \leq 0.1$ : The evidence to reject  $H_0$  is very weak.

$p \geq 0.1$ : No indication to support  $H_0$  rejection.

A  $p$ -value of  $p < 0.05$  is an acceptable level for a sample, as it shows that the study's hypothesis is true against the null hypothesis.

### **3.2.6 Chi-square test**

A chi-square test, written as ( $\chi^2$ ), is a statistical test used for random variables and the hypothesis method which can be applied to samples which can represent a larger distribution.

**Definition 3.4** [108]: The  $\chi^2$  test of Goodness of fit is used to find if it is possible to reject  $H_0$ .

The formula for the  $\chi^2$  test is:

$$\chi^2 = \sum_{k=1}^n \frac{(O_k - E_k)^2}{E_k}$$

where  $O_k$  is the observed value and  $E_k$  is the expected value.

### 3.2.7 Balance theory

The keystream output should be balanced to avoid any bias which may lead to an attack. By using a function which can give the appearance of an unbiased keystream, the Boolean function can be considered balanced when its output with probability is close to 50% for 1s in the keystream. The balanced Boolean function is essential to avoid a biased keystream appearance [109]. The randomness test will reveal any bias in the keystream, and detect if the keystream was unbalanced.

### 3.2.8 Solomon Golomb for MLS

Maximum Length Sequence (sequence with length  $X^m - 1$ ) is essential in order to guarantee pseudo-randomness. When using the pseudo-random number generator it is necessary to generate such a sequence [106], [107], [110]. Golomb [111] states the properties required for MLS as the following:

#### a) Balance property

In a binary sequence the number of 0s and 1s should be approximately equal. If there is a sequence with maximum length  $X^m - 1$  there are  $X^{m-1}$  number of 1s and  $X^{m-1} - 1$  number of 0s, so the number of 1s = the number of zeros + 1 as the term in the sequence contains only zeros does not exist.

#### b) Run property

If there is a sequence  $X$ , we can consider the runs is a subsequence containing consecutive 1s or 0s. Should be as Solomon Golomb rules for  $X$  which started the runs of '1' or '0' should adhere to the following properties.

Consider  $X$  a binary sequence: 50% of the runs with length 1, 25% of the runs with length 2, 12.25% of the runs with length 3, etc.

For example, 1 is run of length 1, 0 is run with length 1, 11 is run with length 2, 00 is run with length 2, etc.

### c) Correlation properties

The binary sequence has a good correlation property as the coefficients are either 1 or 0 only. For a statistical explanation for the binary sequence and Solomon Golomb for MLS correlation property, see [111].

### 3.2.9 Linear complexity and nonlinear complexity

Assume we have an  $LFSR$  with length  $l$ , that generates a binary sequence. If the  $l$  is the smallest length, then  $LFSR$  is the shortest  $l$  which can generate such a sequence, that is the linear complexity is  $l$ . On the other hand, nonlinear complexity is similar with the difference that  $LFSR$  is replaced by the  $NLFSR$  with the shortest length  $l$  that can generate a given binary sequence [112]. The linear complexity was defined for both  $SG$  and  $SSG$  as seen in this chapter.

### 3.2.10 Maximum order complexity

Maximum order complexity of a given binary sequence can be defined as the degree of the shortest  $NLFSR$ , which generates this sequence. For more explanation about maximum order complexity, see [113], [114]. The Unique Window Size ( $UWS$ ) is a type of maximum order complexity pseudo-randomness test, as demonstrated in this chapter.

## 3.3 Synchronous stream ciphers

In synchronous stream ciphers, the internal state is updated independently from the ciphertext and plaintext, so that the updated state depends on the previous state and the initial state uses the key as a seed [16]. One advantage of synchronous stream ciphers is that if errors occur in a bit or in ciphertext, this will affect only one bit of plaintext after decryption. Another advantage of this kind of stream cipher is to give users more speed, which is important when work is time-consuming [115]. On the other hand, there are some disadvantages of synchronous stream ciphers. Attacks can happen by insertion or deletion,

which may reveal the plaintext [115]. Additionally, synchronisation between the sender and recipient should be maintained at all times as this process will accrue mostly over a noisy channel and this may cause loss of some bits. Users should therefore employ different methods of resynchronisation depending on the protocol used in every process. Examples of synchronous stream ciphers from the eSTREAM project [65] include Trivium [116] and Grain [117] ciphers among others (see Chapter 5) and the MICKEY 2.0 cipher (see Chapter 6). Other examples are considered. This chapter focuses on the shrinking generator and the self-shrinking generator. Figure 3.1 summarises the general design principles for this kind of stream cipher.

**Definition 3.5:** Let  $z_i$  be a keystream output,  $i = 0, 1, 2, \dots$

Let  $s_i$  be the state at time  $t$ , with  $s_0$  being the initial state. Then for each iteration  $i = 1, 2, \dots$  we can describe the process as follows:

$$k_i = L_k(s_i) \quad (3.1)$$

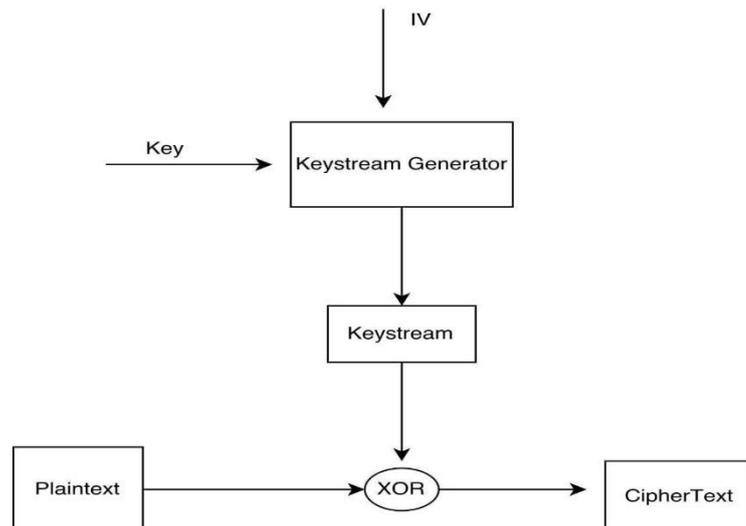
$k_i$  is the keystream function

$$c_i = g(k_i, m_i) \quad (3.2)$$

$c_i$  is the ciphertext,  $g$  is the *XOR* operation and  $m_i$  is the message.

$$s_{i+1} = f_k(s_i) \quad (3.3)$$

$s_{i+1}$  is the update function.



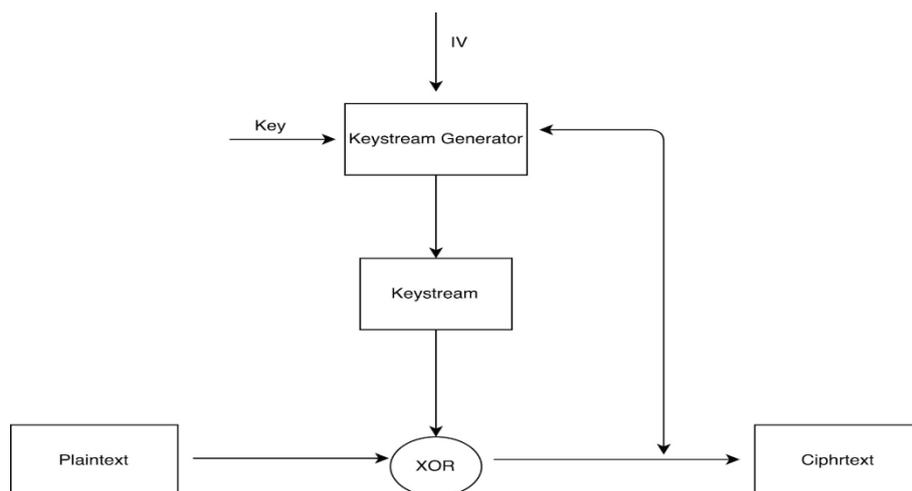
**Figure 3. 1 Synchronous stream ciphers general design**

As there is no use of bits from plaintext or ciphertext, the keystream is *XORed* with the plaintext (message) to get the ciphertext in the encryption process, while at the decryption stage the ciphertext is *XORed* with a keystream according to the cipher algorithm used to obtain the original message.

### 3.4 Self-synchronous stream ciphers

In this kind of stream cipher, the internal state is updated depending on the ciphertext. The advantage of using self-synchronous stream ciphers is that there is no need for synchronisation all the time between the sender and receiver [118]. One disadvantage of self-synchronous stream ciphers is that if errors accrue in the ciphertext during the transmission phase, this will affect  $n$  bits from the plaintext after decryption. Furthermore, if errors accrue in the ciphertext, they can be detected, which in turn can lead to an attack.

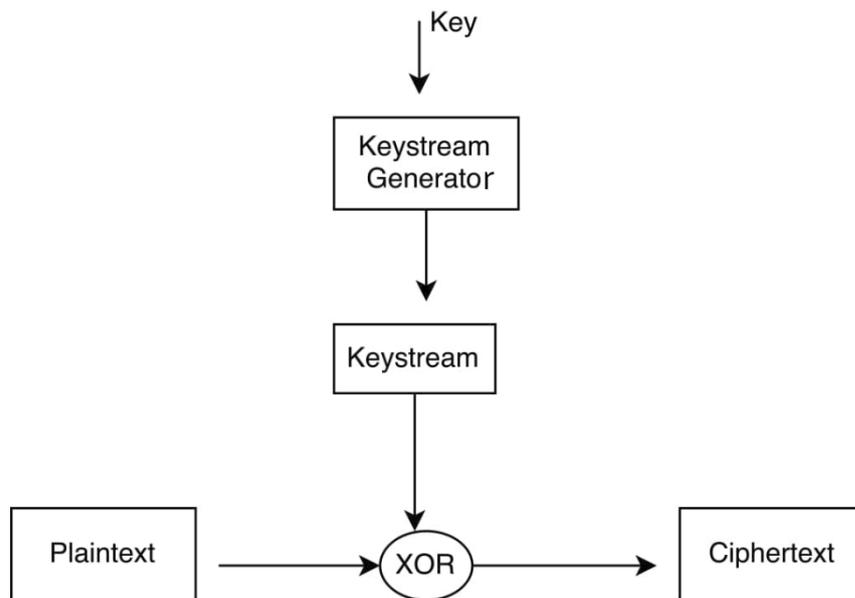
A block cipher working in cipher-feedback mode (CFB) can be considered a self-synchronous stream cipher. Figure 3.2 summarises the general design principles for this type of stream cipher. To describe the operations within self-synchronous stream ciphers, refer to equations 3.1, 3.2 and 3.3 in Section 3.3: the change will be  $s_0 = (c_{-t}, \dots, c_{-1})$  in the initial state and  $s_{i+1} = (c_{i+1-t}, \dots, c_i)$  in the update state. In these conditions, the change will be in equations 3.1 to 3.3 [16].



**Figure 3. 2 Self-synchronous stream ciphers general design**

### 3.5 IV-less stream ciphers

The loading phase uses a key (which should be kept secret), using the internal cipher function to generate the keystream, as shown in Figure 3.3. The difference with IV-based stream ciphers (ciphers that use IV) is using an IV which is mixed with a key to create more confusion, making it harder to solve. This chapter examines the shrinking generator and self-shrinking generator as IV-less stream ciphers, while Chapter 5 discusses MICKEY family ciphers as IV-based stream ciphers.



**Figure 3. 3 IV-less stream ciphers**

The truly random number generator (*TRNG*) is used to produce truly random sequences of bits. This could be a physical object such as mouse movement, or tossing a coin  $n$  times, that if repeated will be impossible or very unlikely to produce the same sequence of bits.

The pseudo-random number generator (*PRNG*) is an algorithm used to generate sequences of bits with random appearance and similar properties as the *TRNG* [8], [118]. There is a need for the pseudo-random number generator as it is easy to study and implement, whereas the truly random number generator requires a physical source and connection to hardware in order to generate a truly random bits sequence. The *TRNG* can also be very costly. One example of a pseudo-random number generator is a primitive polynomial used on *LFSR*. The pseudo-random number generator can be applied where randomness is needed, such as in gambling machines where randomness is needed to make it hard to find

correlations within the sequence bits and therefore make the machines more resistant to prediction of their outcomes.

### 3.6 Linear Feedback Shift Register (LFSR)

All bits on *LFSR* can be represented as a linear function from their previous states' bits. Let us consider the primitive polynomial which has the degree 5. If  $f(x) = x^5 + x^4 + x^3 + x + 1$ , then the *LFSR* with this polynomial output sequence bits are as follows: 00001110011011111010001001010110000111001101111101000100101011. Here (00001) is used as a seed (key).

The *LFSR* function is initialised with bits called a seed (also called the key) and then shifted according to the function which is used. An  $n$ -bit loaded into the shift register will have  $2^n - 1$  values with random appearance. We can obtain the maximal length if a primitive polynomial is used, discussed at the end of this section. *LFSR* quickly produces the output as it has very few logical arguments using *XOR* gates and the bits controlling the input bits called taps. The positions of taps will influence the output bits so that the tap is positioning *XOR* operations to be performed on certain places to get the sequence of bits.

Advantages of *LFSR* include clear algebraic analysis and easy hardware implementation, and the ability to produce sequences with long periods. The advantages of using the *NLFSR* are similar to those of Trivium and Grain ciphers: hardware, high linear complexity [105], and harder to predict the sequence generated by it. It is hard to predict the period with *LFSRs*, and statistical properties are hard to analyse [119]. The structure and properties of *NLFSR* are discussed in Chapter 5 with the use of the MICKEY 2.0 cipher.

#### 3.6.1 Primitive polynomials and *LFSR*

Primitive polynomials are an essential part of *LFSR* as the primitive polynomial will produce a sequence of bits that has a maximal length  $2^n - 1$ . The number of binary primitive polynomials of degree  $d$  of (*LFSR*) is given by  $\phi(2^d - 1)/d$  where  $\phi$  is the

Euler totient function. Table 3.1 lists the number of all primitive polynomials for Boolean functions with degrees from 4 to 35.

**Table 3. 1 Number of primitive polynomials per degree (degree 4 to 35)**

Degree	Total Number	Degree	Total Number
4	2	20	24,000
5	6	21	84,672
6	6	22	120,032
7	18	23	356,960
8	16	24	276,480
9	48	25	1,296,000
10	60	26	1,719,900
11	176	27	4,202,496
12	144	28	4,741,632
13	630	29	18,407,808
14	756	30	17,820,000
15	1,800	31	69,273,666
16	2,048	32	67,108,864
17	7,710	33	211,016,256
18	7,776	34	336,849,900
19	27,594	35	929,275,200

Consider a shift register with  $n$ -bit and pseudo-random moves happening between  $2^n - 1$  values ( $LFSSR$  length). These moves can occur quickly since there is minimal involvement of combinational logic (logical gates) and the shift register will therefore navigate the sequence precisely as before once the final state is achieved. In this case, the  $X^n$  ( $n$  is the highest power in the polynomial) and 1 occurs in primitive polynomials, and the former can be used as shift register output, whereas the latter can be used as shift register input. For implementation, it is important to discover the primitive polynomial for  $n$ -bit  $LFSSR$  that is associated with it. For example, the internet tap tables can list taps such as  $N = 4$  (taps at 0, 1, 4) that correspond to  $1 + x + x^4$  [17], [119]. The taps 0, 1 and 4 have been used in this case since they match powers of  $x$  in primitive polynomials. However, there are tap tables that omit 0 and 4 because they are assumed to be naturally present. The recorded taps are normally connected with one primitive polynomial, however several polynomials exist with the same degree. This results in diverse tap tables demonstrating distinctive numbers. For example,  $n = 8$  and a primitive polynomial  $1 + x^2 + x^3 + x^4 + x^8$ . Moreover, every degree has different primitive polynomials that must fulfill other mathematical and numerical conditions. For instance, the most critical

property of these mathematical conditions is that their reciprocals form primitive polynomials. For example,  $1 + x^3 + x^4$  is termed degree 4 and its reciprocal is  $1 + x + x^4$  (10011 and 11001, which are both primitive) [16].

### 3.6.2 General attack on *LFSR* based stream ciphers

If the attacker has a good number of bits from the ciphertext and its corresponding plaintext, they can apply this information to find the *LFSR* output and try to find a minimal sequence producing the same output. This may reveal the original *LFSR* function using the Berlekamp–Massey algorithm [120]. To minimise the risk of this attack, it is possible to use nonlinear combinations of *LFSR* bits on internal states or to use more than one *LFSR* so that one is controlling the output of the other (as in *SG*). Another method is to use an Alternating Step Generator, where irregular clocking for *LFSR* is used [119], [120].

### 3.7 Unique Window Size (*UWS*)

The pseudo-randomness tests are based on the maximum order complexity. The goal of this test is to examine the sequence generated by the targeted cipher if it is complex enough, such that it is hard to find a certain pattern that allows it to be simulated by the possible shortest *NFLSR*. The larger *UWS* is better, thus it is hard to find the *NFLSR* that can generate the same sequence. This kind of test investigates the situation when every state in the keystream is unique by applying a slide window. Let  $Z$  be the keystream sequence. Now we have sliding window  $w > 2$ , as  $w = 2$  there will be repetition, so we start with  $w = 3$ , then increase by 1. If the state is repeated, then  $UWS \neq w$ . See the following example for illustration.

#### Example of finding the *UWS*

This is an example of how to find a unique window size. Assume we have a binary sequence:

$$s = 1001011010\dots$$

Then we are looking for a sliding window where every state is unique (no repetition), so we start with size 3 then 4, etc until every state is unique then that will be  $UWS_2$ . We start

with window size  $w = 3$  as with size  $w = 2$  there will be always repetition as following (first row is the sequence  $s$ ):

1	0	0	1	0	1	1	0	1	0...
<b>1</b>	<b>0</b>	<b>0</b>							
	<b>0</b>	<b>0</b>	<b>1</b>						
		<b>0</b>	<b>1</b>	<b>0</b>					
			<b>1</b>	<b>0</b>	<b>1</b>	<b>duplicated</b>			
				<b>0</b>	<b>1</b>	<b>1</b>			
					<b>1</b>	<b>1</b>	<b>0</b>		
						<b>1</b>	<b>0</b>	<b>1</b>	
							<b>0</b>	<b>1</b>	<b>0</b>

With sliding window size = 3, 1 0 1 was duplicated.

Now we try the next size which is 4:

1	0	0	1	0	1	1	0	1	0	0...
<b>1</b>	<b>0</b>	<b>0</b>	<b>1</b>							
	<b>0</b>	<b>0</b>	<b>1</b>	<b>0</b>						
		<b>0</b>	<b>1</b>	<b>0</b>	<b>1</b>					
			<b>1</b>	<b>0</b>	<b>1</b>	<b>1</b>				
				<b>0</b>	<b>1</b>	<b>1</b>	<b>0</b>			
					<b>1</b>	<b>1</b>	<b>0</b>	<b>1</b>		
						<b>1</b>	<b>0</b>	<b>1</b>	<b>0</b>	
							<b>0</b>	<b>1</b>	<b>0</b>	<b>0</b>

With sliding window size  $w = 4$ , there is no duplicate.

The time complexity of the  $UWS$  test is linear which depends on the keystream size.

The following is the algorithmic description for  $UWS$ :

### Algorithm 3.1

STATE Given a periodic bit sequence  $B[i]$  with period  $P$ , the *UWS* algorithm calculates the *UWS*.

STATE Given:  $B$  = Periodic bit sequence,  $P$  = Period of  $B$

STATE Calculate:  $L$  = Minimum subsequence length such that all  $L$ -bit subsequence are unique

STATE Initialise  $L$  with 1

STATE REPEAT

FOR {each bit index,  $i$  of  $B$ }

STATE Test

FOR {each bit index,  $j$ , of  $B$  greater than  $i$ }

IF { $L$ -bit subsequence of  $B$  starting at  $i$  =  $L$ -bit subsequence of  $B$  starting at  $j$ }

STATE INCREMENT  $L$

STATE CONTINUE the next REPEAT loop

ENDIF

ENDFOR

ENDFOR

STATE UNTIL all  $L$ -bit subsequences of  $B$  are unique

STATE Returns  $L$

### 3.8 The $d$ -monomial test

The  $d$ -monomial test is a random statistical test, in which the algebraic normal form (*ANF*) is used to represent the Boolean function used in the keystream generated by *SG* and *SSG*. This test was introduced by Filiol in 2002 [121] to detect the biases in the keystream in some chosen stream ciphers such as *DES* and *AES*<sub>2</sub>, and also in the hash function. Filiol observed that the *ANF* for the pseudo-random binary sequence contains a monomial with  $d$  weight following an approximate normal distribution as follows:

$$\mathcal{N}\left(\frac{1}{2}\binom{n}{d}, \frac{1}{2}\sqrt{\binom{n}{d}}\right) \quad (3.4)$$

In 2006 Saarinen [122] applied this test in some IV-chosen stream ciphers and found the relationship between detectable biases on keystream and gate complexity. Englund et al. [34] applied this test using polynomial characterisation and introduced other IV-chosen

stream ciphers based on  $d$ -monomial tests which they named the monomial distribution test and maximal monomial test.

### 3.8.1 Running $d$ -monomial tests

We start with a truth table representation of the Boolean functions to the  $ANF$  representation by using Möbius Transform. Next, the  $d$ -monomial C program calculates the number of monomials of weight  $d$  in the  $ANF$  of a Boolean function. Multiple functions can be analysed by providing their  $ANF$  in separate rows in the file, so each row in the input file contains the  $ANF$  of a single Boolean function. Then, we apply the  $\chi^2$  (goodness of fit) test to find if the keystream passes the  $d$ -monomial test and is monomial.

#### Example

In the following example, we apply the monomial on  $ANF$  of four Boolean functions. The first row is:

```
0101
1001
1011
0011
```

where 0101 corresponds to a Boolean function with two input bits:  $f(x_1, x_2) = 0 + 1.x_2 + 0.x_1 + 1.x_1x_2$ . Note that the coefficients of each term in the expression correspond to the  $ANF$  of the function. Now, from the expression, the  $ANF$  of the function has one monomial of degree  $d = 1$  as only one term of  $(x_2)$ , and one monomial of degree  $d = 2$  as only one term of  $(x_1x_2)$  with two multiplied variables.

As mentioned earlier for Saarinen and Filiol's adoption for  $d$ -monomial tests, see [121] and [122], as well as the other approach in [34]. However, the treatment of this test here is that each bit of the  $SG$  and  $SSG$  keystream is considered a Boolean function which is a representation of all the variables obtained by different keys as the initial input of  $LF SR$ .

**Definition 3.6:** The ( $f$ ) Boolean function as a  $\mathbb{F}_2^n \rightarrow \mathbb{F}_2$ , let  $x_i$  elements of the vector  $x$ , then:

$$f(x^T) = a_0 \oplus a_1x_1 \oplus \dots \oplus a_nx_n \oplus a_{2^n-1}x_1x_2x_3, \dots, x_n \quad (3.5)$$

Then  $f$  is transformed as:

$$\hat{f}(x^T) = \sum_{a \in F_2^n} f(a) \prod_{i=1}^n x_i^{a_i} \quad (3.6)$$

Such that  $\hat{f}$  polynomials is the *ANF* representation of  $f$ .

So  $\hat{f}$  is monomials  $(x_i)$  permutations, that is  $d$  tests based on finding the non-zero bits in  $\hat{f}$ , in other words,  $d$  the Hamming weight (the longest monomial in *ANF*) [123].

### 3.9 The shrinking generator

The shrinking generator (*SG*) [24] is an IV-less synchronous stream cipher. It has simple design features where the internal structure of the *SG* contains two *LFSRs*. *LFSR<sub>A</sub>* works as input and *LFSR<sub>B</sub>* as controlling bits (see Figure 3.4). Both *LFSRs* must be primitive polynomials. The selection rule is that if the bits of *LFSR<sub>B</sub>* are 1s then the input bits 1 or 0 of *LFSR<sub>A</sub>* will be selected, and if the bits of *LFSR<sub>B</sub>* are 0s then the input bits of *LFSR<sub>A</sub>* will not appear in the *SG* outputs. Consider the following example:

For *LFSR<sub>A</sub>* we choose a primitive polynomial  $f_1 = x^3 + x + 1$  and if the key is (001) then the *LFSR<sub>A</sub>* output will be 1010011.

Let  $f_2 = x^4 + x + 1$ , a primitive polynomial for *LFSR<sub>B</sub>*. If the key is (0001) then the *LFSR<sub>B</sub>* output will be 101011001000111. The *SG* keystream will be: 11010...

The following example for two *LFSRs* with their polynomials shows how the *SG* keystreams were generated.

$$\text{LFSR}_A \quad x^3 + x + 1, S_{t-1} \oplus S_{t-3}$$

Output:

1 0 1 0 0 1 1

$$\text{LFSR}_B \quad x^4 + x + 1, S_{t-1} \oplus S_{t-4}$$

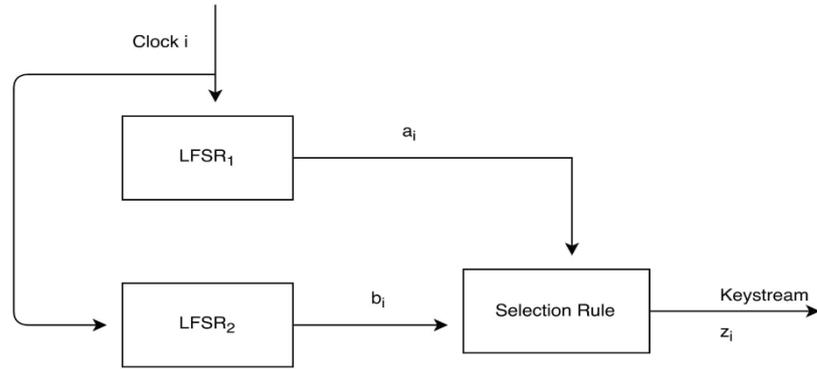
Output:

1 0 1 0 1 1 0 0 1 0 0 0 1 1 1

*SG* keystream output:

(1,1) (0,0) (1,1) (0,0) (1,0) (1,1) (0,1) (0,1) (1,0) (0,1) (0,0) (0,0) (1,1) (1,1) (1,1) . . . .

1 - 1 - 0 1 - - 0 - - - 1 1 1...



**Figure 3. 4 Shrinking generator design**

### 3.9.1 Shrinking generator period

To explain the  $SG$  period, consider the following:

**Definition 3.7:** Let  $2^{|A|} - 1$  and  $2^{|B|} - 1$  be the periods of  $LFSR_1$  and  $LFSR_2$  respectively, where  $|A|$  is the degree of  $LFSR_1$  and  $|B|$  is the degree of  $LFSR_2$ , then the period of  $SG$  is  $(2^{|A|} - 1) \cdot 2^{|B|-1}$ .

### 3.9.2 Linear complexity

**Definition 3.8:** For the linear complexity ( $LC$ ) of the  $SG$  keystream, the lower bound is  $|A| \cdot 2^{|B-2|}$  and  $LC$  is greater than the lower bound, and the upper bound is  $|A| \cdot 2^{|B-1|}$ , and  $LC$  is less than or equal to the upper bound, so  $LC$  is located within tight boundaries.

### 3.9.3 Attacks on shrinking generator

One interesting attack on  $SG$  is the fast correlation attack by Golic [124]. This represents an important conjecture that the  $SG$  output sequence can be a good target for correlating attacks with some  $LFSRs'$  initial states, without the need to go through all of them. This is further investigated by Zhang et al., through studying the  $SG$  with some known connections [125].

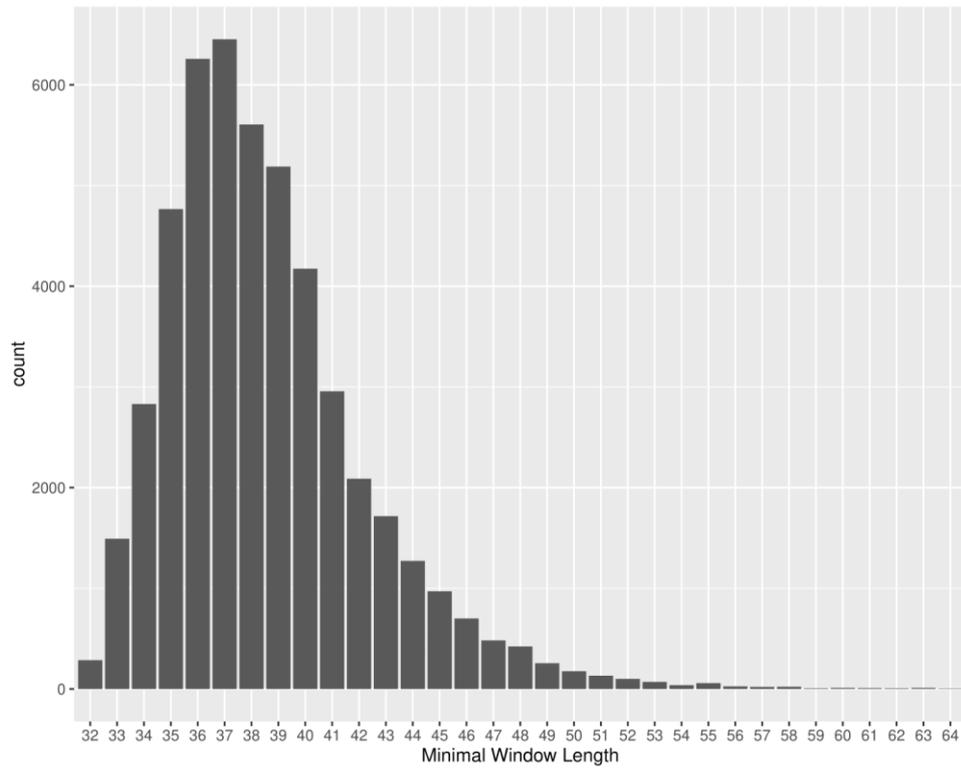
In addition, Golic et al. introduced a statistical distinguisher which reduces the time complexity for the computations. The idea is based on reconstructing the bits which clocked in  $LFSSR_A$  in a regular way. The resulting sequence of bits should satisfy the recurring low weight polynomial which is constructed from multiple  $LFSSR_A$ , where  $LFSSR_A$  was chosen randomly [126].

Another interesting observation by Kdahl et al. [127] proposed an attack and observation by investigating the low weight polynomial which can generate the  $LFSSR$ . They found that most of the bits on the  $LFSSR$  output can be represented by the linear recursion with more occurrence than random ones, which in turn can give an estimation of the occurrence of bits. Furthermore, that can give some prediction of bits distribution on the given generation sequence.

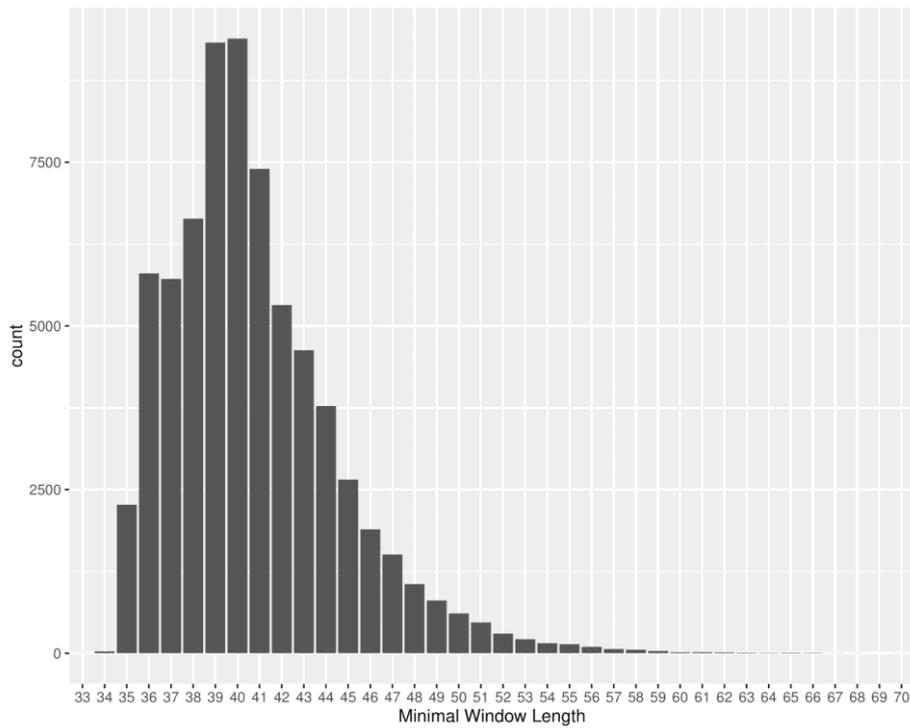
#### **3.9.4 UWS on shrinking generator**

By applying Unique Window Size to find the minimal length for the  $SG$  keystream to make sure that every state is unique, the results showed that identifying the  $UWS$  can lead to finding the  $LFSSR_s$  pairs' weakness. This, in generating the keystream that is pseudo-random, can in turn help understand which  $LFSSR$  pairs should be avoided. In addition, the  $UWS$  is linked to maximum order complexity (as opposed to linear complexity), thus yielding experimental evidence of possible weaknesses in the  $SG$  keystream against attacks which may use this measure. This is another possible area for further research.

As discussed, the  $SG$  consists of two  $LFSSRs$ . The C code for the  $SG$  was used to generate the keystream, then another C code was used to calculate the  $UWS$  by using super computation. EC2 was used to run the C codes, as these computations are time consuming. Figure 3.5 shows the distribution for the  $UWS$  for  $SG$  of degree 19 and Figure 3.6 shows  $UWS_{20}$ .



**Figure 3. 5 Unique window size 19 distribution for Shrinking Generator**



**Figure 3. 6 Unique window size 20 distribution for Shrinking Generator**

The shrinking generator design takes into consideration how to build a cipher with a simple structure and yet still have good cryptographic properties. Since this cipher appeared, there

have been many attacks targeting it but no attack has successfully recovered the key when the *LFSRs* pairs were chosen carefully. However, the attacks reveal some weaknesses and our results also confirmed weaknesses in the keystream when multi  $\chi^2$  tests were applied [128]. Some statistical dependencies were also revealed by applying *UWS* tests. More details of these results are given in sections 3.12, 3.13 and 3.14.

### 3.9.6 *d*-monomial test results for SG

By applying statistical randomness tests on *SG* in order to find a weakness in the keystream, using the *d*-monomial test, results were obtained. The *d*-monomial test was run for *SG* degrees 16–19, in order to find the passing rate using the  $\chi^2$  test. See Table 3.2, and for more extensive results with variations on *LFSRs* initial input, see Appendix 3.1.

**Table 3. 2 *d*-monomial test implemented on Shrinking Generator with degrees 16–19**

<i>SG</i> Degree	Number of <i>LFSRs</i> Pairs	Number of $f_i$	Total Number of Fails at $\alpha = 0.01$	Passing Percentage
16	50	409600	198895	51.441%
17	50	819200	510001	37.744%
18	50	1638400	932719	43.071%
19	50	1114112	655568	41.158%

### 3.10 The self-shrinking generator

The self-shrinking generator consists of one primitive polynomial working as *LFSR* and the selection rule is within this *LFSR*. If the *LFSR* output pairs are (1,0) the *SSG* output will be 0 or if the pair is (1,1) the output will be 1, otherwise the output will be neglected, as shown in Figure 3.7. The self-shrinking generator was designed by Meier and Statfelbach, as described in their initial paper [114].

For example, if the *LFSR* primitive polynomial is  $f = x^4 + x + 1$ , then the *SSG* output will be 10010...

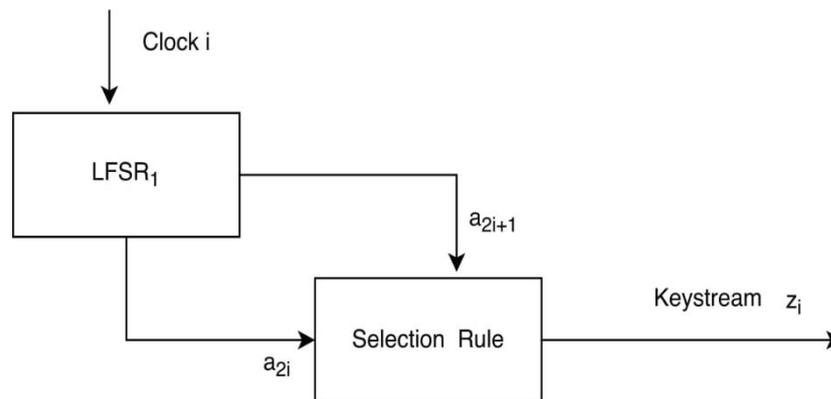
The following illustration for *LFSR* with its polynomial shows how the *SSG* keystreams were generated.

$LFSR(x^4 + x + 1), S_{t-1} \oplus S_{t-4}$

Output: 0 0 0 1 1 1 1 0 1 0 1 1 0 0 1 0 0 0

SSG output:

0	0	0	1	1	1	1	0	1	0	1	1	0	0	1	0	0	0
			1	0	0	1			0								



**Figure 3. 7 Self-shrinking generator design**

### 3.10.1 The period

$SSG$  uses one  $LFSR$  with an output sequence  $\{S_t\}_{t=0}^{2^n-2}$  with length  $2^n - 1$ . Within the  $LFSR$  there are two decimated sequences: one is the output  $\{S_{2t}\}_{t=0}^{2^n-2}$  and the other one is the controlling sequence  $\{S_{2t+1}\}_{t=0}^{2^n-2}$  which generates  $SSG$  keystreams with the period  $2^{n-1}$ .

### 3.10.2 Self-shrinking generator linear complexity

The linear complexity  $LC$  is  $LC < 2^{\lfloor N/2 \rfloor - 1}$ , where  $N$  is the length of the keystream [25], so  $LC$  in  $SSG$  is more complex than in  $SG$ . In this thesis, the relation between the  $LC$  keystream and the primitive polynomial degree was found to be as follows:

The relationship between degree and  $LC$  is estimated to be:

$$\ln(LC + 1) = -0.33 + 0.33 * degree \quad (3.7)$$

This can be rewritten as:

$$LC = 0.72e^{0.33*degree} - 1 \quad (3.8)$$

See Appendix 3.2 for the complete explanation with scatterplots.

### **SSG linear complexity profile results and observation**

To perform the  $LC$  profile test is vary from of the  $LC$ , by considering the keystream as a binary sequence, and subsequences  $s_1, s_2, \dots, s_n$  which will be calculated by their  $LC$ , that is the  $LC$  profile. In this case  $s_1 =$  first bit,  $s_2 =$  first 2 bits,  $s_3 =$  first 3 bits, ...,  $s_n =$  the whole  $n$  bits. Appendix 3.2 has a computation for  $SSG$  keystream string from primitive polynomial; from degree 4 to 8.

And the relation between  $LC$  profile and the degree can be represented as:

$$\ln(LC + 1) = -1.37 + 0.58 * degree \quad (3.9)$$

This can be rewritten as:

$$LC = 0.25e^{0.58*degree} - 1 \quad (3.10)$$

### **3.10.3 Attacks**

Several types of attacks have already been performed on  $SSG$ . One of these attacks relies on some known bits of the keystream and involves trying to establish an algorithm which simulates a sequence that can produce the same bits. By producing this sequence, an attacker can establish the key. This attack is presented in the work of Zenner et al. [129], who state that the algorithm takes about  $\mathcal{O}(2^{0.694L})$  steps and the key length is  $L$ . Another kind of attack is proposed by Debraize et al. [130]. Their attack is performed with a feedback polynomial which has a Hamming weight of at most 5, and they perform some guesses on the internal bits of  $SSG$  and use the SAT solver to solve the system.

Zhang et al. [129], in their guess and determine attack, aim to simulate the initial state and reduce the time complexity to  $\mathcal{O}(2^{0.556L})$  where  $L$  is the keystream length. Their attack requires reasonable keystream length.

Some statistical dependencies on  $SSG$  are summarised. Computations were run for all primitive polynomials to find the unique window size up to degree 25, and non-exhaustive computation from degree 26 up to degree 35 as can be seen in [128].

In addition,  $LFSRs$  seeds were changed. First, the  $d$ -monomial test was applied, performing the test for  $LFSRs$  and applying it to all possible combinations of  $LFSRs$  and all initial states from degree 7, applying three different scenarios in order to vary their initial states. Varying percentages keystoream bits  $z_i$  of failing the  $d$ -monomial test from 18% to 58% were failed with significance level  $\alpha = 0.01$ . When 50  $LFSRs$  pairs from degrees 16 to 19 were chosen, the same kind of failure rates continued at the significance level  $\alpha = 0.01$ .

#### **3.10.4 Statistical tests on SSG**

This section presents results obtained on the shrinking generator ( $SG$ ) and the self-shrinking generator ( $SSG$ ). The computations for the  $d$ -monomial based test and  $UWS$  were done mostly with fast computing power using the Victorian Partnership for Advanced Computing as well as using  $EC2$  from Amazon Web Service ( $AWS$ ) for more extensive computations.

#### **3.10.5 UWS test on SSG**

As  $UWS$  is based on maximal order complexity [113], [114], this thesis tries to find within the binary sequence (keystream) if every state of bits is unique, by using the sliding windows. The statistical randomness tests expose the flaws in the keystream which in turn show the poor choice of polynomials that generate such a keystream. Thus caution in the choice of the  $LFSRs$ (polynomials) is critical.

The findings show that based on the unique window size of the  $SSG$  outputs and the imbalance property, the choice of  $LFSR$  primitive polynomials on the  $SSG$  should avoid certain polynomials, regardless of the degree of the polynomial. This is because certain  $SSG$  patterns derived from such polynomials produce biased  $SSG$  outputs. For example, out of 48 primitive polynomials of degree 9, three should be avoided. These extreme polynomials of degree 9 include:

- 1)  $x^9 + x^8 + x^7 + x^5 + x^4 + x^2 + 1$
- 2)  $x^9 + x^7 + x^4 + x^2 + 1$
- 3)  $x^9 + x^7 + x^5 + x + 1$

The above examples of degree 9 polynomials were not suitable for *LFSR*, and certain primitive polynomials of all degrees may also produce extreme results to be avoided because they affect the randomness appearance of the *SSG* output.

Regarding primitive polynomials of degree 9, each *SSG* of degree 9 generates an output length of 256. Because there are 48 primitive polynomials of degree 9, the total *SSG* windows will generate 48 x 256 windows. If the degree of the polynomial in the potential computation is increased, repeated output segments will grow exponentially.

It is worth investigating the primitive polynomials for *UWS* with certain weights as shown in Figures 3.10 and 3.11, which was done for all weight counts as seen for *UWS21* in Table 3.3 and with only weight 5 in Table 3.4, because the certain weight is easier to analyse, with less computation effort.

**Table 3. 3 Unique Window Size 21 for Self-Shrinking Generator counts and probability, with all weights**

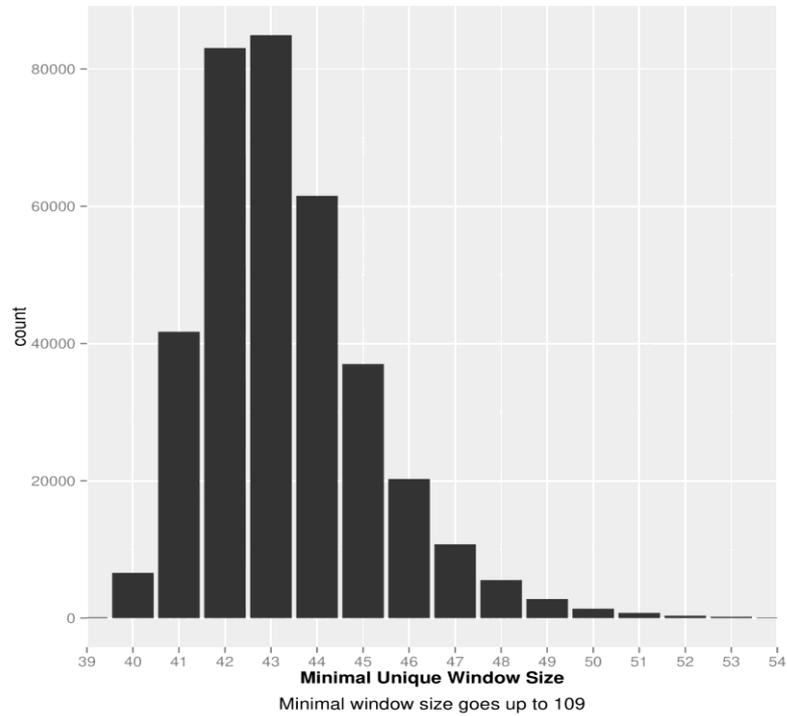
UWS21	UWS21 Count	P(UWS21)
35	32	0.000377929
36	1522	0.017975246
37	9835	0.116154101
38	19746	0.233205782
39	20229	0.238910147
40	14697	0.17357568
41	8729	0.103091931
42	4790	0.05657124
43	2538	0.02997449
44	1237	0.014609316
45	639	0.007546769
46	355	0.004192649
47	160	0.001889645
48	75	0.000885771
49	32	0.000377929
50	28	0.000330688
51	13	0.000153534
52	6	0.000070862
53	2	0.000023621
56	3	0.000035431
58	1	0.00001181
62	1	0.00001181
63	1	0.00001181
85	1	0.00001181

**Table 3. 4 Unique Window Size 21 for Self-Shrinking Generator counts and probability, with weight = 5**

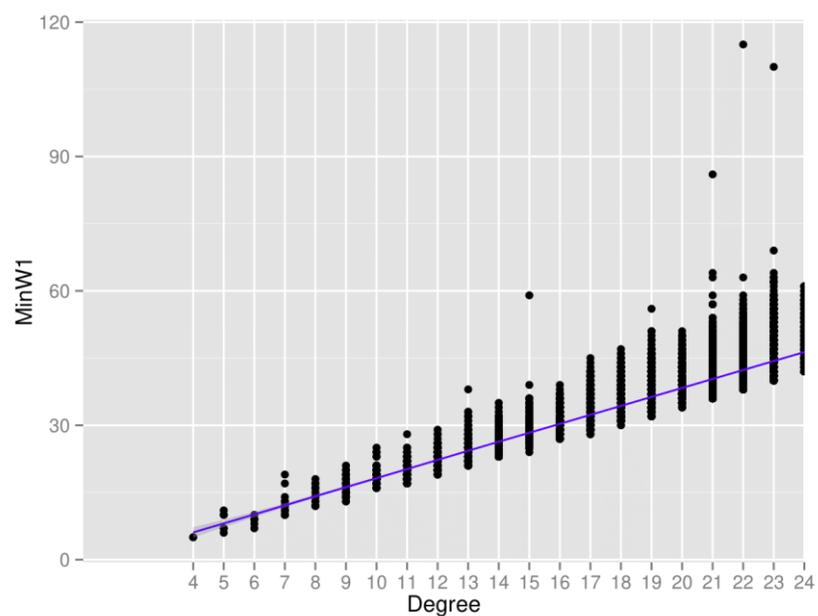
UWS	UWS Count	P(UWS)
36	2	0.012195122
37	21	0.12804878
38	38	0.231707317
39	36	0.219512195
40	32	0.195121951
41	13	0.079268293
42	16	0.097560976
43	5	0.030487805
46	1	0.006097561

The same implementation as *SG* is done for the *SSG*. The histogram in Figure 3.8 shows the *UWS23* for the *SSG* key streams, the *UWS* range from length 39 up to 106, and *UWS* length concentrates between 42 to 45, so the number of all possible simulations is 356,960. In Figure 3.9, the plot for *UWS* degrees is from 4 to 24 and the number of

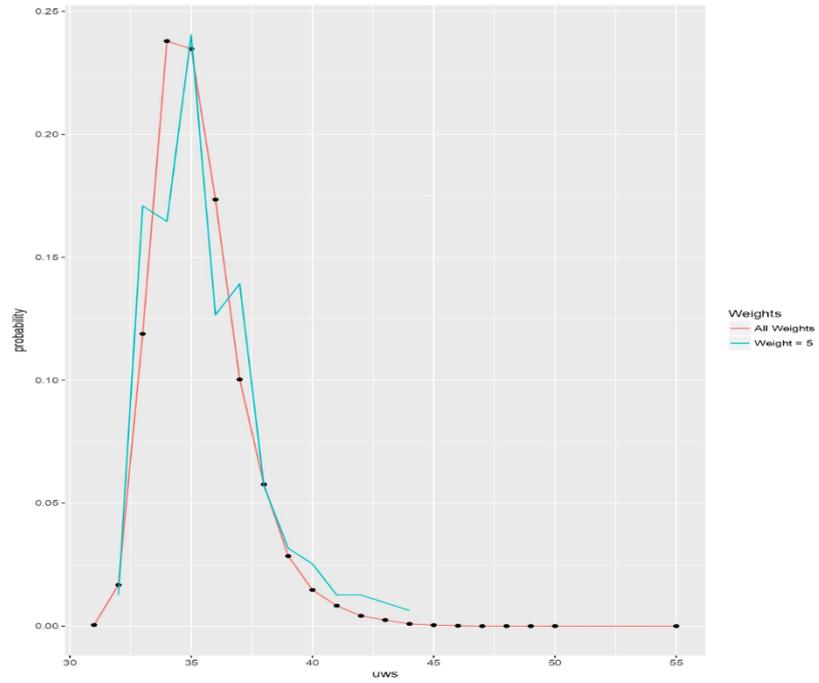
primitive polynomials is 901,934. In addition,  $UWS19$  is plotted by choosing the  $LF SRs$  with weight = 5 and see how they fit against  $LF SRs$  with all weights including 5 in Figure 3.10 and Figure 3.11. The variation is small. Therefore, other kinds of tests on  $SSG$  other than  $UWS$  must be considered, thus the need to apply the  $d$ -monomial based tests. Appendix 3.3 provides some statistical analysis for  $SSG$  with  $UWS4 - 24$ .



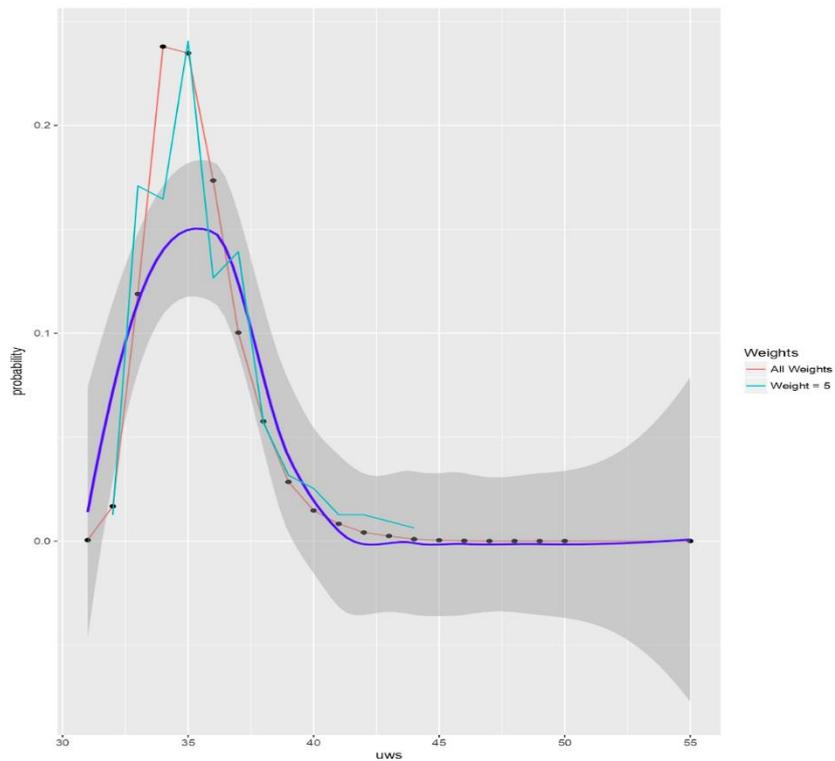
**Figure 3. 8 Unique window size 23 distribution for Self-Shrinking Generator**



**Figure 3. 9 Self-Shrinking Generator degrees from 4 to 24 vs Unique Window Size**



**Figure 3. 10 Unique Window Size 19 for Self-Shrinking Generator with weight 5 and all weights**



**Figure 3. 11 Unique Window Size 19 for Self-Shrinking Generator with weight 5 and all weights, with smooth line**

Appendix 3.4 provides more results for  $UWS9$  and  $UWS10$ , counting all weights, as well as a histogram showing the calculations for primitive polynomials of degree 14, for weight versus  $UWS$ .

Let us take an example for degree 12. Out of 144 polynomials, there are about 10 out of 2048 weak keys (initial seeds). Overall, 680 out of  $144 \times 2048$  keys are weak, resulting in a bias keystream which can be identified. It is very computationally difficult to find weak polynomial and key combinations for much higher degrees as higher degrees generally produce better random appearance keystream [128].

For degrees 2 to 24 for  $SSG$  as one set, Table 3.5 summarises how many of each  $UWS$  were repeated.

**Table 3. 5 Total count of each Unique Window Size occurrence for degrees 4 to 24 for Self-Shrinking Generator**

UWS	Count								
4	2	14	15	24	312	34	8856	44	117
5	1	15	14	25	453	35	10376	45	56
6	3	16	24	26	617	36	10889	46	22
7	1	17	55	27	785	37	8789	47	10
8	3	18	60	28	995	38	5854	48	7
9	8	19	75	29	1577	39	3303	49	6
10	6	20	71	30	2472	40	1818	50	3
11	7	21	110	31	3064	41	1004	55	1
12	6	22	220	32	3730	42	492	58	1
13	15	23	259	33	5940	43	286		

### 3.10.6 The d-monomial test on SSG

Table 3.6 shows, for degree 6 and degree 7, the number of bad, weak and failing polynomials with different initial seeds (keys). Table 3.7 shows other computations for all possible variations of keys, as for degrees 6, 7, 12 and 14.

**Table 3. 6 Finding weak polynomials with degrees 6 and 7 with different initial seed (key)[128]**

Degree $n$	No. of bad $g(x)$	No. of Failing $f_i$	$\alpha = 0.01$	$\alpha = 0.05$	$\alpha = 0.1$
6	2	1	0	0	1
6	2	1	0	1	0
6	1	2	0	2	0
<b>6</b>	Total number of weak polynomials is 5 out of 6				
7	2	1	0	0	1
7	1	1	1	1	0
7	2	2	1	0	1
7	5	1	1	0	0
7	1	3	1	0	2
<b>7</b>	Total number of weak polynomials is 11 out of 18				

**Table 3. 7 Finding weak polynomials with degree 6, 7, 12 and 14 with all initial seed (key)**

Degree $n$	No. of $g(x)$	Number of $f_i$	Number of Passing $f_i$	Percentage of Passing $f_i$
6	6	216	186	86.10%
7	18	1152	1136	98.60%
12	144	294912	294236	99.80%
14	756	6193152	6186899	99.90%

### 3.11 Comparison between Shrinking Generator and Self-Shrinking Generator results

Table 3.8 shows that for degree 7 for  $SG$  and  $SSG$  the passing rate for  $SSG$  is approximately 98.6% and for  $SG$  it is about 67.3%, which shows that  $SG$  is weaker than  $SSG$ .

**Table 3. 8 d-monomial with degrees 7 to 15 with full keystream string for Shrinking Generator and Self-Shrinking Generator results for comparison**

<i>SG</i> Degree	Number of <i>LFSRs</i> Pairs	Number of $f_i$	Number of Fails at $\alpha = 0.01$	Total Number of Passes	Passing Percentage	<i>SSG</i> Passing Percentage
7	24	768	251	505	67.310%	98.6%
8	72	4608	1561	2255	66.124%	
9	72	9216	3121	4329	66.135%	
10	216	55296	27847	27359	49.640%	
11	624	319488	55203	198895	82.721%	
12	648	663552	339627	206071	48.817%	99.8%
13	2520	5160960	2831437	2083684	45.137%	
14	3840	15728640	7938000	6542587	49.531%	99.9%
15	3840	31457280	18213724	13183968	42.100%	

### 3.12 Other $d$ -monomial based tests and results

This section provides some *ANF* based tests which show some variations of the  $d$ -monomial test. The monomial distribution calculates the frequency of each monomial with a certain weight over a list of Boolean functions.

The  $d$ -monomial test is looking for a monomial of certain degree ( $d$ ) per polynomial. In contrast, the monomial distribution test is looking for all monomials with degree  $d$  across all functions in *ANF* [34], [121].

The maximal distribution test finds the maximal degree monomial per polynomial in *ANF*. Table 3.9 shows results for degrees 7, 8 and 9, with the number of observed maximal monomials vs the number of functions (polynomials). This test is simple and indicates if the key produces a proper mixing of monomials, which in turn can illustrate the strength of a given *LFSRs* pair.

**Table 3. 9 Shrinking Generator exhaustive testing results for maximal monomial test for combined LFSR lengths 7 to 9**

$LFSR_A$	$LFSR_B$	Observed with Fixed $LFSR_B$	Observed with Fixed $LFSR_A$	Number of functions
$x^3 + x^2 + 1$	$x^4 + x^3 + 1$	0	12	32
$x^4 + x^3 + 1$	$x^3 + x^2 + 1$	0	14	32
$x^4 + x^3 + 1$	$x^3 + x + 1$	0	14	32
$x^4 + x + 1$	$x^3 + x^2 + 1$	0	16	32
$x^4 + x^3 + 1$	$x^5 + x^2 + 1$	0	66	128
$x^4 + x^3 + 1$	$x^5 + x^3 + 1$	0	66	128
$x^5 + x^2 + 1$	$x^4 + x + 1$	0	70	128
$x^5 + x^3 + 1$	$x^4 + x^3 + 1$	0	80	128

### 3.13 Data distribution

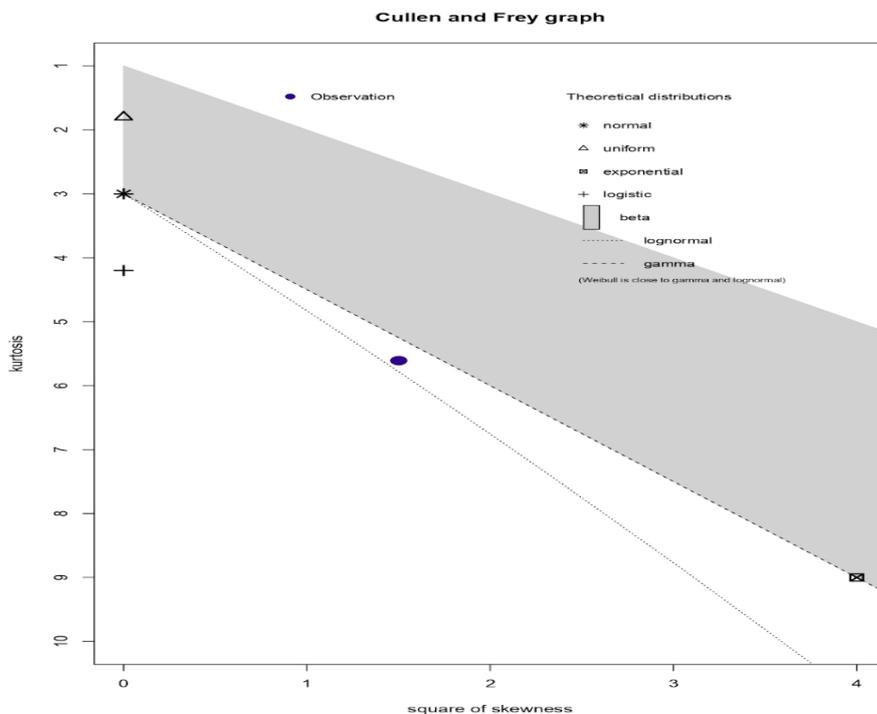
Usually the sequences of data that are generated using  $SG$  or  $SSG$  would be a discrete array of data. Most of the data columns or variables follow a pattern, such as one of the probability distributions: normal, log-normal, gamma, beta, Poisson, binomial, negative binomial or hyper-geometric. Figure 3.12 compares some types of distributions. It is useful to eliminate the type of distributions that do not appear to be good candidates, and reduce them to the best few candidates. One of the most important population data distributions is the normal or Gaussian distribution. If data follows this distribution, or could be transformed to follow a normal distribution, then parametric statistical methods can be used to further analyse the data as most of the parametric methods are based on the assumption of normality of data [34], [107], [121]. One of the advantages of parametric methods is that they are very strong and powerful compared to the non-parametric methods.

The normal distribution provides a frequency distribution along a bell shaped curve. This distribution has two parameters: mean and standard deviation. The importance of the normal distribution is to study and analyse various statistical phenomena, and in particular to find the probability of a specific event and its occurrence.

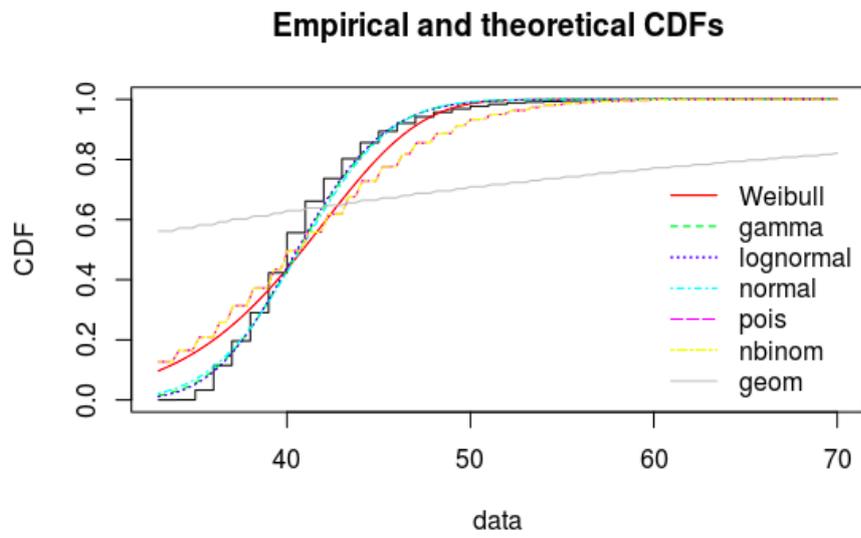
### 3.13.1 Statistical modelling: Predicting *UWS*

To predict or model any variable, the inherent probability distribution of that variable must be known. A goodness of fit test is conducted on several or most of the probability distributions that could relate to this kind of data, based on the plot or best guess. Easyfit software [131] runs goodness-of-fit tests on most of the statistical probability distributions and ranks the most suitable distributions in preferential order. Easyfit and the “fitdistrplus” package in R software [131], [132] were used to manually check the preference criteria using graphs and goodness-of-fit test results on the most expected probability distributions.

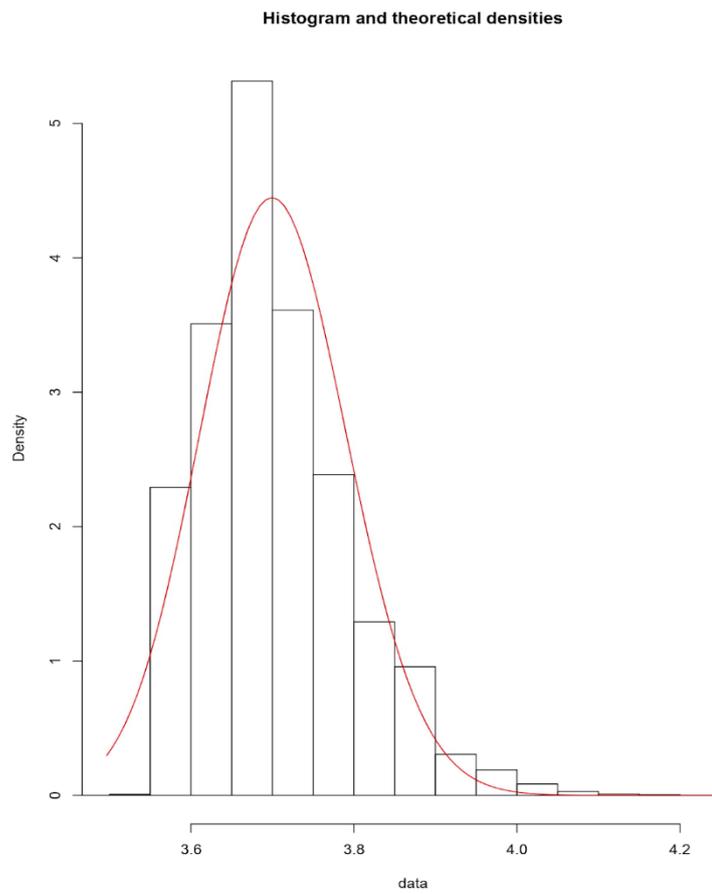
One of the data sequence variables, the Unique Window Size of degree 20 (*UWS20*), was used to test the process of prediction modelling or forecasting. *UWS20* was plotted and found it to be skewed (Figure 3.12). Using the “descdist” command in R, a graph was produced for data fit and the distribution of *UWS20* approximately follows any one of the log-normal, normal, gamma or Weibull distributions (Figure 3.13) which was also supported by the Easyfit software. R was then used to approximate the probability distribution by using goodness-of-fit tests and graphs (Figure 3.14). By comparing the cumulative density functions of these distributions, it can be seen that the lognormal distribution has better fit than other kinds of distributions.



**Figure 3. 12 Unique Window Size 20 different kinds of distributions for Shrinking Generator**



**Figure 3. 13 Comparison of cumulative density functions of observed and theoretical distributions, Unique Window Size**



**Figure 3. 14 Unique Window Size 20 lognormal distribution for Shrinking Generator**

Figure 3.14 and the Easyfit software distributional preference showed that the distribution of *UWS20* approximately follows a lognormal distribution. More detailed statistical observation showed the lognormal was the best probability distribution based on goodness of fit (see the table in Appendix 3.5, as well as plots and figures for comparison of different statistical distributions).

**Definition 3.9:** The distribution function of the lognormal distribution is:

$$\mathcal{N}(\ln x; \mu, \sigma) = \frac{\exp[-\frac{(\ln x - \mu)^2}{2\sigma^2}]}{\sigma\sqrt{2\pi}} \quad (3.11)$$

Where  $\mu$  is the mean, and  $\sigma$  is the standard deviation(sd),  $x$  is the variable and  $\ln x$  its variance.

Using the Bartel rank test, Cox Stuart test, rank test and runs test showed that the sequence of data (*UWS20*) is actually a non-random sequence (p-value <0.001) as shown in Table 3.10.

**Table 3. 10 Three randomness tests for Shrinking Generator with Unique Window Size 20**

Test	Statistic	P value	Decision (alternative hypothesis)
Bartels Ratio test	-83.926	< 2.2e-16	non random
Cox Stuart test	8957	< 2.2e-16	non random
Runs test	-66.186	< 2.2e-16	non random

This indicates that this sequence of data can be predicted using statistical models.

Two methods were used to learn about the pattern and predict the sequence: (1) generating a theoretical lognormal sequence (simulated *UWS20*) from the mean and standard deviation of the observed data (e.g., *UWS20*) of the same length and comparing the accuracy of prediction, and (2) using a linear regression method to predict and check the accuracy of the predicted sequence including validation and calibration.

## Method 1

**Definition 3.10:** The lognormal variate ( $X$ ), and  $z$  the standard normal variable is defined as:

$$X = e^{\mu + \sigma \cdot z} \quad (3.12)$$

which follows the probability density function:

$$\mathcal{N}(\ln x; \mu, \sigma) = \frac{\exp\left[-\frac{(\ln x - \mu)^2}{2\sigma^2}\right]}{\sigma\sqrt{2\pi}}$$

Using *R* software, the mean ( $\mu$ ) and standard deviation ( $\sigma$ ) of the observed *UWS20* were 3.70 and 0.09 respectively. Then, *rlnorm* commands in *R* software were used to generate equal numbers of observations (70,416) from the lognormal distribution using the above mentioned mean and standard deviation in Figure 3.14. The difference between observed *UWS20* and simulated *UWS20* was calculated and the accuracy compared, with the difference between the observed and simulated predicted *UWS20* being 0. The accuracy was 7.8%. It would be higher if the accuracy is allowed to be flexible e.g., within 1 or 2 units of the observed *UWS20*. This was expected, as *UWS20* was found to be non-random and hence a random sequence could not entirely predict the sequence.

## Method 2

The linear regression model was used where the outcome was “observed *UWS20*” and explanatory variables were input degree, input weight, control degree, control weight, input polynomials and control polynomials. Each of the input and control polynomial variables had up to 17 degrees of polynomials and hence produced 17 separate variables based on the possible terms in a primitive polynomial of degree 17. For *SG* with degree 20, the highest *LFSSR* degree is 17 and the lowest is 3 for primitive polynomials combination.

The first step in method 2 was to extract 34 variables from the input and control polynomials and add them with four other independent variables (input degree, input weight, control degree and control weight) to form a pool of independent variables. The input and control polynomial variables extracted from mathematical formulas (which were initially used to generate *UWS20*) are binary variables and are recorded in binary form, either in “Yes” or “No” depending on whether that degree of polynomial was present in

that equation or not. It is not unusual to include a mix of continuous, discrete and binary variables among the independent or predictor variables as long as the outcome or dependent variable is numerical (continuous or assumed as such) [107]. In the following step, a simple linear regression model was run on each of these 38 variables to find univariate significance, in order to find suitable candidates for the multivariable linear regression model. It appears that, out of all the variables, input degree is the strongest predictor, as it has the highest R square value and accuracy rate. Interestingly, it is highly correlated with control degree and, as such, the two cannot be included in the same model. This means that control degree and input degree as variables are correlated, or input degree =  $f(\text{control degree})$ , so only one of them is necessary.

To run a multivariable model with all the suitable predictors altogether, multicollinearity, the correlation between the predictors or independent variables, was checked. One of the assumptions of linear regression is that the independent variables are not correlated. Although most of the variables were highly correlated, three of them were so highly correlated (almost perfectly correlated with one or more of the independent variables) that they could not be included in the same model due to multicollinearity. Hence, these three variables were discarded: control degree, input<sub>14</sub> and input<sub>17</sub>. Then the rest of the variables were put in a backward elimination stepwise regression where all the variables are included in the multivariable linear regression model and then variables that have p-values higher than 0.05 are discarded. From the stepwise regression 11 variables were discarded (input<sub>2</sub>, input<sub>3</sub>, input<sub>4</sub>, input<sub>5</sub>, input<sub>6</sub>, input<sub>7</sub>, input<sub>8</sub>, input<sub>9</sub>, input<sub>10</sub>, input<sub>15</sub>, input<sub>16</sub>) and the rest of the 24 variables were kept in the multivariable model for predicting *UWS20* as they were all significant. After putting all of the chosen variables in the final model, the prediction model is:

*Predicted : log(UWS20)=*

$$3.9541011 - Input_{Degree} * 0.0121524 + Input_{Weight} * 0.0005625 - Control_{Weight} * 0.1042675 - input_1 * 0.0023569 + input_{11} * 0.0051394 - input_{12} * 0.0057957 - input_{13} * 0.0122590 - input_{17} * 0.0106996 + control_1 * 0.1060109 + control_2 * 0.1035639 + control_3 * 0.1034507 + control_4 * 0.1062923 + control_5 * 0.1051447 + control_6 * 0.1061192 + control_7 * 0.1001495 + control_8 * 0.1053034 + control_9 * 0.1104407 + control_{10} * 0.1036258 + control_{11} * 0.1058865 + control_{12} * 0.0999843 + control_{13} * 0.0953821 + control_{14} * 0.1055764 + control_{15} * 0.1042613 + control_{16} * 0.1049406$$

The  $R$  square of the above model is 0.2605 which is the prediction performance of the linear regression model. It indicates that 26% of the variation in the predicted  $UWS20$  can be explained by the multivariable linear regression model. The prediction accuracy was 11.66% and it increased to 37.04%, 61.78% and 78.54% with 1 unit, 2 units and 3 units of deviation from the observed  $UWS20$  respectively. For example, if  $UWS=40$  then within 1 unit prediction,  $39 \leq UWS \leq 41$ , and 2 unit  $38 \leq UWS \leq 42$  .. etc. Appendix 3.6 shows the most effective variables on the prediction based on  $R^2$  value.

### **3.13.2 Sensitivity analysis**

To run sensitivity analysis, three datasets were randomly produced containing almost 50%, 25% and 10% of the original dataset.

#### **Running the model with 50% of the data**

After running the model with 50% of the data ( $n = 33800$ ), the following model and prediction performance of adjusted  $R$  square = 0.2587. The prediction accuracy was 11.85% and it increased to 37.25%, 62.12% and 78.62% with 1 unit, 2 units and 3 units of deviation from the observed  $UWS20$  respectively.

#### **Running the model with 25% of the data**

After running the model with 25% of the data, the following model and prediction performance of adjusted  $R$  square = 0.2560. The prediction accuracy was 11.33% and it increased to 47.23%, 60.74% and 77.15% with 1 unit, 2 units and 3 units of deviation from the observed  $UWS20$  respectively.

#### **Running the model with 10% of the data**

After running the model with 10% of the data, the following model and prediction performance of adjusted  $R$  square = 0.2643. The prediction accuracy was 11.22% and it increased to 36.81%, 62.12% and 79.03% with 1 unit, 2 units and 3 units of deviation from the observed  $UWS20$  respectively.

The sensitivity tests show that even a dataset reduced to 10% produces similar prediction capability or accuracy as the full dataset. The model does not lose its prediction performance due to loss of data, but the dataset still needs to be considerably large, more than 1% of data, for a better performing model.

As another sensitivity analysis,  $UWS19$ ,  $UWS18$  and  $UWS17$  were modelled, with similar prediction performance (adjusted R square) and accuracy.

Lastly,  $UWS20$  was modelled using around 50% of the data and the  $R - square$  was approximately 26.6% which means modelling can be done with less data and still achieve similar accuracy, which will reduce the computation complexity.

### 3.14 Conclusion

This chapter introduces the importance of randomness and the potential to use the pseudo-random number generator as a powerful tool to obtain a sequence of bits that look random with good application for cryptographic usage. In addition, it discussed using  $LFSR$  as a tool for a pseudo-random number generator and how the choice of primitive polynomial is important to gain a good security level.

It discussed the shrinking generator and the self-shrinking generator as IV-less based synchronous stream ciphers and explained how they also implement  $LFSRs$ , including a description of how they are designed and how they work.

Finally, it provided statistical test results for  $SG$  and  $SSG$  and discussed the weaknesses of both  $SG$  and  $SSG$ . These tests are valid to implement in similar ciphers, which can help improve the tests and also enhance the ciphers' strength by finding their flaws to overcome them at the design stage. In addition, the prediction for  $UWS$  was modelled using a multivariate linear regression model and will use a superior prediction method with a higher prediction rate by implementing the neural network models in Chapter 4.

The following chapter presents a new prediction and randomness method based on the proposed neural network models which had high accuracy for predicting the  $UWS$  for the binary sequences which represent the keystream for  $SG$  and  $SSG$ .

## **Chapter 4: Proposed neural network-based prediction models**

### **4.0 Chapter overview**

This chapter provides discussion about proposed neural network prediction models as a new tool for randomness tests, which were implemented using the shrinking generator and self-shrinking generator. Section 4.1 gives a general introduction, with transition from Chapter 3. Section 4.2 provides a brief introductory background on neural networks, and section 4.3 discusses the related work on neural networks and security. Furthermore, section 4.4 details neural network importance for the unique window size with neural networks. Section 4.5 details the neural network prediction model implementation, while section 4.6 provides mathematical background. Additionally, section 4.7 provides Python usage for building prediction models, section 4.8 offers results analysis, and section 4.9 concludes the chapter.

### **4.1 Introduction**

Neural networks are widely used in various fields and have applications in cryptography. However, there is a lack of research regarding measuring randomness for a given binary sequence. Because the binary sequence is the building block of any secure cryptosystem, as mentioned in Chapter 3, the keystream generated by the shrinking generator (SG) and self-shrinking generator (SSG) ciphers was used to calculate the unique window size (UWS). UWS is important for ensuring the cipher has high nonlinear complexity by using, for example, the Berlekamp–Massey algorithm [133], which works by finding the shortest feedback shift register (FSR) that can generate the same sequence. Chapter 3 also discussed how UWS is important for ensuring the cipher has high nonlinear complexity, which works by finding the shortest FSR that can generate the same sequence, such as with the Berlekamp–Massey algorithm, which is important for preventing simulations.

The neural network model proved its importance in measuring the effectiveness of the cryptosystems used in this study, especially in the field of information protection and confidentiality, where it was used as a predictive tool. It has also demonstrated its superiority as a randomised test when compared to the tests presented in Chapter 3.

Although the multilinear regression model was introduced earlier in Chapter 3, it is worth noting the importance of the knowledge it provides of the significance of independent variables on the prediction, illustrating the weaknesses in the selection of the primitive polynomials pair for the SG. However, its effectiveness with the prediction increases by adding  $\pm 1$  to the output prediction data and is even better with  $\pm 2$ . Hence, there is the need to test the possibility of more accurately predicting the UWS by applying the neural network model. Therefore, this chapter focuses on the following main questions:

1. How can neural networks be applied as a prediction method for the nonlinear complexity of a binary pseudo-random sequence?
2. How can randomness tests be applied to find weaknesses in a given stream cipher?
3. How can this study contribute to real-world applications in terms of security?
4. How can this study inspire further new directions in research?

The neural network model is enforced to predict nonlinearity and pseudo-randomness levels for given binary sequences. Being able to predict the maximum order complexity is important for identifying if the sequence has a random appearance [114].

Using the calculation of UWS as the maximum order complexity tool, which was explained in Chapter 3, can establish the measurement of pseudo-randomness for a given binary sequence due to the following:

1. Use the UWS as a maximum order complexity measurement tool for the binary sequences generated as keystreams by the SG and SSG.
2. Implement the neural network models to predict the UWS in order to evaluate the keystream (binary sequences) strength, hence, providing a solid indication of the strength of the ciphers generated by the SG and SSG as keystream(s). In addition, this method could be generalised for other ciphers' keystreams, as well as applied to their internal components, such as linear feedback shift registers (LFSRs) and nonlinear feedback shift registers (NFSRs), which can also produce a binary sequence, and then calculate the UWS and implement the neural network model for prediction. This may inspire other uses of neural network models in cryptography and in security in general, such as communications security. This would help cipher designers evaluate a cipher's efficiency in producing a highly pseudo-random keystream as well as provide users with the option of an optimal encryption method [134]. This, in turn, enhances the evaluation methods for a given binary sequence in general. The results are very promising, as they have both high levels of accuracy and a very small error margin.

## 4.2 Background

The neural network model, in principle, simulates how the human brain works by learning from collected information and data to deal with future incoming data. The neural network model uses an algorithm to learn from input data in order to predict expected outputs in the future [135]. The many applications of neural network models and varieties include those used for climate forecasting [136]; stock market price prediction [137]; health care, for forecasting diseases such as cancer [138]; image recognition [139]; pattern recognition [140]; and for production in the music industry [141].

There are two main approaches for training an artificial neural network: supervised training and unsupervised training. Through unsupervised training the network adapts its parameters in order to form a structure representing the training data. On the other hand, in supervised training, as in [135] the protection of information shown by the example of selecting secret keys, the algorithm used in the neural network model is applied by entering the given information, obtaining the output, and comparing the input and output data with the expected data. The model is then developed to reach the nearest result so that the prediction is highly accurate, with possible errors minimised, and the process continues until the model is adapted [135].

The ability of the neural network to learn by processing incoming data and producing predicted outcomes makes it attractive for applications in cryptography [142]. Because any cryptosystem has different mixes of components such as the generation of secure keystreams, protection of keys, and ensuring the security of communications, neural networks can be considered for implementation on different levels of a given cryptosystem, taking into consideration the particular cryptosystem's design and special properties. Recently, the importance of finding and calculating a UWS to measure the nonlinearity of the keystream, as explained in Chapter 3, has become apparent. Through this, the extent of the pseudo-randomness of binary sequences produced by the ciphers to be studied is known, which in this case are SG and SSG. In addition, the predictability of UWS, as mentioned in Chapter 3 and this chapter, allowed applying the neural network model as a predictor of UWS, and good results with a high accuracy ratio were obtained, as shown in Section 4.8.

This contributes to the study of binary sequences, which are essential for use in ciphers and, in turn, that extends to security in general and also contributes to research being conducted in these areas. Neural networks have great potential for predicting attacks on encryption systems or devices. It is necessary to know the importance and quality of the given data and then how to design neural network models that are able to learn and predict. Other uses of neural networks beyond establishing and sharing keys include designing symmetric ciphers [143].

### **4.3 Related work on neural network and security**

In terms of public cryptography methods, available keys are typically created using the Diffie–Hellman algorithm, developed in 1976 [144]. These keys can be used to create discrete algorithms through unsafe connection channels in public networks, but this makes it difficult for devices with limited capabilities to deal with them, especially with the choice of many keys. This led to the study of the circulation of these keys through general communication channels when using an interactive neural network model [145]. In addition, there is investigation into the implementation of an interactive neural network to produce secret keys on public channels by implementing the DES algorithm [146].

When implementing a neural network for public cryptography, two multi-layer neural networks can be used, where one model provides the defined data, and the other one learns from that data in order to establish a general classification pattern. The idea is to then make the two models interact. Although communication can be recorded by an attacker, because the encryption key was generated over public networks, it cannot be calculated [147]. The advantage is that this makes communication faster than other asymmetric encryption methods, but that does not guarantee there will not be a feasible algorithm facilitating an attack.

One of the important applications in the field of cryptography using the neural network model is the implementation of a random algorithm with a time variable for the production of a binary sequence used to cover plaintext, which has the aim of protecting the transmission of information through an unprotected network [148]. In a similar study, the aim was to use a neural network model to design a symmetric cryptosystem using a neural network model that has a matrix permutation process and classifying the data based on their randomness properties, ultimately producing a new coding technique. This has been

tested and found to be an effective system by using the permutation process through the matrix as a secret key [149].

With neural network protection systems, machine learning helped by engineering (designing specific hardware to facilitate the algorithm learning) may produce cryptographic systems that provide higher data flows, while ensuring protection by communicating with the cloud server. This is particularly critical for protecting sensitive economic or medical data, as examples of cloud server needs [150].

It is possible to design an asynchronous network model that works on two devices, where the first random weight can be used as the secret encryption key. The weight is updated only if the values of the output devices are identical. The results of a study examining this idea showed its effectiveness when using a large common key size [151], where researchers implemented a recurrent neural network, and the cipher worked in two stages, where the first stage was extending the key, and the second stage was the encryption process.

In searching for a study to facilitate the template for the web model, a classification algorithm was adopted to provide a larger framework for learning data and obtain a generalised model through testing datasets. This, in turn, is reflected through a learning phase with access to accurate classification, avoiding overfeeding of the classification. This has been applied to online combat phishing, and better results were obtained than with other methods, such as average harmonicity [152], providing more proof of the superior effectiveness of using neural network models for assessment and testing.

Implementing neural network models for prediction of the UWS means it is another measurement tool for binary sequence randomness testing and will, therefore, evaluate the level of cryptosystem using these binary sequences. In order to guarantee a good binary sequence is generated by a pseudo-random number generator, in this case SG and SSG, it must be hard to distinguish the sequence from a truly random sequence and hard to predict, while using limited computational resources. Therefore, the neural network was used to predict the pseudo-randomness of the sequence and ensure it will be indistinguishable from a truly random sequence [16]. Conversely, a cipher using a pseudo-random number generator to generate a pseudo-random binary sequence, requires that the sequence be

evaluated to make sure the cipher is secure, using a truly random number generator, which requires physical resources, making it an impractical and costly method [153].

Predicting the pseudo-randomness of binary sequences and finding approximations of its complexity are important for determining if an FSR can produce the sequence or its subsequences [114]. The lower predictability of the UWS increases the complexity of these binary sequences as well as their pseudo-randomness, and thus it will be more resistant to attacks. It will also require a substantially more complex attack because the UWS requires that the sequence be nonlinear to ensure its effectiveness [154]. Also, a stronger UWS for the binary sequence is essential for ciphers that use NFSRs, which have become more common, so deep research into these kinds of encryption systems has become more attractive to developers [155]. Also important is the nonlinear complexity profiles, which can be determined by looking at the complexity of the subsequences of a given binary sequence.

Given the NN model's efficiency in generating results with high rates of accuracy [156], it will be important to further investigate the performance of the neural network model. Examining neural network models to establish a high correlation measurement tool for the automatic sequence is also worth investigating [157]. In addition, the predictions from the UWS can be added to the most recently introduced methods to gain more insight into the pseudo-random behaviour of binary sequences [158].

#### **4.4 The importance of the neural networks for the UWS**

As already noted, UWS is a tool for measuring the strength and randomness of a given sequence to learn how to use the SG as a source for the binary sequence that is used to calculate the UWS. However, other ciphers can also be used as a source for the UWS.

The possible choice for the two LFSRs of the SG and the one LFSR for the SSG is important; hence, neural network models are useful predicting tools for learning how the UWS is predictable, meaning that the choice of an LFSR can be avoided. Because the UWS is predictable, it can easily be simulated, which leaves it vulnerable to certain kinds of attacks, such as correlation attacks [124] and divide and conquer attacks [159].

The pseudo-random binary sequences are also important because, in this case, this sequence is the keystream, and this sequence is calculated by finding the UWS. Using the

neural network models for UWS prediction shows how neural networks are a strong cryptanalysis tool. This could be considered a black-box analysis approach. Typically, the keystream generated from the cipher's LFSRs is needed. However, this could be handled by simulating the LFSR output and then applying the neural network model for predictions in order to determine generalisations about the cipher's internal components that could lead to similar kinds of attacks as discussed above.

## **4.5 Implementation**

This section introduces the proposed neural network models that were implemented for the UWS for the different degrees that resulted from the SG and SSG keystream.

### **4.5.1 Model specifications**

Assume there is a neural network model with four independent variables (input) and one dependent variable (output). A sequential model is implemented with four layers used only for data input, and no computations will be performed in these layers. There are also four hidden layers using a rectified linear unit (ReLU) [159], [160] activation function. A sequential function is used to give the model the ability to learn layer-by-layer. For the first layer, it needs to know the shape of the data, and the other layers will recognise it automatically. The first hidden layer has 50 nodes, the second hidden layer has 20 nodes, the third hidden layer has 10 nodes, and the fourth hidden layer has five nodes. There is also an output layer with one node to make the shape of the desired output (the UWS) and also to classify the predicted datasets. The learning rate = 0.0001; the optimiser used is AdaGrad [161], an algorithm for gradient-based optimisation; and the batch size is 8.

### **4.5.2 Terminology**

1. Batch size: It is difficult to feed in the entire sample at once, so it is divided into smaller parts called batches. The size of the batches is determined based upon the ability of the model to learn well.
2. Iterations: The number of batches required for one epoch.
3. Epoch: The complete dataset going through the neural network forward and backward one time.

4. Gradient descent: An iterative optimised algorithm to find the best fit for the model by processing the results multiple times until it determines the most possible optimal results.
5. Learning rate: The step size the gradient descent needs to perform to gain the optimal results.

### 4.5.3 Variables

For the SG, the independent variables are input degree, input weight, control degree, and control weight, and the dependent variable is the UWS. For the SSG, the independent variables are the two LFSR degrees and weights.

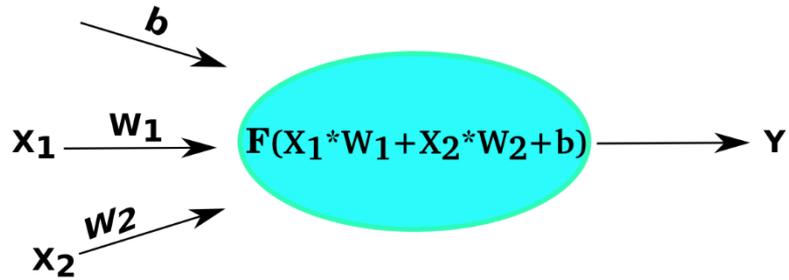
The backend used Keras [162] with TensorFlow [163], which are Python (version 3) [164] packages for data analysis. In order to use the neural network model, the UWS calculations was conducted with EC2, provided by Amazon Web Services [165]. The characteristics of the server components were based on the number of CPUs and the memory used with Linux-based instances. Ubuntu 16.04 was used to issue mathematical commands and Filezilla to investigate communication between a PC and the instance of EC2 [166] on the server. The operations were completed for different grades of UWS, and then the previous model was calculated to get the prediction for UWS.

## 4.6 Mathematical background

As the neural network has the neuron as the main data processing unit, it has a particular task of checking how data obeys a given condition:

$$Y = \sum(W_i * X_i) + b_i \quad (4.1)$$

where  $X_i$  is the input data to be checked by the neuron for correctness and is then converted to single input  $Y$ , and every input  $X$  is combined with weight  $W_i$  (the weight depends on how the  $X_i$  is important or how much influence it has on the model) to be fed to another neuron in the following layer with the associated weight. But to make it work, the activation function must decide if the neuron will activate or not. Here, ReLU was used as an activation function. An example of neuron work is illustrated in Figure 4.1, which is an example with two independent variables (inputs) of  $X_1$  and  $X_2$ , and the dependent variable (output)  $Y$ , where  $W_1$  and  $W_2$  are the weights of the independent variables. The equation will be  $Y = f(X_1 * W_1 + X_2 * W_2 + b)$ , and  $b$  is the bias vector.



**Figure 4. 1 The output of the neuron**

#### 4.6.1 Mean square error (MSE)

Mean square error (MSE) is a statistical tool used to evaluate the performance of an estimator. The lower the value of MSE, or the nearer to zero, the more accurate the model is considered to be. The MSE values can be used to compare two or more models.

If  $\hat{X}_i$  is the value of the given data prediction, where  $X_i$  is the actual value:

$$MSE = \frac{1}{n} \sum_{i=1}^n (X_i - \hat{X}_i)^2 \quad (4.2)$$

where n is the number of data inputs for the UWS simulated data.

The lower the MSE, the lower the difference between the predicted value and the actual value, and then the weight  $W_i$  are updated until the possible minimum MSE.

#### 4.6.2 Rectified linear unit (ReLU) activation function

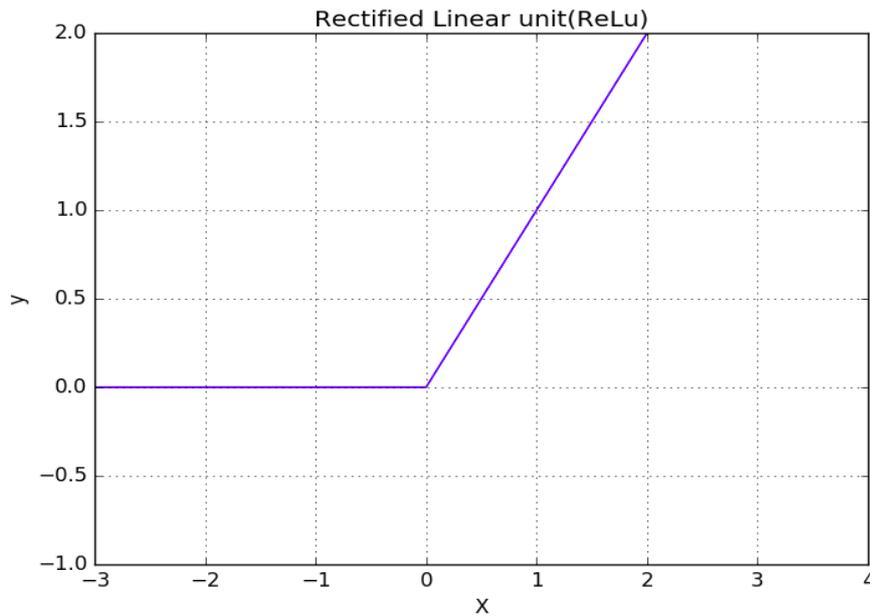
The ReLU [160] is a nonlinear activation function used in numerous models because it can be implemented widely in a deep learning network. As the formula suggests, the negative value is set to zero. While ReLU can be used for datasets that have both negative and positive values, the discrete datasets here have only positive values, which makes this activation function even more effective.

$$R(x) = \max(x, 0) \quad (4.3)$$

And based on equation (4.1),

$$R(x_i) = x \text{ if } x_i \times w_i + b > d \quad (4.4), \quad \text{and } d \text{ is a predefined value.}$$

When the neuron reaches it, the activation function  $R(x_i)$  will be decided and  $R(x_i) = x$ . Otherwise, if (4.4) is not satisfied, then  $R(x_i) = 0$ . The ReLU helps the model to transfer faster, and it is computationally uncomplicated, so it helps reduce costs. Figure 4.2 shows the ReLU graph.



**Figure 4. 2 ReLU activation function graph**

#### 4.6.3 Mathematical representation for the neural network models

For the input value  $X_i = X_1, X_2, X_3, X_4$  for SG and the independent variables, and it is a matrix of size  $N_{x \times r}$ , with  $N$  being the number of training examples and  $r$  being the number of features. In this case, this would be four features. Let  $M_{n \times m}$  be a matrix with  $n$  rows and columns. Therefore, the matrix  $X_i$  would be  $X_{N \times r} \cdot W_i$  with the weight matrices such that each row  $x$  and each column  $y$  denotes the connection of the previous layer's neuron  $X$  to the next layer's neuron  $y$ . Additionally,  $b_i$  is the bias vectors that span the total number of rows of  $W_i$ , and there are five layers, so  $i=1$  to 5. Also, let us introduce the **broadcasting** operation, or the basis vector  $b_i$ , such that it duplicates the vector for as many columns as there are in  $W_i$ , which is denoted as  $[b_i]^*$ . This allows us to add the bias vector for every example's transformed output at each layer. Also let  $h_i$  represent the raw output after each layer and let  $R(\cdot)$  represent the nonlinear activation function applied after the output of each layer. In this case, this is represented as the ReLU activation function. Applying the activation function to the raw output produces  $y_i$ , which gets sent to the next hidden layer. The above operations are chained for each hidden layer until the final output layer. Therefore:

Layer 1: Dense (100 neurons)

$$h_1 = W_1 * X_i + [b_1]^* , \quad M_{n*m} \quad \text{for} \quad W_1 = M_{100*4}, \quad X = M_{4*1}, \quad b_1 = M_{100*1}$$

Then:

$$y_1 = R(h_1)$$

Layer 2: Dense (50 neurons)

$$h_2 = W_2 * y_1 + [b_2]^* , \quad W_2 = M_{50*100}, \quad y_1 = M_{100*1}, \quad b_2 = M_{50*1}$$

Then:

$$y_2 = R(h_2)$$

Layer 3: Dense (20 neurons)

$$h_3 = W_3 * y_2 + [b_3]^* , \quad W_3 = M_{20*50}, \quad y_2 = M_{50*1}, \quad b_3 = M_{20*1}$$

Then:

$$y_3 = R(h_3)$$

Layer 4: Dense (10 neurons)

$$h_4 = W_4 * y_3 + [b_4]^* , \quad W_4 = M_{10*20}, \quad y_3 = M_{20*1}, \quad b_4 = M_{10*1}$$

Then:

$$y_4 = R(h_4)$$

Layer 5: Dense (1 neurons) → Output layer

$$h_5 = W_5 * y_4 + [b_5]^* \quad W_5 = M_{1*10}, \quad y_4 = M_{10*1}, \quad b_5 = M_{1*1}$$

$$y_5 = R(h_5) = \text{predicted UWS}$$

In general, to change the feature space of the input layer, the columns are simply subsampled to extract the relevant features, and the number of input neurons into the neural network changed.

Let us define a subset of indices  $I = \{i_1, i_2, \dots, i_p\}$  representing unique columns of the input matrix  $X$  to be extracted. To extract the index at position  $j$ , this is denoted as  $I(j)$ . Define this new input matrix as  $X'$ , and further introduce the notation of extracting out all the rows for a single column as  $X(:, i)$ , where  $i$  is the column of interest in matrix  $X$  to be extracted.

Therefore, the new matrix  $X'$  is formed such that  $X'(:, j) = X(:, (I_j))$ , for  $j=1,2,\dots,p$ . The neural network can be retrained using this feature subset stored in  $X'$ , ensuring that the number of input neurons is the same size as the number of features in this new matrix  $X'$ . The notation for propagating information forward into the network remains the same, conditionally dependent on the number of features represented in the input neuron changing to accommodate the reduced feature set.

Explicitly:

Layer 1: Dense (100 neurons)

$$h_1 = W_1 * X_i + [b_1]^* , \quad M_{n*m} \text{ for } W_1 = M_{100*|I|} , \quad X_i = M_{|I|*1}, \quad b_1 = M_{100*1}$$

Then:

$$y_1 = R(h_1)$$

$|I|$  is the *cardinality* of the set of indices  $I$ . For example, this could be 2, which represents two indices and thus two features (as in SSG). The rest of the notation for the other layers remains the same.

## 4.7 Python for model building

The Python language has a diverse open source community. The following libraries in the code, available for free, are used:

- Pandas: This library helps read a dataset and do basic data manipulation.
- Scikit-learn: This library helps split and scale the dataset to meet the requirements.
- Keras: This library helps train the dataset on the neural network architecture.

Figure 4.3 is an example of NumPy (the Python library supports multi-dimensional arrays and matrix manipulations) and Keras implementation using the formatted data and with the neural network architecture as proposed.

Figure 4.3 section 1 examines the desired value for prediction in the dataset and determines the smallest and largest values. This helps undo the preprocessing performed on the data to scale the data so that each feature has a range of  $[0, 1]$ . Figure 4.3 section 2 uses Scikit-learn to transform the data so that each feature is scaled to the  $[0, 1]$  range. The desired value for each sample in the dataset is also extracted as it is embedded with the data and is in the last column. It pulls out the column with the desired value as a separate variable and deletes the last column in the original dataset for compatibility with Scikit-learn and Keras. Figure 4.3 section 3 thus defines the neural network model in the same order of presentation for each of the layers as explained above. It defines 100, 50, 20 and 10 neurons

and 1 neuron in the same order, with each activation function being the ReLU function. The model is configured for training such that the loss is the MSE, and the AdaGrad optimiser is the parameter optimisation rule used [167]. The model is then trained using the training data and expected output values, and the network used to test accuracy on a validation dataset. Finally, Figure 4.3 section 4 uses the predicted values, unscales them to their original ranges for each feature, and calculates the average deviation for the predicted and true values in the validation dataset. Figure 4.3 section 5 measures the feature importance in the dataset. Specifically, if the columns are reshuffled and retrained on the same data, this will determine how sensitive each feature is to the learning and shows how much the output is influenced. Specifically, one pair of columns is changed while the other columns remain fixed. Features that have very little sensitivity can be safely removed from the dataset without any loss of generalisation. Features that have a greater sensitivity are quite important and contribute to the overall accuracy of the model.

```

# Section 1
rangetop = dataset['MinW'].max() #getting the max value of y
rangebot = dataset['MinW'].min() #getting the min value of x
rangel = rangetop - rangebot #getting the range of values in y

# Section 2
scaler = MinMaxScaler() #defining a scaler object
X = scaler.fit_transform(dataset) #creating a dataset with all the values with min max normalization
y = X[:,4]
#y = dataset.iloc[:,4].values #extracting the output variable from the dataset
X = np.delete(X,4,1) #extracting the input variables from the dataset

# Section 3
model = Sequential() #defining the model as sequence model -- it is used so that we can define model layer by layer
model.add(Dense(100, input_dim=4, kernel_initializer='normal', activation='relu')) #defining the len of input
dimensions, ouput neurons for hidden layer as 100, activation function as relu
model.add(Dense(50, kernel_initializer='normal', activation='relu')) # defining the hidden layer 2 with 50 nodes
model.add(Dense(20, kernel_initializer='normal', activation='relu')) # defining the hidden layer 3 with 10 nodes
model.add(Dense(10, kernel_initializer='normal', activation='relu')) # defining the hidden layer 4 with 5 nodes
model.add(Dense(1, kernel_initializer='normal', activation = 'relu')) #defining the ouput layer as 1 node
adagrad = optimizers.Adagrad(lr=0.001, epsilon=0.0001, decay=0.0)
model.compile(loss='mean_squared_error', optimizer= 'adagrad', metrics = ['mse']) #defining the loss function as mean
squared error and the learning rate optimisation technique as adam optimiser
model.fit(X_train, y_train, epochs=5, batch_size=8, callbacks=[stop_here_please], validation_data = [X_test,
y_test]) #fitting the data to train the model

# Section 4
y_pred = model.predict(X_test) #getting the prediction for the test data
y_predscaled = [int(i*rangel + rangebot) for i in y_pred] #unsaling the predictions
y_testscaled = [float(i*rangel + rangebot) for i in y_test] #unsaling the test
dev = []
for i in range(len(y_pred)):
    dev.append(abs((y_predscaled[i] - y_testscaled[i])/y_testscaled[i)
    #dev.append(abs((y_predscaled[i] - y_testscaled[i])/y_testscaled[i)
1 - sum(dev)/len(dev) #getting the accuracy number
#Average deviation from the actual to the predicted is 5.6% on average (across the test dataset)

# Section 5
perm = PermutationImportance(model, random_state=1, scoring = "neg_mean_squared_error").fit(X_test, y_test)
eli5.show_weights(perm, feature_names = dataset.columns.tolist()[1:4])

```

**Figure 4. 3 Python code using Keras for the neural network model for SG (for SSG, the input in Section 2 is changed to two inputs)**

Table 4.1 shows the model structure for SG taken from Python code, as well as some sample inputs for an SG cipher that were fed into the models in order to predict UWS24, which is shown in Table 4.3.

**Table 4. 1 Shrinking Generator unique window size 24 model summary**

Layer (Type)	Output Shape	Parameter #
dense_1 (Dense)	(None, 100)	500
dense_2 (Dense)	(None, 50)	5050
dense_3 (Dense)	(None, 20)	1020
dense_4 (Dense)	(None, 10)	210
dense_5 (Dense)	(None, 1)	11
Total parameters: 6,791		
Trainable parameters: 6,791		
Non-trainable parameters: 0		

## 4.8 Results analysis

This section reviews the results obtained for the predictions and the results for the SG and SSG ciphers.

### 4.8.1 Shrinking generator (SG) results

The SG results were obtained by implementing neural network models for UWS (20, 21, 23 and 24), as shown in Table 4.2, and with a sample of 10 inputs for UWS24 for SG, as shown in Table 4.3 for illustration.

**Table 4. 2 Shrinking generator model results for the new neural network models, including results for degrees 20, 21, 23 and 24**

	UWS20 Model	UWS21 Model	UWS23 Model	UWS24 Model
Number of layers	3	4	4	4
Number of nodes	(10, 5, 2)	(50, 20, 10, 5)	(50, 20, 10, 5)	(50, 20, 10, 5)
Learning rate	0.0001	0.001	0.0001	0.0001
MSE	0.0088	0.0138	0.0033	0.0019
Training set	56,333	196,502	713,395	770,997
Validation sample	14,084	49,126	178,349	192,750
Total parameters	6,791	6,791	6,791	6,791
Prediction percentage	96.01	94.07	95.39	95.42

Table 4.2 shows how accurate the predictions are, with accuracy around 95% and with error margin  $MSE < 0.008$  and  $MSE = 0.0019$ . UWS24 has the largest dataset, and hence, the model is better able to learn.

**Table 4. 3 Unique window size 24 chosen 10 input samples for the shrinking generator**

<b>Input degree</b>	<b>Input weight</b>	<b>Control degree</b>	<b>Control weight</b>	<b>UWS</b>
2	3	22	3	96
7	5	17	11	51
8	5	16	11	28
9	5	15	9	43
10	5	14	9	43
11	7	13	5	56
5	3	19	11	52
4	3	20	11	43
3	3	21	13	47
6	5	18	9	39

#### 4.8.2 Self-shrinking generator (SSG) results

The SSG results for neural network models degrees UWS21 to UWS25, and one neural network model for UWS from degree 4 to 20, all in one dataset, are shown in Table 4.4. The independent variables used to predict the 10 input UWS25 samples for SSG are shown in Table 4.5.

**Table 4. 4 Neural network model for a self-shrinking generator, with different unique window size degrees**

	<b>UWS model for degrees 4 to 20</b>	<b>UWS21 model</b>	<b>UWS22 model</b>	<b>UWS23 model</b>	<b>UWS24 model</b>	<b>UWS25 model</b>
Layers	4	4	4	4	4	4
Nodes	100, 50, 20, 10	100, 50, 20, 10				
Learning rate	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
MSE	0.0012	0.0014	0.016	0.0046	0.0098	0.0052
Training set	58232	67737	96025	285568	221184	1036800
Total parameters	6591	6591	6591	6591	6591	6591
Validate sample	14558	16935	24007	71392	55296	259200
Prediction percentage	96.05	96.66	89.61	90.14	97.01	97.14

**Table 4. 5 Unique window size 25 for the chosen self-shrinking generator (SSG) input sample**

Polynomial weight	<i>UWS</i>
3	51
5	51
5	47
3	47
5	45
5	46
9	51
5	50
7	46
5	46

The data for UWS21 to UWS25 was modelled in one model each and, for comparison, all data from UWS4 to UWS20 was included in one model to study the behaviour of the model when the data is merged into one dataset.

#### **4.8.3 Comparison of the shrinking generator (SG) and self-shrinking generator (SSG) results**

Although SG is weaker than SSG, as found in Chapter 3, the UWS could be predicted by implementing the neural network models for both ciphers with close accuracy, which is more evidence of the strength and effectiveness of neural networks as a predictor and measurement of the actual ciphers.

#### **4.8.4 Influences of model features**

The influence of the independent variables on the other variables for UWS24 (SG cipher) is shown in Table 4.6.

**Table 4. 6 The importance of independent variables for the neural network model using unique window size 24 for the self-shrinking generator**

Feature	Influence of feature
Input_Weight	$0.0062 \pm 0.0004$
Input_Degree	$0.0088 \pm 0.0005$
Control_Degree	$0.0171 \pm 0.0001$
Control_Weight	$0.0000 \pm 0.0000$

The analysis strongly supports that `control_degree` is the most important feature in deciding the value of the dependent variable (the influence value is highest at 0.0171), followed by `input_degree`, `input_weight` and `control_weight`. Thus, any changes to `control_degree` data (randomly reordering the data) will have the most impact on the final output of UWS prediction, producing the worst prediction. Conversely, doing the same with any other independent variable, the prediction will suffer less.

#### **4.8.5 Comparison of the neural network models and linear regression results in Chapter 3**

Comparing the results of the predictions based on the neural network models demonstrates the neural network model's superiority to the multilinear regression model in Chapter 3. In a general sense, the neural network can be used as a measurement tool for binary sequence randomness, which can help in two main research directions. First, the same models can be used with some modifications in similar ciphers. Second, this can further enhance neural network modelling for testing the appearance of cipher randomness with different methods other than testing of the keystream, as in this research.

As previously shown in Chapter 3, the maximum order complexity is an important tool for investigating pseudo-randomness behaviour. UWS can determine the minimum window size, which guarantees that every subsequence is unique. Thus, the research evaluated how the cipher generated this sequence is strong and has statistical properties that are necessary for cipher pseudo-randomness, which will help to evaluate the attack complexity required to simulate the sequence. A further step the research can take after calculating the UWS for different degrees is to predict it, which will reflect the ability to predict the cipher pseudo-randomness property.

The neural network model showed its effectiveness as a tool for predicting UWS. This has an important role for investigating the effectiveness of the random binary sequences resulting from an encryption system. Neural network models can add another randomness testing tool to help establish how strong a cipher is and whether it is sufficient to use, as done here for the SG and SSG ciphers, by using the keystream as a binary sequence. Hence, the models can be adapted for different ciphers, which is another advancement in cryptanalysis methods, as well as in security in general. The use of neural network models designed and applied to SG and SSG ciphers opens the way for their application, with some modifications, to test other ciphers.

Comparing the multilinear regression model in Chapter 3 with the neural network model in this chapter revealed that the neural network model was better than multilinear regression, which confirms its importance in the field of cryptography and security in general. The use of neural network models as a randomness measurement tool and for investigating and developing generalised models for a standard test of different cryptosystems is also worth studying.

## 4.9 Conclusion

The calculation of UWS, which is a form of maximum order complexity, shows the level of pseudo-randomness of a given binary sequence, as done in this chapter with the SG and SSG cipher keystreams as pseudo-random binary sequences. Hence, it can be a strong indicator of the level of cipher security and resistance against attacks.

By using the neural network model to predict the UWS, this chapter showed how the pseudo-randomness of a given cipher can be predicted well with the neural network model, revealing it as a new measurement tool for cipher security. As well, the neural network prediction models were far more effective in prediction than the multilinear regression models in Chapter 3.

There are several future research directions. Firstly, using UWS and neural networks for the internal cipher components of, for example, SG with two LFSRs each, can generate a binary sequence by calculating the UWS and applying a neural network as a predictor to investigate the internal ciphers' strength, which enhances the entire structure of cipher security. Secondly, using this method in other ciphers, with some modification of models, which depends on the targeted ciphers' internal structure, will expand applications. Thirdly, converting the image data into a binary sequence and then applying the neural network model to predict the outcome will help in fields such as facial recognition, which is an active research area with many applications.

The following chapter demonstrates the implementation of the MICKEY 2.0 cipher in mobile cloud computing. SG and SSG and the MICKEY 2.0 cipher are all lightweight synchronous stream ciphers suitable for small hardware devices such as RFID tags and microprocessors. The differences include their structures, where SG and SSG are IV-less

stream ciphers, and MICKEY 2.0 is an IV-based stream cipher. MICKEY 2.0 is more secure than SG and SSG, and the existing body of literature reports a number of attacks on SG and SSG. For MICKEY 2.0, the main attack is a differential fault attack [168]. In addition, SG and SSG show weakness based on the statistical analysis and the results in Chapter 3 and this chapter, but there have been no successful statistical attacks with any complexity against MICKEY 2.0, as shown in the next chapter.

# **Chapter 5: Proposed lighter and faster MICKEY 2.0 reduced variant for low cost implementations**

## **5.0 Chapter overview**

This chapter is devoted to the thesis proposed lightweight secure protocol. Section 5.1 offers a general introduction for lightweight encryption methods, with security challenges, Section 5.2 provides optimisation methodology, Section 5.3 provides the randomness test designed by the US National Institute of Standards and Technology (NIST), Section 5.4 provides the power consumption estimator, Section 5.5 provides the MICKEY cipher family design principles, Section 5.6 provides the reduction process, Section 5.7 provides the NIST tests results, Section 5.8 provides the proposed cipher performance test, Section 5.9 provides the power consumption testing, Section 5.10 presents the cryptanalysis for the proposed cipher, Section 5.11 discusses the results and analysis, and Section 5.12 concludes the chapter.

## **5.1 Introduction**

Transferring information over different networks, especially insecure networks, using mobile devices and RFID technology must overcome security issues to maintain data security. Not all encryption methods are suitable in these situations as some methods require high computation power, storage and power consumption. Lightweight encryption methods must be implemented, where lightweight means smaller size and power use. A greater range of lighter encryption methods are needed for even smaller devices such as tiny microcontrollers like those with 16 bit and smaller devices with less computational capability which also need to consume less power. With advances in IoT technology and networking, there is a pressing need for lighter secure encryption to be used in the devices used in such technologies such as the small 16-bit Raspberry Pi.

To design secure and lighter ciphers based on secure existing ciphers, MICKEY 2.0 is chosen as a base as it has more resistance to attacks, and performance and high throughput. This chapter proposes a lighter MICKEY 2.0 based version, named MICKEY 2.0.85, based on the internal registers length, to meet the need for less power consumption, and fewer

gate equivalents (GEs) to be more suitable for smaller devices and reduce the overall cost as the chip size will be small and cheap.

The challenge is how to develop lighter ciphers while maintaining a high level of security. The efficiency and security of a new cipher are evaluated by using randomness tests to test security and methods to test the speed of keystream generation and power consumption and then comparing results to existing ciphers.

The pseudo-randomness of the binary sequences, which is the keystream generated by the cipher, is an important part of cipher security. In study[169] provides an important insight linking the authentication and encryption with IoT device communications by implementing an artificial intelligence approach to ease authentication management. In another study[170] provides further insight about the security challenges on small devices and in IoT with communications networks including cloud computing.

Lightweight stream ciphers have been an active research area for the last 20 years, due to the increasing usage of network communications which use some small hardware within their main components [171].

The number of gate equivalents (GEs) is important in hardware structure, influencing both performance and power consumption. Smaller devices have fewer GEs. Therefore, designing new ciphers or even optimising or modifying existing ciphers to be adaptable for such devices is challenging. In designing encryption methods for small devices, designers must consider the possible required number of GEs [172].

Optimising the popular AES cipher for use in IoT technology has been a focus in recent literature. For example, [173] introduced a 32-bit AES for implementation in small devices by tweaking the S-box structure to reduce the original required size of the device hardware by 20%. Another cipher “PRESENT” [174], which is a block cipher targeted at IoT technology and sensor networks which requires fewer GEs, aims to introduce an alternative to AES and Data Encryption Standard (DES), however, some cryptanalysis and recent attacks based on linear attacks, as shown in [175], were able to establish an attack on PRESENT in round 28 with key = 80-bit and key = 128-bit. In study [176] which provides an overview of the recent investigation of lightweight cryptographic methods.

The chapter makes several contributions. First, it proposes a lighter version of the MICKEY 2.0 cipher and tests it for pseudo-randomness. Second, it shows how the algorithm was altered for new MICKEY 2.0.85, with fewer GEs. Third, it shows by tests how MICKEY 2.0.85 is faster and consumes less power than MICKEY 2.0, with a passing rate which is quite high and slightly better than MICKEY 2.0. Fourth, by applying cryptanalysis methods it shows that the MICKEY 2.0.85 cipher is resistant to attacks. Methods for reducing power and time can inspire more research in security applications. This lightweight stream cipher with enhancements will be more efficient for use in IoT technology, including RFID and near field communication.

This chapter presents and evaluates the novel proposed and lighter secure version of MICKEY 2.0, named MICKEY 2.0.85 based on the internal registers' length. MICKEY 2.0 is a lightweight synchronous stream cipher with good throughput, fast encryption and suitable for hardware security, and the design and overall concept are presented in this chapter.

## **5.2 The proposed cipher and the design optimisation methodology**

In order to find an optimal lightweight encryption system, this study follows the best scientific methodology to ensure the validity of the study framework. Several experiments are required to select the best MICKEY 2.0 interior possible modifications. Randomness tests were performed to confirm the validity of the proposed variant and the level of confidentiality that it can provide was also analysed.

### **Study hypothesis**

The initial hypothesis which proposed an optimised version of MICKEY 2.0 which is lighter and faster and still has randomness property is achievable. The reduction is in the two internal shift register sizes. To achieve the optimised version, the research questions are:

1. How can MICKEY 2.0 ciphers be optimised, and how can lighter versions be proposed to avoid shortcomings in implementation in small devices?
2. How can cryptanalysis be tested to provide sufficient confidence in the proposed novel MICKEY 2.0 reduced variant cipher to ensure validity for use?

This study used the following methodology.

1. The number of bits were reduced by 15 bits for both MICKEY 2.0 registers, so the new internal state was reduced from 200 bits (100 bits for each register) to 170 bits. The reason and the process are explained in the algorithm in Section 5.6.
2. A suite of randomness tests suite designed by the US National Institute of Standards and Technology (NIST) was implemented to evaluate the required statistical properties of the reduced version. The passing rate was calculated to compare the new version with MICKEY 2.0 and MICKEY 1.0.
3. Power consumption was estimated by using Xilinx Power Estimator (XPE) [177] to see how MICKEY 2.0.85 can consume less power. MICKEY 2.0.85 was compared to MICKEY 2.0 and another reduced variant of other ciphers such as the Trivium cipher (Micro-Trivium).
4. C code was used to test the encryption speed for MICKEY 2.0.85, compared to MICKEY 2.0.
5. Randomness, power consumption and encryption speed for MICKEY 2.0.85 were compared to the original MICKEY 2.0, to evaluate the overall performance and security of MICKEY 2.0.85. The proposed version should be lighter, faster, require fewer GEs and consume less power, and pass the randomness tests.
6. Levenshtein Distance and Cosine similarity attacks were used to show how MICKEY 2.0.85 is resistant against statistical attacks.

### **5.3 NIST randomness test**

The NIST suite of 15 different randomness tests [27] is carefully designed to catch any biases in the sequences that need to be tested. These tests are still trusted for randomness testing as NIST determined the standard requirements for encryption methods, which are sufficient for security evaluation based on passing the randomness tests [178].

NIST tests provide a standard test customised for lightweight cryptography methods including authentication, hash functions, ciphers and data management, as well as the hardware implementation guidance [179]. Cipher designers need to take into consideration the pseudo-randomness of the keystream, which needs to satisfy the statistical properties to ensure the cipher is a pseudo-random number generator. The NIST test suite plays an important role [178], because if a cipher passes these tests it means the cipher has met the

requirements to be valid for use. For instance, [180] test a chaotic system based cipher which consists of two NFSRs, by implementing NIST tests to measure the validity of their cipher which targeted small devices.

Another study [181] implemented multiple pseudo-random number generator sources, by taking their keystream and altering them to be suitable for IoT small devices, and their NIST test results provided a good success rate. NIST randomness tests help improve IoT technology, as the tests can evaluate security, based on the encryption methods. [182] used NIST tests to analyse the pseudo-randomness of some chosen algorithms, to evaluate their performance based on encryption speed, and to measure the performance using different microcontrollers.

## **5.4 Power consumption**

With increasing growth in small devices such as mobile phones, as well as increasing dependency on wireless communications which rely on small microcontrollers and RFID technology, security remains a massive challenge. Lightweight encryptions that consume less power which are adequate for such constrained technology and proposed lightweight ciphers are needed to satisfy the power consumption limitations. Therefore, it is important to use a power estimator to measure cipher usage. This study used Xilinx Power Estimator (XPE) [177] which is a useful tool to estimate power consumption. Many studies have implemented this estimator, including [183].

## **5.5 MICKEY 2.0 internal design**

MICKEY 1.0 was first submitted by the designers Babbage and Dodd to the eSTREAM project [184]. It is designed for use in hardware as it is a lightweight synchronous stream cipher, however it can also be implemented in software. The key and IV are both 80-bit, and 80-bit for each R (Linear register) and S (Nonlinear register). It is described in full in [184]. However, In [185] found that there is a detectable weakness in the state convergence. The designers then developed a stronger version, MICKEY 2.0 [186], to solve this issue. The MICKEY 2.0 [186] cipher consists of two shift registers with a length of 100-bit each, R (Linear register) and S (Nonlinear register). Each stage in the internal state contains only one bit. Figure 5.1 summarises the MICKEY family cipher design. The (Key, IV) loading is indirect as the mixing is done before clocking, which is different from

ciphers such as Trivium that allow direct loading, therefore, there is a pre-clocking phase before loading into registers.

### Loading bits

The MICKEY 2.0 cipher accepts 80-bit for key(K), and 80-bit and for initialisation vector(IV).

Let C= ciphertext, P = plaintext and Z = keystream

$$K=K_0,K_1,K_2,\dots,K_{79}$$

$$IV=IV_0,IV_1,IV_2,\dots,IV_{79}$$

$$Z=Z_0,Z_1,Z_2,\dots,Z_{79}$$

$$C= Z \text{ xor } P \pmod{2}$$

Every (K,IV) generates up to  $2^{40}$ bit (maximum length). The length of K and IV should be the same. It is possible to reuse the same K. However, it is not acceptable to reuse the IV with the same K.

**R register clocking:** Clock\_R=(R, INPUT\_bit\_R, CLOK\_bit\_R). Figure 5.3 and Section 5.6 provide an explanation.

**S register clocking:** S register internal structure consists of controlling components: COMP0, COMP1, FB0 and FB1, each of 100 states, and controlling taps (positions) as in tables (5.1-5.10) where if there is 1 that is the bit control position. Figure 5.4 and Section 5.7 provide an explanation. Figure 5.1 summarises the general structure of the MICKEY family cipher.

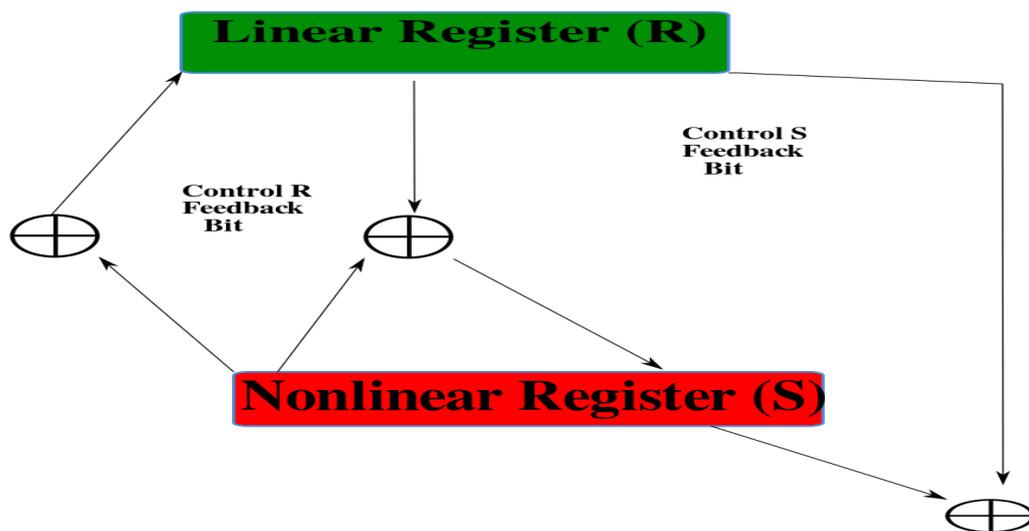


Figure 5. 1 MICKEY cipher family general internal design

The principle of the MICKEY family of ciphers is that the CLOCK\_KG drives both CLOCK\_R and CLOCK\_S in order to perform the XOR operation for the bits positions that form the keystream, as shown in Figure 5.2.

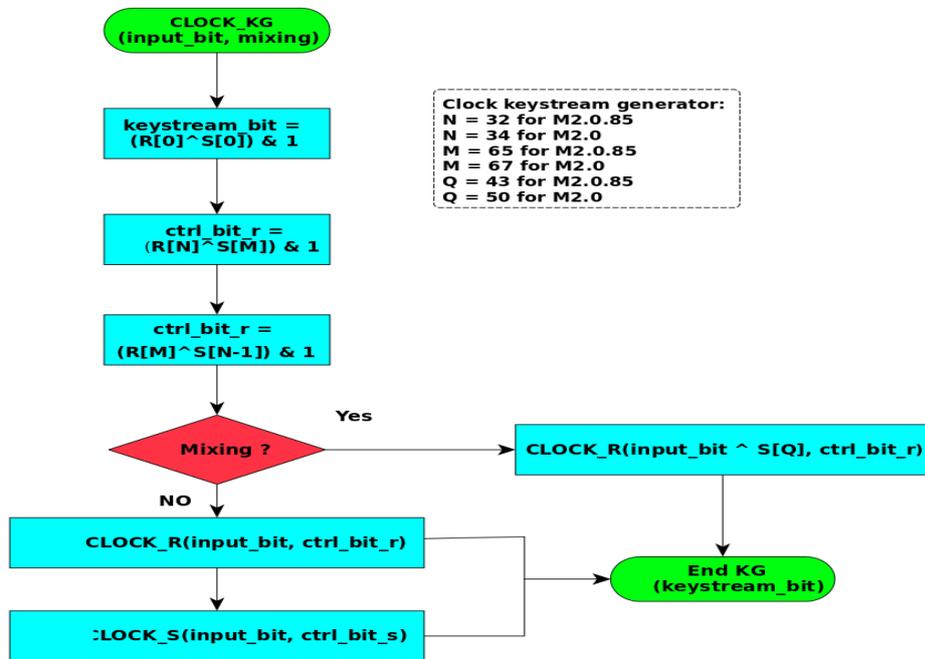
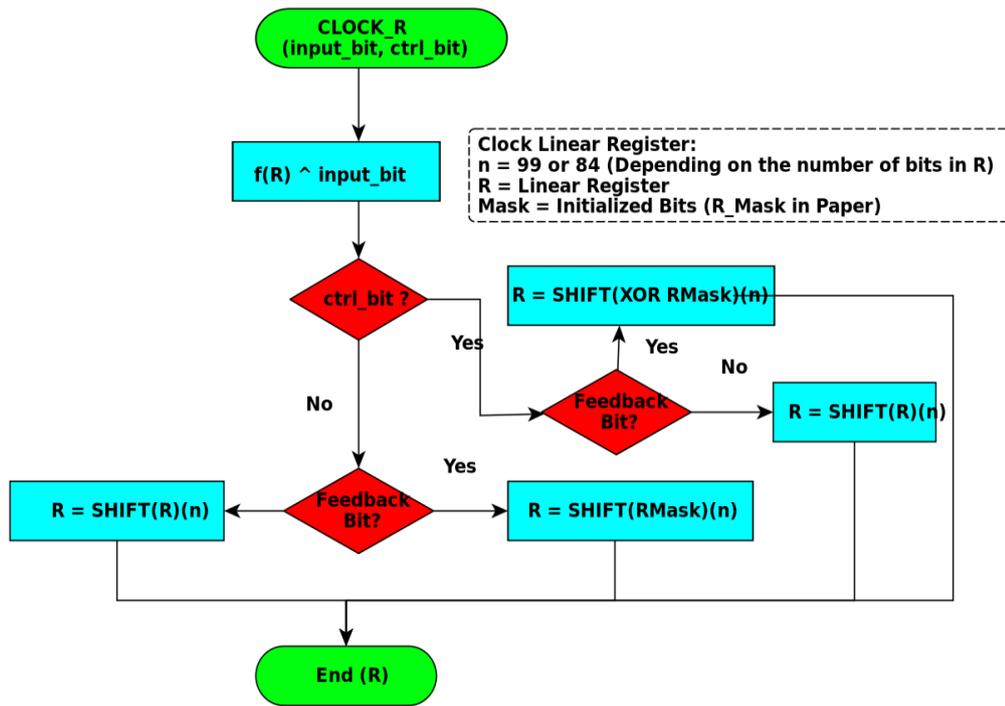


Figure 5. 2 The core logical process for the MICKEY cipher family

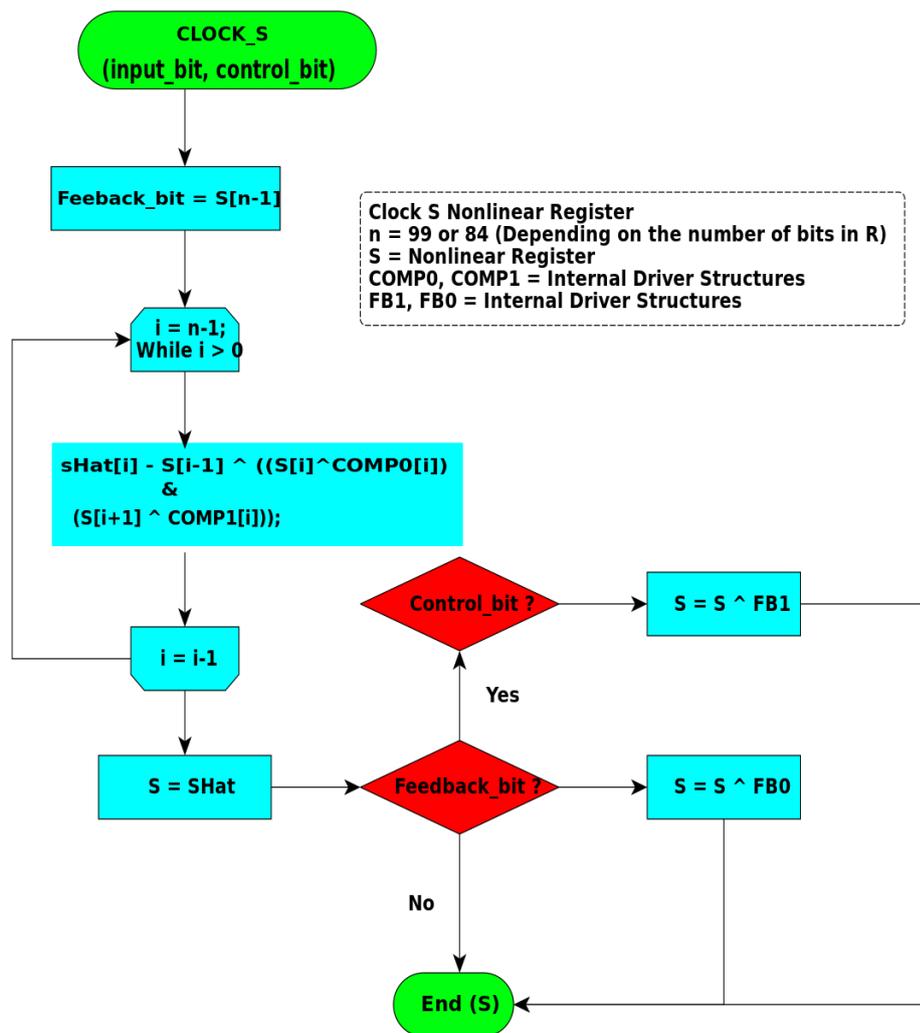
CLOCK\_KG produces the keystream bit by performing the XOR operation on the first bit of the register R (R[0]) with the first bit that comes from the register S (S[0]). The bits positions are advanced for both R and S, by clocking both CLOCK\_R and CLOCK\_S functions. In the initialisation stage there is mixing of the key and the IV bits by CLOCK\_R in such a random way. Figure 5.2 shows the selected values of (N,M,Q) which are selected to be scattered through R and S. The mechanism to advance the register R is shown in Figure 5.3.



**Figure 5. 3 Process flow for the linear register (R)**

Figure 5.3 shows the flowchart for the CLOCK\_R clocking in order to mix the bit position of the register R, by determining if the current XOR bit is 0 or 1 according to the previous bit and the position of the taps on the R\_MASK. For MICKEY 2.0.85, the number of bits of both R and the taps on R\_MASK were reduced and reorganised as shown in Section 5.6.

For the register S, the principle of clocking can be seen in the flowchart in Figure 5.4. The bits positions are determined in a more random way, determined based on the current state of S. The mixing parts of S are the functions COMP0, COMP1, FB0 and FB1 which ensure the random appearance of S bits.



**Figure 5. 4 Process flow for the nonlinear register (S)**

Figure 5.4 shows that for the nonlinear register S the way of clocking is by the CLOCK\_S function to control the bits positions in a very mixed way to ensure bits confusion among the S register. The CLOCK\_S function implements internal controlling randomiser sequences which are COMP0, COMP1, FB0 and FB1, which add more complexity to the output of the S register, which also works to randomise the bits generated by the R register. Therefore, the S register as a whole adds more complexity to the R register output, by adding more nonlinear complexity of the keystream generated by the MICKEY family of ciphers.

To be more precise, within the CLOCK\_S the functions COMP0 and COMP1 are the controlling randomisers for the S register bits that control bits from FB0 and FB1. Furthermore, the sequences FB0, FB1, COMP0 and COMP1 bits arrangement were done carefully and expertly chosen in such a way to have appropriate positions, which results in

randomness of both the S output and the keystream. Section 5.6 provides more detail on the FB0, FB1, COMP0 and COMP1 precise arrangements.

To ensure the randomness of the keystream, the initial design by the MICKEY 2.0 designers [186] was done by ensuring the initial state of the internal clocking functions and controlling sequences guaranteed the appearance of randomness as much as possible. For the proposed MICKEY 2.0.85 cipher in this thesis, the bits in the internal structure were adjusted by multiple experiments with a target to avoid any biases or predictability. As there were multiple bits selections and arrangements, many trials were conducted on the NIST randomness tests. The experiments were repeated until an optimal bits arrangement was achieved which passed the NIST tests.

## **5.6 Reduction process**

To explain the thesis reduction approach and how reductions were achieved, the MICKEY 2.0 algorithm is described, together with the MICKEY 2.0.85 algorithm.

### **5.6.1 MICKEY 2.0 and MICKEY 2.0.85 algorithms**

**(Some contents of the this section were published as a part of this thesis in [23])**

In the MICKEY family of ciphers, the basic concepts of producing the keystream rely on clocking operations. Clocking is the main part of the algorithm that drives the rest of the functions, including the clocks functions in both registers R and S which change the state of bit positions separately, or together considering each register's current state. For the initial inputs, the position of controlling bits either changes the bits positions by performing the XOR operation, or shifts it based on the specific bit in the register. Critically, these operations are performed in such a way to ensure the positions change rather unpredictably.

The complexity of the hardware electronic circuits is determined by GEs. If the number of GEs is large, that will affect the speed performance and consume more power. Thus, reducing the number of GEs will improve overall performance of lightweight devices. Appendix 5.1 for MICKEY 2.0 and Appendix 5.2 for MICKEY 2.0.85 illustrate the GE counting approach and the reduction of GEs for MICKEY 2.0.85, and compare both ciphers. The following is an algorithmic description for MICKEY 2.0 and MICKEY 2.0.85:

### **Algorithm 5.1: CLOCK\_R**

The **CLOCK\_R** function forwards the linear register.

The **CLOCK\_S** function forwards the nonlinear register.

The **CLOCK\_KG** function generates a keystream by joining both **CLOCK\_R** and **CLOCK\_S**.

The function **CLOCK\_R** advances the position of the linear register and determines whether or not the current XOR mask bit is 1 or 0 based on previous operations and the **R\_MASK**.

**CLOCK\_R**: 1: Initialisation of the Internal Register (Single XOR)

**CLOCK\_R** 2: Loop (Conditional) Feedback Bit Logically Assigns Linear Register Bit (Within Loop)

MICKEY 2.0: for i = 0 to 99

MICKEY 2.0.85: for i = 0 to 84

**CLOCK\_R** 3: Linear (**R\_MASK**) Logic to Invert Bit (Single XOR) (Within Loop)

MICKEY 2.0: for i = 0 to 99

MICKEY 2.0.85: for i = 0 to 84

**CLOCK\_R** 4: Multiple Related Operation (Single MUX) – Conditionally executed based on control bit.

**CLOCK\_R** 5: Multiple Related Operation (Single MUX) – Conditionally executed based on feedback bit.

With the nonlinear register (**S**), the correspondence between bit and position is more arbitrary, but it is driven by the state of the linear register. The internal data structures **COMP0**, **COMP1**, **FB0** and **FB1** are also arbitrary (random) bits.

The function **CLOCK\_S** advances the position of the nonlinear register. Using four internal and random structures (**COMP0**, **COMP1**, **FB0** and **FB1**) the current bit position is far more nonlinear.

### **Algorithm 5.2: CLOCK\_S**

**CLOCK\_S**: 1 Initialisation of Internal (Nonlinear) Register (Single XOR)

MICKEY 2.0: For i = 0 to 99

MICKEY 2.0.85: For i = 0 to 84

**CLOCK\_S**: 2, **CLOCK\_S**: 3: Bitwise operations on internal structures 3 XORs and One AND (gates)

**CLOCK\_S**: 4: Conditional Logic on Feedback and Control Bit (Single MUX)

MICKEY 2.0: For i = 0 to 99

MICKEY 2.0.85: For  $i = 0$  to 84

**CLOCK\_S**: 5: Change Nonlinear Register (Single XOR)

**CLOCK\_KG** invokes **CLOCK\_R** and **CLOCK\_S** for the purpose of determining the appropriate XOR bit for each bit position within the generated keystream.

**CLOCK\_KG**: 1-5: Simple Initialisations: (4 XOR, 1 AND)

The IV and key were used along with the internal masks to initialise the registers in the function ECRYPT\_keysetup, ECRYPT\_ivsetup. By arbitrarily mixing the bits of the key and the IV, the initial state of both the linear and nonlinear registers will be unpredictable.

MICKEY 2.0: IV\_i 1: For  $i = 0$  to 79: Initialise on IV (Single MUX)

MICKEY 2.0: IV 2: For  $i = 0$  to 80: Initialise on Key (Single MUX)

Therefore, the MICKEY algorithm works as follows:

### **Algorithm 5.3: MICKEY algorithm**

MICKEY 2.0: Process (Single MUX to represent Logic):

1. Initialise the internal state using: IV, key and **CLOCK\_KG** (which uses **CLOCK\_R** and **CLOCK\_S**) to mix in the IV and key bits based on the internal driver structures. (**R\_MASK** and **COMP0**, **COMP1**, **FB0**, **FB1**)
2. For each bit in the message invoke **CLOCK\_KG**.
  - a. **CLOCK\_KG** invokes **CLOCK\_R**, which advances the linear bit and masks it with **R\_MASK** to determine its final value.
  - b. **CLOCK\_KG** also invokes **CLOCK\_S**, which may or may not advance the nonlinear bit depending on the linear position and the values of (**COMP0**, **COMP1**, **FB0** and **FB1**).
  - c. **CLOCK\_KG** determines the keystream bit by XORing the current linear and nonlinear registers and ANDs them with 1.
  - d. *Ciphertext Generation*: The current plaintext message bit is XORed with the current keystream bit, which becomes the ciphertext output [23].

Tables 5.1–5.10 show the controlling bits position for R\_mask, COMP0, COMP1, FB0 and FB1 for ciphers MICKEY 2.0 and MICKEY 2.0.85, to highlight the change of bit positions for MICKEY 2.0.85 from MICKEY 2.0

**Table 5. 1 MICKEY 2.0 R\_mask**

BIT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Value	1	1		1	1	1	1			1			1	1			1			1
BIT	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
Value	1	1	1			1			1									1	1	
BIT	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59
Value		1	1			1	1				1		1		1		1		1	
BIT	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
Value	1	1		1	1	1	1	1				1	1							1
BIT	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99
Value	1	1	1					1	1	1	1	1	1		1	1	1	1		

**Table 5. 2 MICKEY 2.0.85 R\_mask**

BIT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Value	1	1		1	1	1	1			1			1	1			1			1
BIT	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
Value	1	1	1			1			1										1	1
BIT	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59
Value		1	1			1	1					1		1		1	1		1	1
BIT	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
Value	1	1	1				1	1					1	1	1	1	1	1		1
BIT	80	81	82	83	84															
Value	1	1	1																	

**Table 5. 3 MICKEY 2.0 COMP0**

BIT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Value					1	1				1		1	1	1	1		1			1
BIT	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
Value		1		1		1			1		1	1		1		1				
BIT	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59
Value				1		1			1						1		1			1
BIT	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
Value	1	1	1			1			1		1	1	1	1	1	1	1	1		1
BIT	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99
Value		1	1	1	1	1	1		1		1								1	1

**Table 5. 4 MICKEY 2.0.85 COMP0**

BIT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Value					1	1				1		1	1	1	1		1			1
BIT	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
Value		1		1		1			1		1	1		1		1				
BIT	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59
Value				1		1			1		1	1	1		1		1		1	1
BIT	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
Value	1	1	1	1	1		1	1	1	1	1	1		1		1				
BIT	80	81	82	83	84															
Value			1	1																

**Table 5. 5 MICKEY 2.0 COMP1**

BIT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Value		1		1	1			1		1	1	1				1		1		
BIT	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
Value		1	1		1		1	1	1		1	1	1	1				1	1	
BIT	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59
Value	1		1	1	1					1				1		1	1	1		
BIT	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
Value		1	1	1	1	1	1		1		1	1	1		1	1	1	1		
BIT	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99
Value		1					1	1	1				1			1	1			

**Table 5. 6 MICKEY 2.0.85 COMP1**

BIT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Value		1		1	1			1		1	1	1						1		
BIT	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
Value		1	1		1		1	1	1		1	1	1	1				1	1	
BIT	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59
Value	1		1	1	1					1				1		1	1	1		
BIT	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
Value		1	1	1	1	1	1		1		1	1	1		1					
BIT	80	81	82	83	84															
Value	1	1																		

**Table 5. 7 MICKEY 2.0 FBO**

BIT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Value	1	1	1	1		1		1	1	1	1	1	1	1	1			1		1
BIT	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
Value	1	1	1	1	1	1	1	1	1			1	1							1
BIT	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59
Value	1	1			1			1		1		1			1		1	1	1	1
BIT	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
Value		1		1		1										1	1		1	
BIT	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99
Value			1	1		1	1	1			1	1	1			1	1			

**Table 5. 8 MICKEY 2.0.85 FBO**

BIT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Value	1	1	1	1		1		1	1	1	1	1	1	1	1			1		1
BIT	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
Value	1	1	1	1	1					1	1	1			1			1		1
BIT	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59
Value		1			1		1	1	1	1		1		1		1				
BIT	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
Value								1	1		1	1	1			1	1	1		
BIT	80	81	82	83	84															
Value	1	1																		

**Table 5. 9 MICKEY 2.0 FB1**

BIT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Value	1	1	1		1	1	1					1	1	1		1			1	1
BIT	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
Value				1			1	1			1		1	1				1	1	
BIT	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59
Value					1	1		1	1				1				1			1
BIT	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
Value			1		1	1		1		1			1		1				1	1
BIT	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99
Value	1	1		1	1	1	1	1							1					1

**Table 5. 10 MICKEY 2.0.85 FB1**

BIT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Value	1	1	1		1	1	1					1	1	1		1			1	1
BIT	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
Value				1			1	1			1		1	1				1	1	
BIT	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59
Value					1	1		1	1				1				1			1
BIT	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
Value			1		1	1	1		1	1	1	1	1				1			1
BIT	80	81	82	83	84															
Value				1																

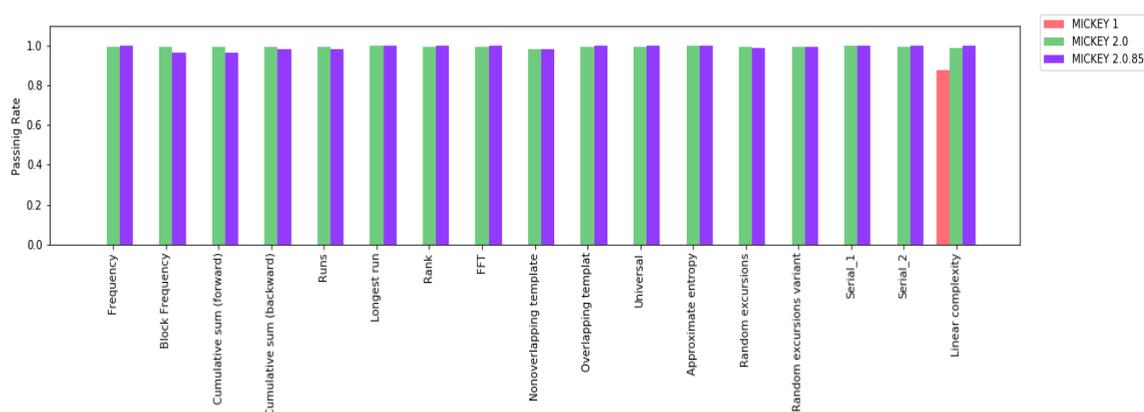
## 5.7 Results of NIST tests

This section presents results of the NIST tests for MICKEY 2.0, MICKEY 2.0.85 and MICKEY 1.0. MICKEY 1.0 failed 14 of 15 tests for the keystream and passed only one test of 15 for the ciphertext randomness test. The results confirm MICKEY 2.0.85 is secure as it shows a high level of randomness based on NIST tests.

### 5.7.1 NIST test results for the keystream

MICKEY 2.0 is a good encryption method and has a higher security level than other popular ciphers such as Trivium and Grain. It has more resistance to algebraic attack and fault analysis attack, for example [168]. MICKEY 2.0’s good attack resistance is due to the irregular mixing in the bits in the internal state. MICKEY 2.0 was used as a reference to measure the proposed reduced variant. This research used 410 (key, IV) difference pairs to generate 410 different keystream sequences with length  $10^6$  bits for both MICKEY 2.0 and 410 sequences for MICKEY 2.0.85 ciphers for comparison, as MICKEY 2.0.85 needs to have a similar security level as MICKEY 2.0.

By applying the NIST pseudo-randomness test suite as shown in Table 5.11 and Table 5.12, with 410 sequences and each sequence of length  $10^6$  bits, MICKEY 2.0.85 has a slightly better passing rate than MICKEY 2.0 with almost 100% in some of the tests, and a passing rate very close to 100% in the rest of the tests. Comparing the results in Table 5.11 and Table 5.12 with MICKEY 1.0 as shown in Figure 5.5, MICKEY 1.0 only passed the linear complexity test.



**Figure 5.5 Comparison histogram of NIST test passing rates for MICKEY 2.0, MICKEY 2.0.85 and MICKEY 1.0**

To further test MICKEY 2.0.85, 1350 sequences with length  $10^6$  bits each were generated. Table 5.13 shows the passing rate is high which confirms the security level of MICKEY 2.0.85 and shows it has a good statistical randomness appearance. The NIST test results can be used as a standard indicator for the validity of the ciphers to be used.

The results of running NIST randomness tests on MICKEY 1, MICKEY 2.0 and MICKEY 2.0.85 ciphers showed that MICKEY 1 failed all tests except the linear complexity test with p-value = 0.162606 and a passing rate of 100%. Table 5.11 and Table 5.12 show that MICKEY 2.0 and MICKEY 2.0.85 passed all tests, with better results for MICKEY 2.0.85 for tests Rank, FFT, Frequency, Overlapping Template, Universal and linear complexity with a pass rate of 100%.

To optimise the cipher structure, the reductions of the versions were modified and every version was tested on the NIST tests to derive the most optimal possible secure version that has an optimal passing rate with the lowest number of GEs.

**Table 5. 11 MICKEY 2.0.85: 410 sequences, each sequence with a length of  $10^6$  bits**

Tests	Min P-value	Max P-value	Average	Proportion
Frequency	0.122325	0.739918	0.377364	1
Block Frequency	0.350485	0.534146	0.381095	0.966667
Cumulative Sum (Forward)	0.122325	0.911413	0.663548	0.966667
Cumulative Sum (Backward)	0.739918	0.739918	0.739918	0.983333
Runs	0.350485	0.534146	0.381095	0.983333
Longest Run	0.350485	0.350485	0.350485	1
Rank	0.122325	0.534146	0.228988	1
FFT	0.213309	0.534146	0.289644	1
Non-Overlapping Template	0.458868	0.500555	0.466752	0.98705
Overlapping Template	0.066882	0.739918	0.627745	1
Universal	0.122325	0.534146	0.259599	1
Approximate Entropy	0.534146	0.911413	0.785657	1
Random Excursions	0.574584	1	0.726513	0.988839
Random Excursions Variant	0.499518	0.929764	0.700181	0.996032
Serial 1	0.122325	0.213309	0.137489	1
Serial 2	0.350485	0.739918	0.446001	1
Linear Complexity	0.122325	0.911413	0.269004	1

**Table 5. 12 MICKEY 2.0: 410 sequences, each sequence with a length of  $10^6$  bits**

Tests	Min P-value	Max P-value	Average	Proportion
Frequency	0.035174	0.911413	0.397568	0.997561
Block Frequency	0.122325	0.911413	0.428437	0.997561
Cumulative Sum (Forward)	0.066882	0.911413	0.788651	0.997561
Cumulative Sum (Backward)	0.017912	0.911413	0.68968	0.997561
Runs	0.066882	0.911413	0.400601	0.995122
Longest Run	0.122325	0.739918	0.373471	1
Rank	0.122325	0.911413	0.273754	0.997561
FFT	0.122325	0.911413	0.323092	0.995122
Non-Overlapping Template	0.451814	0.528983	0.473251	0.986487
Overlapping Template	0.066882	0.911413	0.639514	0.995122
Universal	0.066882	0.911413	0.29319	0.995122
Approximate Entropy	0.213309	0.991468	0.815439	1
Random Excursions	0.679305	1	0.899804	0.995427
Random Excursions Variant	0.598409	1	0.867777	0.996296
Serial 1	0.066882	0.911413	0.271612	1
Serial 2	0.122325	0.911413	0.397487	0.995122
Linear Complexity	0.008879	0.911413	0.247633	0.992683

**Table 5. 13 MICKEY 2.0.85: 1,350 sequences, each sequence with a length of  $10^6$  bits**

Tests	Min P-value	Max P-value	Average	Proportion
Frequency	0.004301	0.911413	0.247507	0.932593
Block Frequency	0.035174	0.991468	0.408853	0.999259
Cumulative Sum (Forward)	0.008879	0.911413	0.475431	0.931852
Cumulative Sum (Backward)	0.066882	0.991468	0.438661	0.931852
Runs	0.017912	0.911413	0.350936	1
Longest Run	0.035174	0.991468	0.569959	0.997778
Rank	0.008879	0.991468	0.562409	1
FFT	0.017912	0.911413	0.334836	0.997778
Non-Overlapping Template	0.45673	0.544193	0.506632	0.99076
Overlapping Template	0.066882	0.991468	0.684628	0.996296
Universal	0.035174	0.911413	0.444997	0.997037
Approximate Entropy	0.035174	0.911413	0.513536	0.998519
Random Excursions	0.545207	1	0.724354	0.994351
Random Excursions Variant	0.438734	1	0.717153	0.972019
Serial 1	0.017912	0.991468	0.538686	0.999259
Serial 2	0.035174	0.991468	0.313621	0.995556
Linear Complexity	0.008879	0.911413	0.614827	0.999259

### 5.7.2 NIST test results for the ciphertext

Tables 5.11, 5.12 and 5.13 summarised the results for the keystream generated by the ciphers. It is also useful to test the pseudo-randomness of the ciphertexts generated by those ciphers, as there are specific ciphertext attacks [187], [188]. The results for MICKEY 1.0, MICKEY 2.0 and MICKEY 2.0.85 shown in Table 5.14 and Table 5.15, using 13 MB bits for each cipher for comparison, shows how the proposed cipher has good pseudo-randomness properties.

By running NIST randomness tests on MICKEY 1.0, MICKEY 2.0 and MICKEY 2.0.85 with 13 MB ciphertext files for each cipher, MICKEY 1.0 failed all tests except the linear complexity test with p-value = 0.162606 and pass rate of 100%. Table 5.14 and Table 5.15 show MICKEY 2.0 and MICKEY 2.0.85 passed all tests, with better results for MICKEY 2.0.85 compared to MICKEY 2.0 for the Rank, FFT and Serial tests with a pass rate of 100%.

**Table 5. 14 MICKEY 2.0 NIST test results for ciphertext with 13 MB bits length**

Tests	P-value average	Proportion
Frequency	0.090936	1
Block Frequency	0.637119	1
Cumulative Sum (Forward)	0.048716	1
Cumulative Sum (Backward)	0.437274	1
Runs	0.090936	1
Longest Run	0.000648	1
Rank	0.437274	0.9230769
FFT	0.162606	0.9230769
Non-Overlapping Template	0.301262	0.9885655
Overlapping Template	0.964295	1
Universal	0.437274	1
Approximate Entropy	0.162606	1
Random Excursions	0.42722271	1
Random Excursions Variant	0.59290552	1
Serial 1	0.834308	0.9230769
Serial 2	0.275709	0.84615384
Linear Complexity	0.437274	1

**Table 5. 15 MICKEY 2.0.85 NIST test results for ciphertext with 13 MB bits length**

Tests	P-value average	Passing rate
Frequency	0.437274	1
Block Frequency	0.637119	1
Cumulative Sum (Forward)	0.090936	1
Cumulative Sum (Backward)	0.437274	1
Runs	0.437274	1
Longest Run	0.275709	1
Rank	0.437274	1
FFT	0.437274	1
Non-Overlapping Template	0.331186	0.989604989
Overlapping Template	0.275709	1
Universal	0.048716	1
Approximate Entropy	0.048716	1
Random Excursions	0.61271597	1
Random Excursions Variant	0.90774056	1
Serial 1	0.090936	1
Serial 2	0.090936	1
Linear Complexity	0.275709	1

## 5.8 MICKEY 2.0.85 performance tests

As one of the performance measurements is the speed of generating the keystream, multiple speed tests measured in microseconds were conducted to record the encryption time for both MICKEY 2.0 and MICKEY 2.0.85. The hardware used was an Intel i3 processor which has a speed of 2.53 Ghz. A file of 39,900 bytes of plaintext for encryption with random 10 bytes of key and 10 bytes of IV was used. Tests were run 10 times, each encryption time measured, and the average calculated to determine the accurate speed. The test process was:

1. Record the starting time S.
2. Use the 39,900 bytes plaintext as input.
3. Record the end time E.
4. Calculate E-S.

Table 5.16 shows MICKEY 2.0.85 was 23.36% faster than MICKEY 2.0.

**Table 5. 16 Improvement in the encryption speed for MICKEY 2.0.85 compared to MICKEY 2.0**

Bytes encrypted	Elapsed time in microseconds	
	MICKEY 2.0	MICKEY 2.0.85
39,900	719,041	496,029
39,900	640,037	514,030
39,900	670,038	499,028
39,900	670,038	528,030
39,900	648,037	485,028
39,900	646,037	512,030
39,900	661,037	510,029
39,900	649,037	520,030
Average	662,912.75	508,029.25
<b>Improvement percentage over MICKEY 2.0</b>	<b>0</b>	<b>23.364%</b>

MICKEY 2.0.85, which has fewer GEs, has faster encryption speed which will reduce power consumption, and be suitable for smaller devices such as microcontrollers and RFID readers and tags. Even small improvements in speed will considerably improve performance in constrained devices and applications. This research direction is important for further and continuous improvements as digital devices with smaller size and memory capacity will be used more frequently.

## 5.9 Power consumption testing

To evaluate the power consumption of MICKEY 2.0.85, Xilinx Power Estimator (XPE) [177] was used. It is compared to MICKEY 2.0 and Micro-Trivium [189] which is a lighter version of Trivium. Table 5.17 shows MICKEY 2.0.85 has the lowest power consumption of the four ciphers. The values for Trivium and Micro-Trivium were obtained from [189].

**Table 5. 17 Power consumption for MICKEY 2.0.85 and other ciphers**

Cipher	Number of GEs	Power consumption (microAmps @ 100KHz)
MICKEY 2.0	3,131	0.574
MICKEY 2.0.85	2,741	0.481
Trivium	3,091	0.681
Micro-Trivium	2,696	0.517

The relationship between the number of GEs and the power consumption for all four ciphers in Table 5.17 is explained by the polynomial of degree 3 as given below:

Let G= number of GEs, and P= Power Consumption

The relationship between G and P for all ciphers in Table 5.17 is modelled using 3rd degree polynomial as following:

$$P=-0.0000000271175791 * G^3 + 0.000234730685 * G^2 - 0.675800572 * G + 647.741649$$

For MICKEY 2.0.85 and MICKEY 2.0, the relationship between the number of GEs and the power consumption can be represented as a linear model as given below:

$$P=0.00023846 * G - 0.17262308$$

For MICKEY 2.0 and MICKEY 2.0.85,  $R^2 = 1$  for both models which indicates clearly that the models are accurate, as there is a high correlation and all observed values can be represented by these models.  $R^2=1$ -errors, here errors= 0, which indicates the high determination of the model coefficients.

## 5.10 Cryptanalysis

The NIST tests provide a standard evaluation of the cipher pseudo-randomness as standard efficiency requirements. However, including more attack analysis for MICKEY 2.0.85 compared to MICKEY 2.0 adds another level of proposed cipher usage feasibility, as it ensures more security. Analysis of two types of attacks is presented: many time pad attack and cosine similarity attack.

### 5.10.1 Many Time Pad Attack

A repeated key attack, which may be called a many time pad attack, is a type of attack when the same key is reused. Assume we have the same key ( $K$ ) used to encrypt message  $m_1$  and  $m_2$  and the attacker has access to the ciphertexts for the two messages  $C_1$  and  $C_2$ .

The XOR functions as the following:

$$m_1 \oplus K = C_1 \quad (5.1)$$

$$m_2 \oplus K = C_2 \quad (5.2)$$

Then from 5.1 and 5.2

$$C_1 \oplus C_2 = m_1 \oplus K \oplus m_1 \oplus K \quad (5.3)$$

$$\xrightarrow{\text{yields}} m_1 \oplus m_2$$

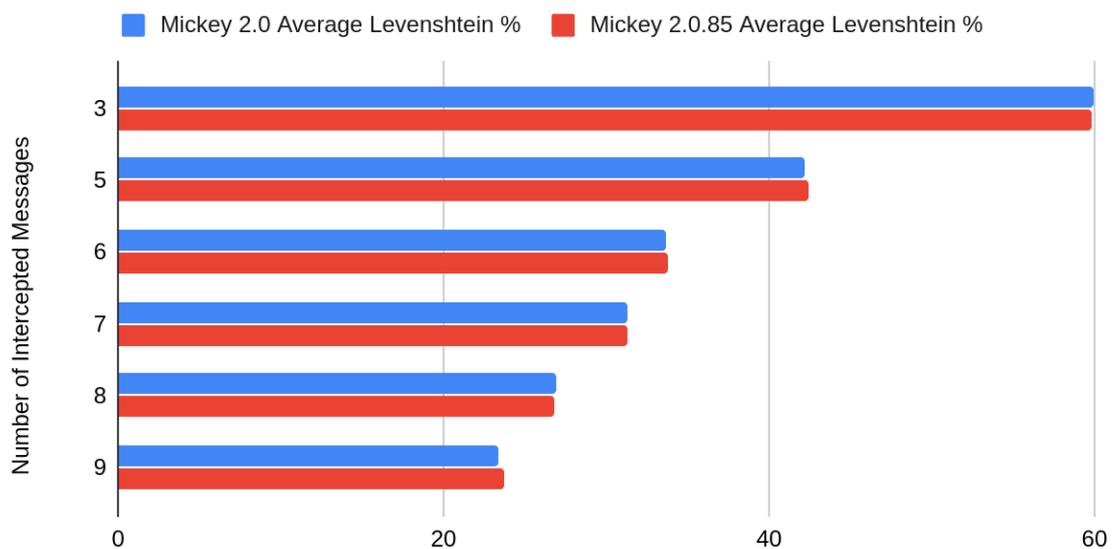
It is not secure to reuse the same key with different messages, as some of the solution is mixed with IV. However, from the previous example, multiple messages were encrypted with the same key, then if the attacker gains access to a sufficient number of the messages then the attacker can extract the plaintext from the given ciphertexts given that the same key was used.

To simulate this attack the IV with the same key with different messages was used, by using multiple plaintext files with length ranging from 56 KB to 96 KB encrypted by both MICKEY 2.0.85 and MICKEY 2.0. Python code was used to perform multiple XORing operations of ciphertext with keystream for each message, then the Levenshtein Distance [175] calculated to find the similarity.

## Calculating the Levenshtein Distance

Let LD be the percentage of the Levenshtein Distance. If the LD between the calculating message  $m_p$  and the original message  $m_o$  by the attacker is  $LD = 0\%$ , then  $m_p$  and  $m_o$  are completely different. On the other hand, if LD between  $m_p$  and  $m_o$  is  $100\%$ , the  $m_p$  and  $m_o$  are completely similar. Figure 5.6 shows the Levenshtein Distance between multiple  $m_p$  and  $m_o$  messages with length mentioned earlier.

Average Mickey 2.0 Levenshtein Percentage vs Average Mickey 2.0.85 Levenshtein Percentage



**Figure 5. 6 Levenshtein similarity test for MICKEY 2.0 and MICKEY 2.0.85 interrupted messages**

Figure 5.6 shows that MICKEY 2.0.85 is as random as MICKEY 2.0 by using the same number of intercepts for the same texts for both ciphers, showing they are very similar, with MICKEY 2.0.85 slightly better in resistance against the many reused key attacks.

### 5.10.2 Cosine similarity attack (cryptanalysis)

Cosine similarity [190] can be an effective tool to determine how similar two documents or texts are. Assume we have two texts represented as non-zero vectors  $v_1$  and  $v_2$  and their lengths are  $\|v_1\|$  and  $\|v_2\|$ , and their dot products are  $v_1 \cdot v_2$ .

Then:

$$\text{Cos}(v_1, v_2) = (v_1 \cdot v_2) / (\|v_1\| \|v_2\|)$$

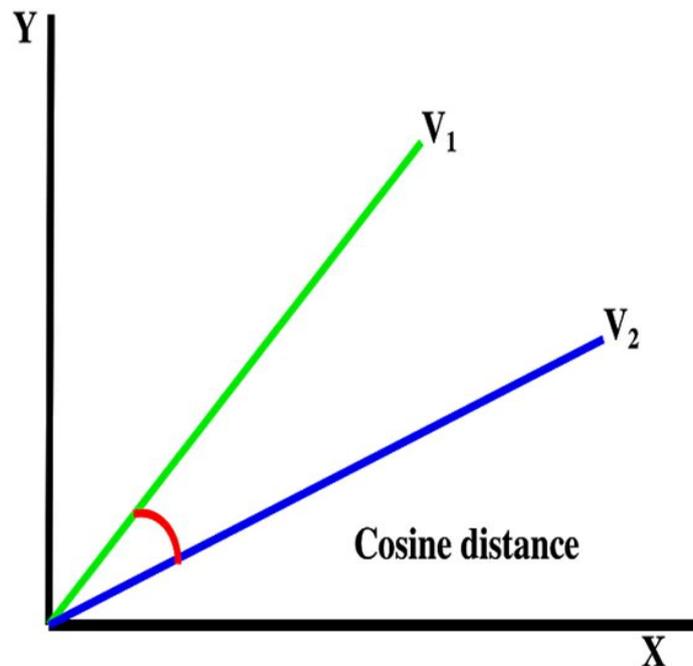
The similarity is  $\cos(\theta)$ ,  $\theta$  is the angle between  $v_1$  and  $v_2$

The similarity between the two texts =  $\cos(\theta) = (v_1 \cdot v_2) / \|v_1\| \|v_2\|$

Comparing two texts is based on the value of  $\theta$ . If  $\theta$  has small value, the texts are more similar and vice versa. If  $\theta$  is close to zero, it implies the texts are very similar. Figure 5.7 shows the cosine similarity between the two vectors  $v_1$  and  $v_2$ .

The plaintext and ciphertext pair are used as numerical bits vectors, by calculating the cosine similarity of the two vectors, then finding the mean of the cosine similarity as the following:

Cosine similarity = 1 – Cosine distance



**Figure 5. 7 Cosine similarity between two vectors  $V_1$  and  $V_2$**

Table 5.18 compares the cosine similarity results of MICKEY 2.0.85 and MICKEY 2.0 for multiple plaintext and ciphertext pairs. The results were obtained by using 132 sequences of length  $10^6$  bits for MICKEY 2.0.85 and MICKEY 2.0. MICKEY 2.0.85 has less similarity between the plaintext and ciphertext. For both MICKEY 2.0.85 and MICKEY 2.0 the Mean Cosine Similarity are far from zero, thus they are both immune from this kind of attack.

**Table 5. 18 MICKEY 2.0.85 and MICKEY 2.0 results by applying cosine similarity**

Methods	Mean Cosine Similarity	STD Cosine Similarity
MICKEY 2.0	0.8472	0.0225
MICKEY 2.0.85	0.8418	0.0298

### 5.11 Discussion of results and analysis

The proposed MICKEY 2.0 variant called MICKEY 2.0.85 is suitable for RFID tag usage as the tags have limited size and power consumption. MICKEY 2.0.85 is also suitable for small devices and microcontrollers. In tests for encryption and read required time (microseconds as seen in Table 5.16) by an Intel i3 processor with speed 2.53 Ghz, MICKEY 2.0.85 was 23.36% faster. The result is a good security improvement for RFID technology and IoT technology in general.

The scaled down MICKEY 2.0.85 version is 23.36% faster than the original MICKEY 2.0 and uses 16.202% less power than MICKEY 2.0, which was achieved by reducing the number of GEs by 12.45% (as in Table 5.17). The suite of NIST randomness tests confirmed that MICKEY 2.0.85 was slightly more random than MICKEY 2.0, hence MICKEY 2.0.85 is resistant against attacks, especially statistical attacks which target the keystream, internal state and ciphertexts.

Further reduction may be achieved, however it is important to ensure it does not compromise the security, and NIST randomness tests can help to evaluate that. The reduced version was based on many experiments and statistical evaluations to derive the optimal possible version. In MICKEY 2.0.85 the internal state was reduced by 30 bits to 170 bits, as the MICKEY 2.0 internal state has 200 bits. Further reduction should be limited within the range of 160 bits to 170 bits, as the general security rule is the internal state should remain at least twice the key length.

The methodology in this chapter for introducing a MICKEY 2.0 based version can inspire further improvement in lightweight cryptography. For example, the same methodology can be implemented to create lighter variants for other lightweight stream ciphers. Designing new lightweight stream ciphers is important with increasing use of IoT technology. Nevertheless, optimising and improving the current lightweight stream ciphers, which have a reasonable level of security, is still crucial.

The importance of statistical randomness tests is emphasised in the current literature, as well as in this study. Chapters 3, 4 and 5 show the use and value of statistical testing and modelling, including NIST tests, to investigate and provide detailed explanation and results. The tests showed how MICKEY 2.0.85 is resistant against two statistical based attacks.

Another possible area worth investigating is the reduction in specific register. For example, for S the nonlinear register, a future research direction is to work around the internal state of length within the range of 160 bits to 170 bits, and make S length > R length and apply the NIST tests to measure the level of randomness. It may also be worth modifying taps positions in R-Masks, COMP0, COMP1, FBO and FB1 to determine the best version which satisfies desired randomness requirements, power consumption, fewer GEs and keystream generation speed, which has already been achieved in MICKEY 2.0.85.

## **5.12 Conclusion**

This chapter has introduced a lighter, secure and faster version of MICKEY 2.0, called MICKEY 2.0.85, which uses less power, making it more suitable for IoT applications. All tests and performance measurements were applied to ensure MICKEY 2.0.85 is an optimal version that has a good trade-off between security and suitable features for IoT implementation. The proposed MICKEY 2.0.85 cipher is an important contribution to the current literature of lightweight encryption methods enhancement.

The proposed version MICKEY 2.0.85 is compared to existing lighter versions of other stream ciphers such as Trivium as there are many reduced and optimised versions, including the recent Micro-Trivium.

The following chapter implements MICKEY 2.0 in an IoT field of mobile cloud computing. It presents a secure protocol for mobile device communication over an insecure communication channel.

# **Chapter 6: Mobile cloud computing and FEATHER, a proposed lightweight security protocol**

## **6.0 Chapter overview**

This chapter develops a lightweight security protocol called FEATHER for use in securing mobile cloud computing communication. The outline the sections of the chapter are: section 6.1 the essential introduction, section 6.2 provides a brief background in cloud computing, section 6.3 introduces the proposed lightweight protocol FEATHER and its design principles, section 6.4 about the protocol implementation, section 6.5 provides the performance results with analysis, section 6.6 show how FEATHER is resistance against possible attacks, section 6.7 for the overall discussion, section 6.8 concludes the chapter.

## **6.1 Introduction**

Cloud computing, and the related mobile cloud computing, are large and growing fields. The continuous and exponential [191] growth of mobile devices in quantity and quality means mobile cloud computing is gaining more attention as it serves important applications such as mobile learning, mobile commerce, mobile gaming, eHealth applications and web searching [9]. Furthermore, the limitations of personal computers and devices' computation abilities generates the need for more powerful computation resources. Cloud computing facilities provide more powerful and affordable personal usage. As cloud computing providers are continually improving their services, new providers are likely to join this growing market which will create competition, which is beneficial for the service users, thus an essential requirement is the security of client data.

Mobile cloud computing is an active and important research area as it supports people using the internet for communications, at any time and at any place, by using the storage and computer activities which can be done by adapting mobile cloud computing [192]. Mobile phones have become very widespread and are a major part of daily lives. According to “Statatista” based on more than 22,500 sources, the number of mobile devices in 2015 was 2.15 billion, 4.3 billion in 2016 and 4.57 billion in 2018, with a prediction of 4.78 billion devices in 2020 [193]. With many devices connected to each other via big networks, there is a threat of attacks which requires the use of good security protocols. Because users

need to communicate confidentially, especially sensitive information transmitted through insecure communication channels such as Wi-Fi, 3G and 4G, encryption systems suitable for these devices are needed. It is important to design a protocol to ensure confidentiality and effectiveness, considering the following requirements:

1. Speed
2. Ensuring identity
3. Ensuring confidentiality
4. Compatibility with mobile devices
5. Effective communication between cloud server and mobile devices through a communication channel which is not safe.

Companies such as Google, Microsoft and Amazon provide platforms for cloud computing users. For example, Google provides services such as Google Maps which can be accessed from mobile phones for navigation and location searching and sharing [194]. Amazon provides services such as EC2 for computing and S3 for storage and they introduced services based on users' demand to customise their service according to the users' requirements [195]. Microsoft provides their version of cloud computing called Azura. Cloud computing related security concerns such as confidentiality, integrity and storage are important challenges [196].

In this age of advanced communications technologies and growth in using communications devices, the demand for privacy and security of systems to ensure the transfer of confidential information is also growing. Many cryptosystems meet this demand, however some of them need a large computation capability. Advanced Encryption System (AES) is widely used as it is considered a very strong and secure cryptosystem [197]. However, it is a "heavy" system which requires large computation resources and power consumption, and is not suitable for small devices with limited computation capacity such as mobile devices.

As AES is the first choice to be used, and due to the need to use a similar cryptosystem in small devices, some researchers have introduced lightweight versions such as ALE [198] as AES needs more resources such as more central processing units (CPUs) and memory to generate the keystream. Some components in cloud computing such as embedded systems on cloud computing with 32-bit, 16-bit and 8-bit microcontrollers usually struggle

to keep up with real time demands for conventional methods of cryptography [199], therefore AES is not a good solution for many embedded devices in cloud computing that have a small computation ability.

### **6.1.1 Cloud computing**

There is a current need to adopt a lighter encryption method in cloud computing, and lightweight stream ciphers can be implemented to provide the required security. Lightweight stream ciphers are schemes of encryption that include a decryption function as well as an encryption function with the capability of handling messages of arbitrary length. Thus, they are better than block ciphers such as AES that only handle inputs of a fixed length (flexibility is important). Due to their functionalities, they are well adapted to low bandwidth or noisy communications and thus are a good solution particularly in cloud computing. Speed, memory, number of CPUs and cost efficiency are important factors [200]. Chapter 5 proposed a MICKEY 2.0 variant, MICKEY 2.0.85, as the preferred choice rather than other lightweight stream ciphers as it needs less size and has less energy consumption, which in turn is cost efficient [201]. However, the protocol can be adapted to implement other lightweight ciphers such as Trivium or Grain.

Lightweight stream ciphers are an important security tool for IoT applications, such as RFID tags, which have a very constrained environment which can adopt small and lightweight cryptosystems [202].

Now mobile device users can use the mobile device terminal to send files to a server to get a cryptographically secure keystream, and also use mobile communication with other mobile devices through Wi-Fi, 3G, 4G and the imminent 5G. This circle of communications between mobiles and servers also needs to be achieved in a secure manner. A secure protocol is needed which can be implemented in mobile devices for secure communication between mobile devices and between mobile devices and a server.

## **6.2 Background for mobile cloud computing**

Data transfer between two mobile devices and transfer from a mobile device to the cloud needs to be done securely, as there are multiple communications through different channels, such as the Wi-Fi network, 4G and 5G. Because those channels do not provide

the necessary security, there is a need to establish a secure protocol for data transfer through this unsecure communication method. The privacy and integrity of files and data must be guaranteed and maintained in all aspects of communications.

As mobile devices have limited computation power, it is hard to address all security cryptosystem tasks. Bahl et al. [203] published a short study based on cloud computing and mobile computing and debated the importance of leaving the offloading tasks to be done in an external application which can be carried out by an external server. They proposed a mobile cloud computing enterprise that consists of four elements: mobile devices, wireless core, Wi-Fi access point, and regional information centres (RDC).

In addition to the limited computational capabilities in mobile devices, battery consumption due to heavy computation adds another challenge which makes mobile cloud computing a good solution. Kumar et al. [204] showed that mobile computing could save energy by offloading some tasks to the cloud server such as battery life and wireless energy which is used for transferring the data in some applications, however some applications are not energy efficient.

Bahrami et al. [205] studied the adequacy of using AES in mobile cloud computing. and explained the cost and how cryptosystems such as AES are beyond the ability of mobile devices to handle. By studying the current methods provided they showed that, in the case of mobile cloud computing, when considering that mobile devices have limited resources, such as limited power energy, low speed processors and tiny RAM capacity, it is not a good approach to use AES as the encryption technique for each file once offload/download is done for every single transferred file. Therefore, they introduced a lightweight method such as pseudo-random permutation based on chaos systems [205]. Another solution for this challenge is lightweight security methods that provide a balance between maintaining energy efficiency and security. A lightweight security technique can be considered an easy operation, in this regard permutation, instead of using complicated and expensive operations when using secret key or public-key encryptions [206-208].

### **6.2.1 The advantage of using stream ciphers in small devices**

A stream cipher is a symmetric cryptosystem which uses the same key for encryption and decryption. Stream ciphers can transform data faster than other ciphers such as block

ciphers which is another branch of a symmetric cryptosystem, and also faster than ciphers in an asymmetric cryptosystem [16], [209].

Stream ciphers are less secure than other symmetric and block cipher types of cryptosystems such as AES which is known as one of the most secure ciphers. The encryption process in AES involves permutations and a substitution process and a number of rounds which need more power and storage space. On the other hand, lightweight stream ciphers such as MICKEY 2.0, Trivium and Grain [210] need much less power and memory which is attractive for small applications and devices. When looking for suitable ciphers for small applications, widely used lightweight stream ciphers include E0 used in Bluetooth, RC4 in Web, and A5 family in GSM [16]. In addition, in a small device, such as a RFID tag, mobile and microcontroller, the large throughput generated by the cipher needs to be offloaded to the cloud even for more recent mobile devices due to the need to encrypt a large file quickly [211]. Furthermore, it can be used for noisy channels and cases with low bandwidth [212], making them the optimal choice for mobile cloud computing. Stream ciphers have advantages due to their high throughput property and low computational complexity. Lightweight stream ciphers [213] are a better choice than block ciphers as they need less memory and less hardware complexity.

### **6.2.3 Using lightweight stream ciphers in cloud computing and mobile cloud computing**

Lightweight stream ciphers have several advantages for cloud computing. They provide fast encryption by generating the secure keystream faster than other popular ciphers such as AES. They need fewer computation facilities such as CPUs and memory required in the cloud which reduces the cost and the power consumption significantly. Microcontrollers in the server with 8-bits and 16-bits make it hard to achieve the heavy computation power using cryptosystems such as AES.

In addition to the advantages of using lightweight encryption in cloud computing, additional benefits in mobile cloud computing include helping more mobile devices to communicate as the encryption is fast, consumes less battery power, and needs less bandwidth.

### **6.2.5 AES and CLOAK protocol**

CLOAK is a lightweight protocol based on the AES cipher which enables two mobile devices to communicate with each other, while leaving the keystream generation on an external server (AWS in their implementation) [214]. As CLOAK can get the keystream from either trusted or untrusted external servers, the main security concern in implementing the protocol is to protect the keystream. Security can be compromised by fetching the keystream from an external server and from communication media.

Lightweight stream ciphers which can be used in mobiles include Trivium [116], Grain [117] and MICKEY 2.0 [215]. The advantage of using MICKEY 2.0 cipher over the others is that it is more resistant to statistical attacks [216], with no successful algebraic attack so far [217], [218]. It can also produce large throughput. The lightweight protocol developed here does not rely on the server to be secure and will not be compromised as in the CLOAK protocol which assumed the security of the server relies on the server provider [214]. Using MICKEY 2.0 in this lightweight protocol to provide a secure keystream is significantly faster than using AES for example, reducing the time needed by the server to generate the keystream which in turn reduces the time to transfer the data between the server and the mobile. If multiple mobile devices need to connect to the server at the same time, that will significantly reduce the overall time to transfer information between the mobiles and the server. The external server is usually used for complex operations that require large processing capacity and large computational capabilities. In this case, the lightweight protocol needs to offload producing a secure keystream task.

It may use two external servers: one is to produce a keystream, and the second is to save the keystream to avoid a breach of the server. This adds another security dimension for sensitive information.

### **6.2.6 Motivation and challenges**

To meet the security challenges, as well as the demand for a lighter security protocol to save time and address computation power, device hardware limitations and battery consumption, the research questions to answer are:

1. How can MICKEY 2.0 be implemented efficiently to secure communication between mobile devices in mobile cloud computing?

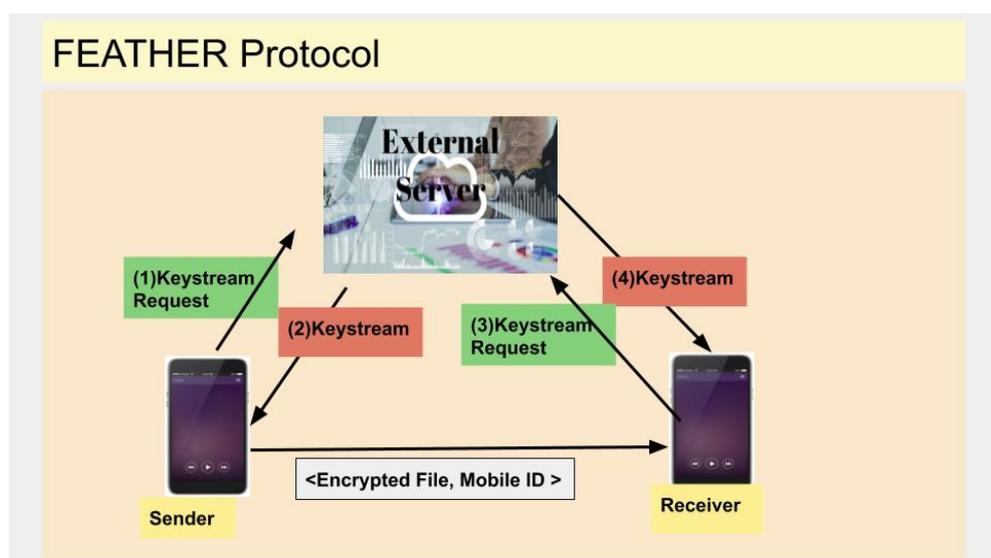
2. How can the performance of a new security protocol be evaluated against the existing protocol?
3. How can a clear justification be provided that the new proposed protocol is immune from possible attacks?

The aims are to:

- Implement MICKEY 2.0 efficiently to secure communication between mobile devices in mobile cloud computing.
- Evaluate the performance of the new security protocol against the existing protocol.
- Provide a clear justification that the new proposed protocol is immune from possible attacks.

### 6.3 The lightweight protocol FEATHER

This thesis research designed a MICKEY 2.0 cipher based protocol, called FEATHER, to strengthen confidentiality and protection during messaging between mobile devices as well as communication between devices and the cloud server, see Figure 6.1. The MICKEY 2.0 cipher produced a secure keystream in the external server to reduce reliance on mobile devices which have limited computing power and memory. The role of mobile devices is only encryption and decryption which provides mobile devices with the ability to compute and reduce energy consumed by the device battery.



**Figure 6. 1 Communications between mobiles and external server – FEATHER protocol**

A lightweight secure protocol is introduced to communicate between devices and the external server over the cloud, as well as design applications on mobile devices for the process of verification and encryption and decryption. The proposed protocol is faster and moves larger files compared to the CLOAK cipher [214]. The protocol also maintains a high level of security. A protocol was designed to achieve security through the application of the MICKEY 2.0 cipher with additional protection systems for identity verification such as hash functions, time stamp, and out-of-band password.

A lightweight stream cipher is needed for important reasons such as generating the keystream faster, and using fewer resources, such as CPUs and memory, so more secure applications can be created to take advantage of advances in mobile cloud computing. If the keystream generated in the server is faster, it will allow more mobiles to get it from the cloud compared to a heavy encryption system like AES. Thus it will be more efficient and will greatly reduce costs. For example, using MICKEY 2.0 meets most of these needs.

### **6.3.1 FEATHER protocol design principles**

The concept is that person A wants to share some secret information with person B. Person A might be using a mobile device and may want to share chat messages, image files, etc. Alternatively, Person A may have several simple IoT devices (microcontrollers) and want them to report back sensor data. Person A considers the information private or sensitive and wants to prevent a third part from intercepting the data.

There are nine design principles for a lightweight protocol.

#### **Principle 1 Avoid implementing a heavy encryption method**

As some popular encryption algorithms, such as AES, require considerable resources in terms of CPU time and/or memory usage, the aim was to design a protocol that offloaded the more computing-intensive steps to a server in the cloud, while simplifying the steps carried out on the mobile device. Therefore, a lightweight protocol, from the viewpoint of the mobile device, can offload generation and storage of the keystream to a server using the MICKEY 2.0 algorithm.

### **Principle 2 Avoid relying entirely on the server**

It is important to avoid relying entirely on the server to ensure the security of the communication. Even if an adversary compromises the server, they cannot easily use the captured keystream data to decrypt messages directly. Therefore, the following is considered. Although the client receives a keystream from the server, the client does not use it directly. Instead, the client selects a few random values using primitive polynomials to apply the keystream to the plaintext to compute the encrypted data.

### **Principle 3 Send messages between the client and server over the internet**

The protocol must assume an adversary may intercept messages, or an impostor may try to insert invalid messages in the client–server communication. One popular approach would be to use a key-exchange algorithm, such as Diffie–Hellman (which is vulnerable to a man-in-the-middle attack), or a more sophisticated Station-to-Station protocol [219], which avoids this vulnerability. Both of these approaches require significant computation that may not be appropriate for simple mobile or microcontroller devices. This protocol needs to assume the ability to send brief out-of-band messages using a different communication medium. For example, if the protocol is implemented on top of the HTTP protocol, a secret out-of-band message may be sent by email or SMS. In this protocol, an out-of-band message is sent from the server to the client to convey a one-time-pad, and from one client to another client to convey a file token and secret values (using primitive polynomials) used to step through the keystream.

### **Principle 4 Focus authentication on unique security parameters**

For authentication, this protocol uses a “bring something, know something” technique. The protocol assumes each mobile device (or microcontroller device) has a universally unique identifier (UUID). It also allows each user to select a username that is not necessarily unique. These are combined using a hash function to generate a unique identifier (UID) for each user. At the initiation of the protocol, each user registers their UID and then communicates an encrypted copy of their secret password to the server.

For subsequent communication, all messages between the client and server are validated using a digital signature based on hashing the message and the secret password. In this case the “bring something” refers to the device and its UUID and the “know something” refers to the user’s secret password. Since an adversary does not know the secret password,

it cannot generate a valid signature, so the client and server can reject messages with invalid signatures.

#### **Principle 5 Secure the communication between the client and the cloud server**

To secure the communication between client and server, they rely on a shared keystream. This shared keystream is first generated by the server when the client sends a message to register the user. In its response to the client, the server sends the shared keystream, encrypted with the one-time-pad, to prevent an adversary from capturing the keystream.

#### **Principle 5 Offload the keystream generation to the cloud server**

The server implementation may use any reasonable technique for keystream generation. In practice, a method is needed that is computationally efficient and still provides a reasonable level of security. To generate a new keystream for each user, the server must first create an initial key (or key+IV pair).

#### **Principle 6 Ensure client request for the keystream from the cloud has time authentications**

When the client submits a request to generate a new keystream, it includes a token and expiry time. There are two possible implementations. The server may simply generate and store a key, and then generate the actual keystream “on the fly” whenever it is requested. Alternatively, the server may generate the keystream right away and store it as a file, to be retrieved later when the client submits the corresponding token. The expiry time allows the client to limit the time the keystream is stored on the server. This reduces the availability of the keystream if an adversary tries to compromise the server.

#### **Principle 7 Ensure there are possible and flexible variations for secure data transfer**

To enhance the security of the protocol, the server never has access to the unencrypted data. The data is encrypted by the client, using a modified version of the keystream, and the modification is unknown to the server. When transferring encrypted data from one client to another, there are three main options available.

1. In one variation, since the data is securely encrypted, the file can be uploaded to any simple file server. This may provide an increased level of security, since it introduces a separation from the keystream server and the file server. In fact, clients

would be free to use a variety of different file servers to transfer encrypted data files, as long as these are communicated between the sender and receiver.

2. In a simpler implementation, the clients can upload or download the encrypted data to the server, and are identified by a unique token. This token can be generated pseudo-randomly to make it difficult to guess. Any other client can download the encrypted file, asynchronously, once it receives the appropriate token from the first client. Some efficiency can be gained if the file upload/download is implemented on the keystream server, since the same protocol mechanism could be used to download a keystream (given a token), or to download encrypted data (given a token). In fact, once a keystream is generated and stored as a file, the keys used to generate the keystream could be deleted, reducing the vulnerability of the protocol.

3. In the third option, the encrypted data could also be transferred directly and synchronously from one client to another. This approach could make sense when a pair of clients wish to send and receive a number of smaller messages, as in a secure chat session. This can be accomplished first by generating and downloading a keystream, and then sending encrypted messages back and forth, without requiring an intermediate file server.

#### **Principle 8 Modify the keystream to further enhance the security**

For efficiency, the client uses a keystream generated by the remote server, but for security, the keystream is modified in a way unknown to the server. In particular, the client randomly selects a small number of parameters that describe a particular pseudo-random permutation of keystream values. By sharing these secret permutation parameters with the other client through an out-of-band communication, the other client will be able to decrypt the encrypted file.

#### **Principle 9 Ensure data in the cloud server is tied to expiry time**

The security of the protocol is enhanced by reducing how long information is retained before being deleted. Both the keystream and encrypted files have an associated expiry time, after which the server deletes them. This reduces the information that is exposed if the server is ever compromised.

## 6.4 Protocol implementation

This communication protocol enables mobile devices with limited computational resources to share encrypted files with the help of an external server with greater computing, storage and bandwidth resources.

The protocol uses two communication channels. The first channel is assumed to be insecure, such as the internet, using HTTP to transport messages between the mobile devices and the external server. The second channel carrying “out-of-band” messages is assumed to be secure and could be implemented using SMS messages to mobile devices, with possible alternatives of email.

The first channel allows mobile devices to initiate six actions, by sending a message to the external server and receiving a response.

The second out-of-band channel is used to send and receive three kinds of secret information:

- a one-time-pad, which could use a more secure parameter instead with justification
- a file id
- a token id (and some additional parameters).

The protocol also uses a cryptographic hash function, such as SHA-256 which outputs a 32 byte hash value. For distinct pairs of strings  $s$ ,  $t$ , we have  $H(s) \neq H(t)$  (with very high probability).

Messages in the protocol are simply concatenated key=value pairs of parameters.

Each of the 11 possible parameters is identified by a unique character:

- a = action
- s = status
- c = code (error code)
- u = uid
- p = phone
- f = token or file
- d = data
- n = number

e = expire  
t = timestamp  
x = signature

The timestamp is Unix time in seconds, and can help prevent “replay attacks”. The cryptographic signature is a hash of the entire message string (before the signature is added) and is used to authenticate messages.

The six actions and messages are: REGISTER, UPDATE, VALIDATE, GENERATE, UPLOAD, and REQUEST.

### 1. REGISTER

The person using the mobile device app provides a username (eg, “Jason”). The device hardware is also assumed to have a unique hardware identifier (eg, DeviceID). The mobile app combines these strings using a hash function to get a unique id that can be sent to the external server, without revealing any private information.

$uid = H(\text{device-identifier}, \text{username}); 32\text{-byte value}$

The mobile device also has a telephone number at which it can receive an out-of-band message via SMS.

The person registers an account on the external server by sending a message:

a = register  
u = uid  
p = phone  
t = timestamp

When the external server receives this message, if no account exists for that uid, a new account is created, and this message is sent back:

s = OK  
t = timestamp

If an account already exists for that uid, the server responds:

$s = \text{ERROR}$

$c = \text{code (indicating type or error)}$

$t = \text{timestamp}$

If an account already exists for the given uid, the person needs to pick a new username, to create a different uid.

## ONE-TIME-PAD via SMS

Following a successful REGISTER message, the external server sends a one-time-pad to the mobile device via an out-of-band channel using SMS to the phone number provided. The person would need to cut-and-paste this string into the mobile device app to be stored.

## 2. UPDATE

In the mobile device app, the person also provides a password (eg, “MySecret”) which provides a type of “bring something, know something” security.

(bring something = mobile device; know something = username, password)

The user’s simple password is combined with the uid to create a “hashed password”, which will be sent to the external server.

$\text{pass} = H(\text{uid}, \text{password})$

The hashed password is encrypted using XOR with the secret one-time-pad. The entire message (before the signature) is hashed to create a cryptographic signature for authentication.

$a = \text{update}$

$u = \text{uid}$

$d = \text{XOR}(\text{pass}, \text{OTP})$

$t = \text{time}$

$x = H(\text{message})$

The external server confirms the validity of the message by recomputing the signature, and then decrypts and stores the hashed password in the account. The response is either OK, or ERROR.

### 3. VALIDATE

This message is optional, but useful for debugging purposes when implementing this protocol for the first time. The mobile device sends the following message asking the external server to confirm the hashed password and signature are valid.

a = validate  
u = uid  
d = XOR(pass, OTP)  
t = time  
x = H(message, pass)

The external server decodes the hashed password, recomputes the signature, and responds with OK or ERROR.

### 4. GENERATE

The mobile device provides a unique 32-byte token and asks the external server to generate a new encryption key that will be used to generate a keystream of “number” bytes that will be stored until a given “expire” time. The unique token is created by hashing the uid, expire and timestamp.

token = H(uid,expire,timestamp)

The token is XOR-encrypted with the shared-keystream. The message sent to the server has these parameters:

a = generate  
u = uid  
f = XOR(token,shared-keystream)  
n = number (of bytes in the keystream)  
e = expire  
t = timestamp

$$x = H(\text{message}, \text{pass})$$

The external server generates a random MICKEY 2.0 key (20 bytes of key+IV). There are two implementation-dependent choices:

- The server can simply store the 20-byte in association with the token and generate the keystream on-the-fly when requested, or
- The server can generate and store the keystream, and then discard the 20 byte key. With this option, the token becomes equivalent to a file-id and the keystream becomes equivalent to the file contents.

## 5. UPLOAD

The mobile device asks the external server to store a file by providing a 32-byte file-id, the encrypted contents of the file, and an expiration time, after which the file will be deleted. The unique file-id is created by hashing the uid, filename, expire and timestamp.

$$\text{file} = H(\text{uid}, \text{filename}, \text{expire}, \text{timestamp})$$

The file-id is XOR-encrypted with the shared-keystream. The mobile device sends a message with these parameters:

$$\begin{aligned} a &= \text{upload} \\ u &= \text{uid} \\ f &= \text{XOR}(\text{file}, \text{shared-keystream}) \\ d &= \text{XOR}(\text{file-contents}, \text{token-keystream}) \end{aligned}$$

The external server stores the file and response with OK, or else ERROR if something went wrong.

## 6. REQUEST

A mobile device can request a token-keystream or encrypted file contents by providing the appropriate 32-byte token or file-id. The message has these parameters:

$$\begin{aligned} a &= \text{request} \\ u &= \text{uid} \\ f &= \text{XOR}(\text{token}, \text{shared-keystream}) \end{aligned}$$

or  $f = \text{XOR}(\text{file}, \text{shared-keystream})$   
 $t = \text{timestamp}$   
 $x = \text{H}(\text{message}, \text{pass})$

The external server uses the token (or file-id) to look up the requested data and sends it back to the mobile device.

$s = \text{OK}$   
 $d = \text{XOR}(\text{token-keystream}, \text{shared-keystream})$   
or  $d = \text{XOR}(\text{file-contents}, \text{shared-keystream})$   
 $t = \text{timestamp}$   
 $x = \text{H}(\text{message}, \text{pass})$

The protocol assumes the first mobile device (the sender) is able to communicate the “token” and “file” to the second mobile device (the receiver) through a secure out-of-band channel, here assumed to be sending an SMS message.

In addition, it is important the communication remains secure even if the external server is compromised by an adversary. Therefore, the token-keystream is not used directly to encrypt the file contents, since someone with access to the server could easily decrypt the file.

Instead, the first mobile device must pick several random numbers  $R_1, R_2, R_3, \dots$  that are used to walk through the bytes of the token-keystream in a deterministic but difficult to predict order. These sets of random numbers must also be communicated to the second mobile device through a secure out-of-band channel. For example, for a token-keystream with length  $N=2^k-1$ , which is a prime number, the index of the next byte to be used could be calculated:

$$\text{index}(i) = R_1 \bmod N$$
$$\text{index}(i+1) = (R_2 * \text{index}(i) + R_1) \bmod N$$

Appendix 6 provides a sample of previous FEATHER operations.

The mobile app was designed by using Android studio, then the app was transferred as a file to be converted as a mobile local app. The code was written in Java on the Android

studio platform which works on the major operating systems of Windows, MacOS and Linux. Tables 6.2–6.5 show the computations by the app after it was installed in five different Android-based devices. Table 6.1 summarises the devices’ specifications.

## 6.5 Results and analysis

The protocol performance is measured based on two items: the overall speed as presented in the following tables, and battery consumption.

### 6.5.1 FEATHER speed performance

Five different mobile devices with Android-based operating systems, shown in Table 6.1, were used to test the protocol performance. The total time from downloading the keystream, encryption and writing to storage was measured.

**Table 6. 1 Specifications of five mobile devices used to test FEATHER**

	<b>D-1</b>	<b>D-2</b>	<b>D-3</b>	<b>D-4</b>	<b>D-5</b>
Model Name	LG V20	Huawei Nova 3e	Samsung Galaxy S9+	Samsung Galaxy A6+	Lenovo M10 Tablet
OS	Android 7.0 Nougat	Android 8.1 with EMUI 8.0	Android 9.0 P	Android 8.0 Oreo	Android 8.0 Oreo
API level	24	26	28	26	27
CPU	Quad-core 2.15GHz + 1.6GHz	Quad-core 2.36 GHz	Octa-core (4×2.7 GHz & 4×1.7 GHz)	Octa-core 1.8Ghz	Octa-core 1.8GHz
Chipset	Qualcomm Snapdragon 820	HiSilicon Kirin 659	Qualcomm Snapdragon 845	Qualcomm Snapdragon 450	Qualcomm Snapdragon 450
RAM	4GB	4GB	6GB	4GB	3GB
GPU	Adreno 530	Mali-T830 MP2	Adreno 630	Adreno 506	Adreno 506
Battery	3200 mAh, Li-Ion	3000 mAh, Li-Polymer	3500 mAh, Li-Ion	3500 mAh, Li-Ion	4,850 mAh, Li-Ion Polymer

Table 6.2 shows the total time average for the five different devices. The LG V20 device was the slowest at 18.44169 seconds, however it was very fast for 8 MB file size. The Samsung Galaxy S9+ device had the fastest total time average (for Download, Decode and

Write) at 10.34381968 seconds. The total time for all five devices was 71.64568994 seconds and the average was 14.329137988 seconds.

**Table 6. 2 Running 8 MB file 60 times and taking the average time (in seconds) for five different devices**

	<b>D-1</b>	<b>D-2</b>	<b>D-3</b>	<b>D-4</b>	<b>D-5</b>
Device Model Name	LG V20	Huawei Nova 3e	Samsung Galaxy S9+	Samsung Galaxy A6+	Lenovo M10 Tablet
Download	18.08334	11.594383	10.162450	17.28501	13.083983
Decode	0.13299	0.090583	0.08720308	0.1512014	0.1143399
Write	0.22536	0.12371666	0.0941666	0.2327666	0.1841967
Total time	18.44169	11.80868266	10.34381968	17.668978	13.3825196

In the experiments, 15 different file sizes from 1 KB to 16 MB were used to measure the overall performance, as shown in Tables 6.3, 6.4 and 6.5. It is clear that FEATHER can handle large files, and 16 MB is sufficient to transfer documents and photos. These calculations use the Samsung Galaxy S9+, and a 16 MB file only needs about 19.0 seconds for the overall time which includes downloading the encrypted file from the external server, decryption time and storing it to the device (write).

**Table 6. 3 Running 1 KB to 16 KB files and calculating time (in seconds)**

<b>File size</b>	<b>1KB</b>	<b>2KB</b>	<b>4KB</b>	<b>8KB</b>	<b>16KB</b>
Download	0.302	0.317	0.445	0.274	0.283
Decode	0.00321577	0.00405	0.00138742	0.00310880	0.00486269
Write	0.092	0.066	0.067	0.071	0.086
Total time	0.39721577	0.38705	0.51338742	0.3481088	0.3738701

**Table 6. 4 Running 3 KB to 512 KB files and calculating time (in seconds)**

<b>File size</b>	<b>32KB</b>	<b>64KB</b>	<b>128KB</b>	<b>256KB</b>	<b>512KB</b>
Download	0.324	0.38	0.424	0.743	1.001
Decode	0.00102811	0.00158865	0.00231273	0.00852277	0.01105330
Write	0.085	0.066	0.068	0.057	0.052
Total time	0.41002811	0.44758865	0.49431273	0.80852277	1.0640533

**Table 6. 5 Running 1 MB to 16 MB files and calculating time (in seconds)**

File size	1 MB	2 MB	4 MB	8 MB	16 MB
Download	1.684	2.957	5.439	9.625	18.664
Decode	0.03689342	0.02132185	0.03243165	0.08367915	0.15980173
Write	0.057	0.085	0.092	0.106	0.19
Total time	1.77789342	3.06332185	5.56343165	9.81467915	19.0138017

### 6.5.2 Power consumption

An Android-based application GSam Battery Monitor [220] was used to measure how much of the overall battery power FEATHER will consume using a Samsung Galaxy S9+ with a 3500 mAh Li-Ion battery. After running GSam and the mobile app for FEATHER, results found that performing the operations on ten files varying from 2 MB to 16 MB consumed less than 1% of all apps running in the background which consumed 1% of battery power so FEATHER consumes only 0.0001% of battery power.

### 6.5.3 FEATHER vs CLOAK

The proposed FEATHER protocol is lighter than CLOAK, and is also much faster. Comparing the file sizes 1 MB, 2 MB, 4 MB and 8 MB shows that FEATHER is faster. For example, in Table 6.6, total time for file size of 8 MB is 110 seconds for CLOAK, and about 9.8 seconds for FEATHER. Therefore, FEATHER will be even more practical if multiple devices need to communicate at the same time. In addition, FEATHER consumes 80% less battery power than CLOAK.

**Table 6. 6 CLOAK and FEATHER protocols: total speed time for different files sizes**

File size/total time in second	CLOAK	FEATHER
1 MB	20	1.77789342
2 MB	30	3.06332185
4 MB	60	5.56343165
8 MB	110	9.81467915

## **6.6 Attack analysis**

This section provides an analysis of common attacks, and shows how FEATHER is resistant to these types of attacks.

### **Man in the middle attack**

The attacker can interrupt the data, can inject information and can redirect the traffic. This can be between the two devices or between the devices and the external server, so it is working on the communication channel. This can be prevented from occurring by providing strong mutual authentication and end point authentication, as the FEATHER protocol does, and by using hashing for messages, which is met as all messages are wrapped in hash functions. Thus FEATHER is immune from man in the middle attacks.

### **Insider attack**

On the server side, if an insider can gain access to the information, the only thing the insider can get is the keystream. However, the message will be included in a hash function, as well as the one-time-pad, another secure parameter such as timestamp and random number only known by the mobile device users. On the mobile side, the mobile will validate the messages received from the server and other mobiles.

### **Denial of service attack**

The FEATHER protocol has steps in the external server to authenticate users before accessing the service by 1) authentication of users' credentials, 2) updating the accessing parameters, and 3) validating the users' messages and hash functions. As the verification by the server and devices is mutual, a denial of service attack is not applicable.

### **Chosen IV-attack**

The keystream is generated by using MICKEY 2.0 and (key, IV) as the initial input. In FEATHER, the IV is not used more than once with the same key, thus FEATHER eliminates this threat by preventing reusing the IV, as well as by including the IV in the hash function, so an attacker choosing the IV will not result in the key being revealed.

### **Two time pad attack**

Assume there are two messages  $m_1$  and  $m_2$ , if the same key ( $k$ ) is used that is called two time pad, and there are two ciphertexts ( $c_1, c_2$ ) then

$m_1 \oplus k$  that results in  $c_1$  and

$m_2 \oplus k$  that results in  $c_2$

So it is easy for the attacker to perform the XOR operation for ciphertexts in order to reveal the plaintext as:

$c_1 \oplus c_2$  that is, using statistical frequency analysis leads to  $m_1 \oplus m_2$

In the FEATHER protocol, each file was encrypted by using a different keystream as well as a different one-time-pad for every session and time timestamp, thus this attack is not applicable.

### **Impersonation attack**

This kind of attack can happen where the attacker gains access to a mobile and requests a response from the server. The server will validate and authenticate the request. As mobile users will be using a hash function including one-time-pad (as discussed in the protocol implementation), the server also will hash the keystream with one-time-pad among other user credentials, meaning this attack is not feasible with FEATHER.

### **Brute force attack**

As the complexity of a brute force attack in key = 80 bit in general  $2^{80}$ , the FEATHER protocol used a hash function. For example, using D-3 (a user may choose other stronger hash functions, and that will not affect the speed performance as the slower part is the downloading time), the computation power relies on the implementation, and adding other secure parameters such as using OTP, that is similar to the one-time-pad cipher, which substantially raises the computation power needed to break the protocol.

## **6.7 Discussion**

In FEATHER, downloading is the most time consuming task compared to the CLOAK protocol. If it requires more than two mobile devices to communicate at the same time, the external server generating the keystream in the FEATHER protocol is much faster than CLOAK. This will reduce the overall time as the decoding time is just performing XOR on messages with the keystream which is fast. The mobile battery lifetime is also longer. The proposed lightweight security protocol FEATHER will help to provide confidentiality, authorisation and security for users in mobile cloud computing technology and IoT

technology. It also helps to reduce power consumption, which will improve mobile applications' overall performance. The proposed lightweight protocol was analysed against possible known attacks, which showed it is secure for implementation. The MICKEY 2.0 cipher was used as a pseudo-random number generator, however the FEATHER protocol can be adapted to use other IV-based lightweight synchronous stream ciphers. The proposed MICKEY 2.0.85 in Chapter 5, which is 23% faster in generating pseudo-random numbers, can also be used. However, even using MICKEY 2.0 in FEATHER is fast enough. MICKEY 2.0.85 is useful for other smaller applications.

## **6.8 Conclusion**

Security in mobile cloud computing is critical and is a demanding challenge. Improvements in this field are essential for the IoT. By developing a new lightweight security protocol, the FEATHER protocol will reduce cost and reduce time used in the external server. Therefore, it will increase the number of devices communicating at the same time, and will enhance mobile cloud computing applications. It will help external server providers serve more users, and assist more new cloud providers to join this market to meet growing consumer demands. The FEATHER protocol is an important step to fulfill the requirements for secure mobile cloud computing with internet connectivity.

The following chapter continues the implementation of lightweight encryption in small hardware with a real-world application. It presents a secure system including prototype device and secure protocol for communications without internet connectivity using MICKEY 2.0 and the proposed variant MICKEY 2.0.85 presented in Chapter 5 as the optimised tools.

# **Chapter 7: Proposed security cryptosystem with proposed device for security application in eHealth without internet connectivity: Near Field Secure Data Extractor**

## **7.0 Chapter overview**

The FEATHER protocol developed in Chapter 6 requires internet connectivity. This chapter presents a lightweight cryptosystem with RFID technology that does not need internet connectivity. It provides security for extraordinary situations, such as emergencies, remote areas and pandemics, where internet connectivity is not accessible.

Section 7.1 provides an overview of the chapter including the aim of the study and the contribution, Section 7.2 introduces the essential background, Section 7.3 presents the chosen scenario to demonstrate the proposed security cryptosystem, Section 7.4 demonstrates the major processes for the prototype device with security protocol and discusses the proposed cryptosystem in an eHealth setting, Section 7.5 describes the major components of the prototype device, Section 7.6 outlines the RFID tag creation procedures, Section 7.7 discusses key management, storage and rotation, Section 7.8 discusses device processes, Section 7.9 discusses the emulation of the device and present device performance testing, Section 7.10 presents attack analysis, and Section 7.11 presents overall analysis and discussion, and shows other potential scenarios which may benefit from the cryptosystem. Section 7.12 summarises the research contributions and the importance of the proposed cryptosystem in security applications with RFID technology.

## **7.1 Introduction**

This chapter describes the development of a prototype device called Near Field Secure Data Extractor (NFSDE), which has strong security and lightweight protocols for RFID tag communications [221]. RFID tags are widely used for recognition and authentication in sensor networks which are prominent in the internet of things (IoT). Older applications of the IoT suffered from an insufficient defence in low-power systems [222]. Poor or non-existent IoT security has allowed devices to be seriously compromised via the internet [223]. In supermarket authentication, RFID tags require data to be transmitted without touch [224]. This ensures RFID tags are an easy and cost-effective validation method for

technologies such as pass keys, monitoring (such as product monitoring, shipping of products and goods) and transmitting data including package information descriptions and receiver directions. RFID systems offer an efficient alternative, and avoid human error because of the lack of manual interaction as RFID tags include prompt verification that can also be verified directly [225].

Because of these benefits, RFID tags can improve healthcare, especially with an ageing population and the risk of medical errors by healthcare practitioners. Solutions should be sought to ensure medical information is stored safely and efficiently and to retain vital health documents in identification cards. Consumers have continued to ignore data protection and safety requirements because RFID tags are simple to use and cost-effective [226].

Scientists have begun suggesting stream ciphers as an appealing security solution for low-cost implementations [227]. Present stream cipher solutions typically lack authorisation, integrity and authentication by public key infrastructure asymmetric algorithms [228]. Stable key exchange is a continuing problem in cryptography [229], [230]. While asymmetric algorithms should fix this issue, they need more computing resources than are available in small and constrained device systems. [231] selected these issues for a hardware solution and proposed custom hardware called Recryptor. For near-sensor IoT implementation, another study [232] used a specialised chip named Fulmine. This thesis research uses a radically different method to achieve the same goal. The approach is better as it uses off-the-shelf software and equipment, meaning the new method and device is more scalable and affordable, and available for common devices. The approach uses symmetric cryptosystems that are suitable for the modern key exchange approach to maintain privacy, authentication and integrity.

MICKEY 2.0 is a lightweight stream cipher used in physical and portable applications and has different uses including hardware and software systems [233]. While MICKEY 2.0 was developed to incorporate hardware, it is also suitable for software usage. Its efficiency and effectiveness were evaluated compared to Trivium and Grain (see [65] and [171] for more information). Banik [234] proved MICKEY 2.0 was not vulnerable to attacks as it uses an unusual combination to change its internal components, rendering it impossible to find R or S bits (R and S are both linear and nonlinear registers, see Section 5.5). Furthermore, the dynamic internal architecture offers a more robust randomness than

Trivium and Grain ciphers [235] to differential fault assaults. For the reasons stated earlier, users use the specific lightweight encryption MICKEY 2.0. There has been one recent encoding approach for portable, computerised devices [236] that examines eight lightweight different hash functions integrated in block cipher using software implementation. They used a passive cryptographic RFID (CRFID) to detect the basis of these encryption methods in their studies and proposed the MD5 hash feature as the most random among them.

The aim of this chapter is to use a lightweight synchronous stream algorithm that fits RFID technology, recognising the need for low-cost, power and computation efficiency, while using RFID tags without internet access while taking advantage of and retaining the usual advantages of a public key infrastructure (see Table 7.1 for explanation). To accomplish this authentication framework, the chapter develops a prototype device named NFSDE, using the MICKEY 2.0 cipher and carried out a safe key-IV exchange. A prototype tool is developed. The proposed system's device emulation includes a stable protocol for encryption. This device's security relies on a reliable record keeper (R) and secure flash drive (USB).

An example of use is a medical practitioner needing to access substantial medical records (for example, allergies and current conditions) where an individual (patient) is situated in a rural location or in a hazardous zone, and internet or cellular access is either not available or cannot be trusted. This medical case has been selected for this research as one example of how the product is used, because it has the good protection criteria and requirements as a demonstration method. For cases where connectivity is insecure, the definition should be used and modified for virtually any security sensitive application involving connectivity to protected networks. As internet access is not always possible, this new system can even be used to authenticate out-of-band for used devices.

### **Main contributions**

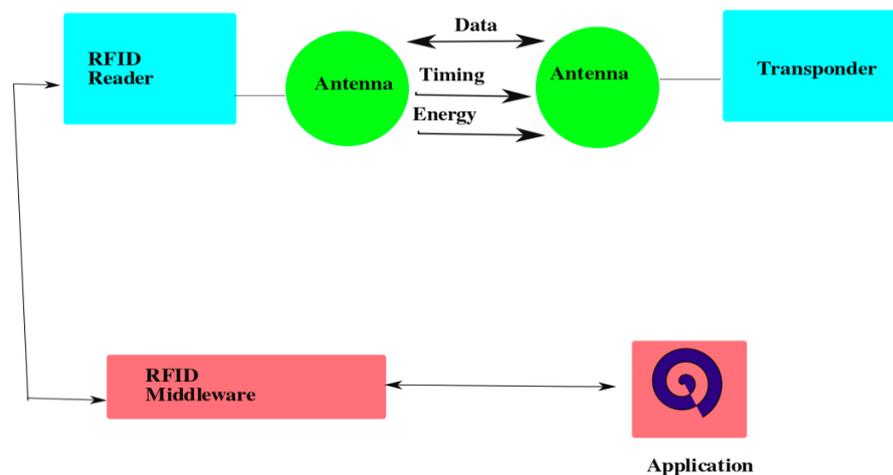
1. This research proposes a protected RFID-based critical data security device named NFSDE. This tool is a concept which could be used directly or adjusted if desired by the customer. It may also be viewed as a reference for building specialised hardware with functionality compatible with the prototype system.

2. By applying the MICKEY 2.0 encryption, a secure eHealth (proof of concept) framework is suggested. The framework is easily adaptable for uses other than those in the health field. This includes a mechanism instead of using a public key infrastructure to make it easier to use various lightweight stream ciphers.
3. The proposed cryptosystem is built using off-the-shelf hardware encryption and streamlined lightweight stream ciphers to improve RFID protection and offer comparable benefits to the public key exchange.
4. It contributes to the creation of robust and secure key and initialisation vector (IV) sharing (multiple key) mechanisms as well as key management and key-update strategies for a secured RFID reader that are not focused on the internet or wireless communication.
5. The process time of using NFSDE to retrieve 4 K of RFID content using MICKEY 2.0 is a matter of milliseconds [221].

## **7.2 Background**

RFID security is challenging in current worldwide applications. RFID technology has become very widespread for authentication, monitoring and object tracking. It is now part of IoT technology. This technology uses radio waves to track objects. For instance in retail, it is used for goods scanning. A label containing the identification number can be scanned to get the price and name of the item and all related information. RFID tags can be in different forms including in cards, wristbands and keyfobs. Advances in technology are making tags smaller with more data storage capacity, and reducing the cost over time.

Figure 7.1 shows the RFID main components of the RFID reader and RFID tags. Tags store the object information, while the reader checks and receives the tag information by implementing stored software. The antenna creates a magnetic range that detects tags. For more information about the RFID technology infrastructure, see [237], [238].



**Figure 7. 1 Illustration of RFID tag basic functionality**

There are two main types of tags: passive tags and active tags. Passive tags do not have an internal power source, so rely on the electromagnetic waves that come from the RFID reader. Passive tags are the cheapest in the RFID system. These tags are used for access verification, contactless payment, goods itemising and labelling for identification, and object tracking [239]. Active tags are self powered by an internal battery. They are more efficient in object tracking as they can identify the object's status and locations. Being self powered they generate signals with a higher range than passive tags, and are more expensive than passive tags. For cost efficiency, the passive tags are more desirable for small verifications and tracking applications [239]. For applications for RFID tags in real-time object tracing and location identification, especially in radio noisy environments, see [240].

This thesis uses an eHealth example as security and medical data protection are very critical in health. The eHealth scenario is used to demonstrate step by step system implementation. The NFSDE device with secure protocol can be implemented in other situations where internet connection is not reliable.

Stakeholders in eHealth implementation include patients needing medical care, and providers of medical care including, for example, paramedics, nurses, medical practitioners, medical records keepers and administrative staff. The patient may make a regular visit or have an urgent situation that requires a medical practitioner.

### 7.2.1 Lightweight cryptosystem

As this thesis cryptosystem is lightweight, lightweight cryptosystems are reviewed. Using lightweight encryption as a security approach, where there are processes like exclusive OR (XOR), which are appropriate for use in smaller devices, is one way to provide fair protection to counterfeit attack resistance as shown in study [241] which presented a lightweight verification protocol for hardware deployment and clarified how their method can enhance protection in RFID.

Research in [18] explain the use of a lightweight encryption framework for RFID applications, as well as for the IoT generally and include an outline of lightweight encryption solutions. They also address the protection offered by every system and illustrate the use of software and hardware. They also demonstrate how lightweight encryption cryptosystems have benefits over using the traditional advanced encryption standard (AES). In recent decades lightweight encryption has become more popular as it provides substantial security and can be used in computation-restricted devices and low memory. Products benefitting from lightweight encryption include paypass cards [242].

While some cryptosystems like AES have better arithmetic capabilities, lightweight encryption makes encryption faster and enables more information transfer in a shorter period. Lightweight encryption enables improved inter-device connectivity. The related research then concentrated on lightweight stream ciphers. The eSTREAM project [19], [65] assessed the proposals of ciphers for their suitability for software and small device hardware. In the final step of the eSTREAM initiative three candidates, Trivium, Grain and MICKEY 2.0 ciphers, were chosen for hardware deployment, however they are compatible with software implementation. Study [242] provides an extended description of lightweight cryptographic algorithms, their implementations and the classification of such systems in terms of their properties and specifications, such as lightweight systems and ultra-lightweight cryptosystem [242].

Lightweight stream ciphers have been used for recognition and verification purposes. The goal of the chapter is to build a secure protocol with a encryption tool with a high level of simultaneous identity authorisation, concentrating on immunity from denial of service (DoS) attacks and compatibility with RFID tags [243].

Lightweight cipher blocks, including LBlock [244], were used for the security of the device with a small number of gate equivalents like 1320 GEs. Nevertheless, another study [245] showed a 23-round LBlock is vulnerable to attack.

Mimicking the Data Encryption Standard (DES), DESL [246] lightweight block cipher was developed to be a similar lightweight encryption cryptosystem to have consistency with the RFID tags, and it was designed to be an alternative competitor to the lightweight stream cipher in the eSTREAM project, which is 25% smaller than DES (45% less than AES) [247].

In situations where the key is reusable and saved for encryption, the data has to be protected. In this sense, the use of electrically erasable read-only programmable memory (EEPROM) is one of the methods suggested. This method has been contrasted with other storage techniques which were introduced in [248].

For low-power computation hardware, many realistic real-life implementations such as RFID tags, need effective encryption. The first to use MICKEY 2.0 as an encryption method was Babbage and Dodd [215]. Depending on the cost effectiveness of RFID tags, they must be tracked, checked and modified to maintain their protection [249], which made this technology effective in securing information with affordable cost. This thesis research uses the identification device, NFSDE and MICKEY 2.0. With the aid of a microcontroller, such as Raspberry Pi, that is paired with a RFID reader and a fingerprint reader, RFID tags can now be verified without the need for internet access.

### **7.2.2 Physically Unclonable Functions**

Physically Unclonable Functions (PUFs) [249] designed for resource-controlled protection became an important research and industrial improvement area. However, PUFs are not suitable for this current use, as PUF products available for the market are still environmentally responsive and are not cost-efficient. PUFs may not have compatible security under severe conditions. Due to the focus on the eHealth scenario which needs to be suitable for unfavourable circumstances, PUFs are not included in this architecture. Recent work demonstrates potential to mitigate this problem [250], [251]. Appendix 7.3 provides more detail on PUF usability with NFSDE and explains improved future PUFs.

### **7.3 eHealth as case study and illustrative scenario**

An example of an eHealth RFID based system includes tracking the performance of students in universities by integrating RFID towards their state of health. These provide patient history, health reports and relevant clinical details such as blood pressure and medication scripts [252]. eHealth applications require strong protection. However, if internet communications are untrustworthy, potential risks by attackers will be aggressive [253] because the current protocols have to be adapted in order to resolve connectivity problems and to include cryptographical methods suited to this case. Nevertheless, they offer approaches on open and untrustworthy networks in their study [253] whereas the system developed here gives protection whenever the network is weak or absent.

Researchers emphasise the value of laws and guidelines to maintain secure contact in order to examine emerging eHealth applications in the IoT [254] and to check further in-depth analysis. See [255-257] for more detail on IoT protection issues and technological upgrades.

#### **7.3.1 eHealth scenario description**

In some situations there is no internet connectivity such as remote places or disaster areas, or when internet coverage malfunctions for any reason. The security and privacy of patient data is still critical, as unauthorised people can delete or manipulate patient health data status, and sensitive information such as medical condition and medicine dosage, which could be life threatening. The eHealth scenario is chosen as an example for demonstration purposes for NFSDE devices and the security protocol.

One application of implementing the RFID technology in eHealth monitoring is linking students' RFID tags for their progress performance at university with their health record, which can include their medical history, current health status, current medications if any and also the relevant useful health data. Some students may have heart conditions, asthma, diabetes or blood pressure, and they may have prescribed medications for these medical conditions [252]. Where there is no internet connectivity, the proposed protocol is important, as the data needs to be protected considering the attacks can be more brutal, which can alter or modify or even delete critical health data [253]. The current protocols must address the communication challenges by modifying and fixing the used

cryptographic methods that are customised for such situations [253]. In a research study, [255] proposed a security solution that targets untrusted or anonymous networks. On the other hand, this protocol provides security in cases where the internet connection is either weak or not available. More insight about eHealth applications and current IoT real time healthcare technologies is in [254], which also emphasises the importance of establishing strong policies, guidance and protocols that regulate the communications to ensure security. [255-257] review the current security challenges in IoT applications.

### 7.3.2 eHealth scenario process

This section describes how to implement the proposed cryptosystem. Table 7.1 describes the notation used in this chapter.

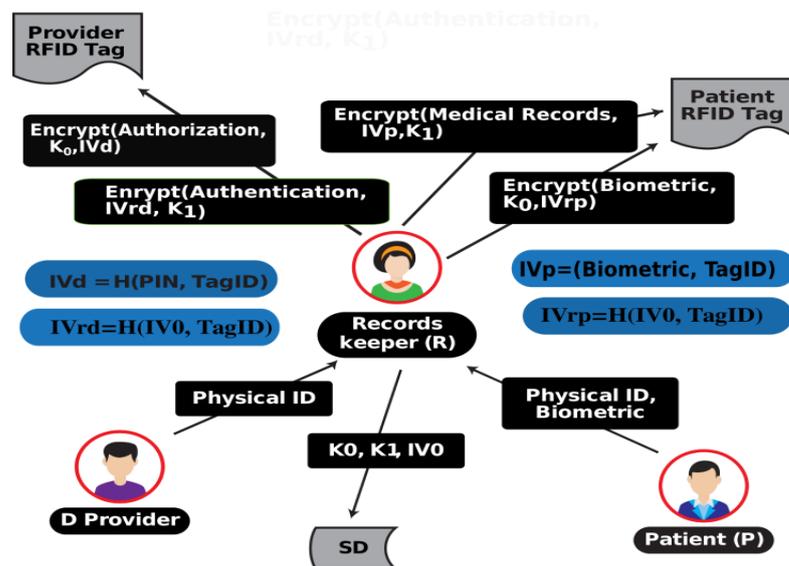
**Table 7. 1 Notation description**

Notation	Name	Explanation/Notes	Notation	Name	Explanation/Notes
R	Medical Record Keeper	This is a role and indicates a person who has privileged access to all data and cryptographic secrets. The person acts as a gatekeeper to sensitive data and authorisation. This role is likely to be fulfilled by multiple individuals depending on the administrative organisation of the issuing authority.	$K_{1e}$	Data key stored in the SD before decryption	When the provider enters the passcode for the SD(USB), $K_{1e}$ is decrypted to $K_1$ .
P	Patient	Is a person in an emergency or a remote situation, where a reliable internet connection is not available. Emergency or rescue workers represent an appropriate use case.	$IV_0$	IV “seed” for the record keeper after decryption, which is considered a “Secret”	$IV_0$ is used as a starting point for generating the initialisation vector for encryption of authentication. It is XORd with a unique tag ID to create the final authorisation IV. For the provider, it is XORd with the provider’s unique tag ID to create $IV_{rd}$ . For the patient, it is XORd with the patient’s unique tag ID to create $IV_{rp}$ . These hashes are used because they

					consume low power/CPU.
D	Medical Provider, (eg EMT, paramedic, nurse)	The provider has a device to read the RFID tag of the patient.	IV <sub>0e</sub>	Encrypted IV <sub>0</sub> (in the SD) for the record keeper	When the provider enters the passcode for the SD, IV <sub>0e</sub> is decrypted to IV <sub>0</sub> .
SD	Secure Flash Drive (USB)	It has a “keypad,” which is used to independently encrypt/decrypt data stored therein.	IV <sub>tp</sub>	IV for authenticating patient signature	Created by XORing IV <sub>0</sub> with the patient’s unique tag ID. These hashes are used because they consume low power/CPU.
K <sub>0</sub>	Key to Encrypt/Decrypt Patient Data	K <sub>0</sub> is read from the SD. Together with IV <sub>p</sub> , this is used to encrypt or decrypt patient data by using MICKEY.	IV <sub>rd</sub>	IV for authenticating provider signature	Is created by XORing IV <sub>0</sub> with provider’s unique tag ID. These hashes are used because they consume low power/CPU.
K <sub>0e</sub>	Data key stored in the SD before decryption	When the provider enters the passcode for the SD, K <sub>0e</sub> is decrypted to K <sub>0</sub> .	IV <sub>d</sub>	IV to encrypt authorisation for provider	Is created by hashing the checksum of the provider’s PIN with unique ID through concatenation. These hashes are used because they consume low power/CPU.
K <sub>1</sub>	Key to Encrypt Authentication (from the record keeper)	Authentication from the record keeper is encrypted with MICKEY using K <sub>1</sub> , IV <sub>tp</sub> (for patient), and IV <sub>rd</sub> (for provider)	IV <sub>p</sub>	IV to encrypt patient data	Is created by concatenating checksum hash of fingerprint template with unique patient tag ID. These hashes are used because they consume low power/CPU.

Source: [221]

In the eHealth scenario, as shown in Figure 7.2, the patient (P) will be given an RFID tag which can store around 4 KB of encrypted patient health information by the MICKEY 2.0 cipher. There are many ways to carry the tag as the patient can use a card, wristband or key fob that contains the tag. The patient’s tag needs to be certified for encryption by those in charge of storing the medical records.



**Figure 7. 2 Relationships between the individuals in eHealth scenario**

To create the patient tag, first the record keeper (R) will scan the tag to ensure ID uniqueness, then as a biometric security and identity measure the patient’s fingerprint will be scanned, the card will be read, encrypted and then signed cryptographically with the medical data. After that, the medical data will be written and stored on the patient’s tag. The patient keeps this tag to ensure that, in a medical emergency, important clinical and medical information is readily available for medical treatment providers. A provider (D) is any medical professional (doctor, nurse, EMT, paramedic, etc.) who might require fast right of entry to a patient’s medical information such as existing conditions and allergies.

The medical provider needs an RFID tag for RFID reader activation to permit decoding of the patient’s medical information. The record keeper (R) creates an authorisation tag by confirming the provider’s identity and the provider level of authorisation (to access some necessary information, not all tag stored data).

The provider will be supplied with a private number (PIN) that is used to encode their card. The identity of the medical provider and level of authorisation are encrypted and the record keeper will sign the previous information cryptographically. Then all this information will be stored with encryption on the provider tag.

To view the medical record, the approved medical provider simply has to check the patient’s tag by scanning it, as well as scanning the patient’s fingerprint. Providers will have an encrypted USB drive, their RFID ID card, and a PIN to authenticate.

The authentication procedure has a similar process as a debit card transaction (retrieval of account data by the payment issuer for checking during the check-out process), but requires less time. Such authentication procedures only take place once a session. Figure 7.2 shows the details received by each of the entities (Record Keeper (R); Provider (D), Patient (P)) as well as the position of every person's task. The record keeper (R) will validate the medical provider (D) and the patient's (P) medical information (data) and biometrics on the patient's RFID tag. The provider (D) must activate the NFSDE using their PIN and their RFID tag. If the individual (P) has a biometric identification matched, the medical provider (D) should be able to access the patient's private medical information using the NFSDE.

### **7.3.4 Proposed cryptosystem as solution for eHealth setting**

Several realistic applications need efficient encryption for low-power hardware. Cost is mostly the motivating factor; however, another more critical aspect includes circumstances where internet quality is inconsistent or non-existent. In such a scenario, sensitive data will need to be locally stored, safe "at rest" and open to low-power and small devices users.

Encryption is an essential requirement for personal identifying information and confidential records, such as medical or financial information. In study [258] discuss implementing personal identifying information for privacy of information, in particular for eHealth. Notably, once a reasonably stable lightweight encryption has been created, no more significant advances are required for protection and privacy.

An important concept of "privacy by design" is that secrecy is not a zero-sum game [259]. Technological constraints cannot be a justification for violating privacy.

The new proposed secure framework has three elements of security: secrecy, integrity and availability, also known as the CIA Triad [260]). The third element, "availability", is very critical, because it actually allows vital data to be accessible in cases where the internet may not be available as this thesis protocol provides. It is also part of "security in Depth" [261]. Compared to the CIA triad, the suggested program incorporates the three As of data security: authorisation, authentication and accountability [262].

The eHealth example illustrates that technological restrictions do not always have to be a source of protection limitations. Data protection is critical even when connectivity and/or electricity is not available, for example in rural areas and in disaster situations, as well as during network outages or maintenance. Lightweight encryption for secure data storage strategies can be crucial in these circumstances. This thesis, as an indication of the implementation of the suggested method, finds a situation in which vital medical details have to be available and stored on the tag for the patient. The internet and cellular connectivity are unstable in this situation. Situations may include rural regions lacking internet connectivity or a crisis scenario where communication equipment has been affected. For such cases, the battery life of an activated unit, such as an RFID reader, is likely to be crucial, because the capacity to refresh will be restricted. Workers or security forces in remote dangerous areas can urgently require medical care. Gaining immediate access to vital medical details can help save lives. This proposed method and prototype was developed for circumstances (such as urgent and serious medical emergencies) where delaying access to critical health data would be extremely undesirable. As a consequence, a considerable amount of time is expended on the “up front” procedures for processing the data and executing verification and authorisation. However, when it is essential to access the data, the processes are designed in such a way as to be fast and reliable. Authentication and permission only take a few seconds, partly because the keystream generation algorithm (MICKEY 2.0) is fairly fast and partly because the number of steps required to reach the reader without losing protection has been reduced. The test method used MICKEY 2.0, but the method is expected to be true for other lightweight ciphers, such as Trivium and Grain. Any threat will be ineffective because the keys and the IVs are secured by a functionally protected tool (USB).

The current protected cryptosystem is very robust. For example, the two key components of the MICKEY 2.0 encryption cipher implementation and the NFSDE components with different application modules may be adapted and changed to be acceptable for the desired applications, allowing users to be connected to a defined protection solution, while promoting consumer ease of usage and innovation.

## **7.4 Major processes of eHealth scenario setting and practical low-power ciphers applications**

This section explains the main processes in the eHealth scenario, why the eHealth scenario was chosen, as well as some other potential scenarios where the proposed cryptosystem can meet security needs.

According to cost or performance considerations, the protection capabilities of certain RFID based systems are disabled. Several researchers have been able to establish cryptographic methods and imitate low cost hardware communication protocols [263], [264]. In [265] refuted the mistaken assumption that security is hard to achieve and that privacy is too costly which has generated an adverse circumstance that has ultimately influenced the whole world. The ideas and realistic recommendations outlined here prove that privacy and protection can be simple, cheap and efficient.

The eSTREAM ENCRYPT project [65] was introduced to resolve the problems alluded to above. Because protection requires more than just using algorithms, a full framework is described to demonstrate security, privacy and authentication at low cost for a low power small device. This thesis suggests a novel prototype device named NFSDE, which does not need an internet link and enables confidential information to be read and written with encryption to the RFID tag in a secure way. The new framework in this thesis meets the existing standard practices for secrecy, integrity and availability [266] with the three As for data privacy: authorization, authentication and accounting.

The proposed security system is intended for use in circumstances where the internet may be stopped for security purposes, or the internet is not accessible, such as in disaster or remote areas. The system concept is a prototype device that requires low power, has minimal memory requirements, and is lighter than a traditional computation machine. Such features are useful in cases of emergencies, where a portable system is needed or where the supply of electricity is not sufficient. Adapting such technologies has culminated in early adopters neglecting encryption which may harm the online community [266].

Many experiments on lightweight encryption tend to have concentrated on automatic object detection, for example shipping boxes, cars and robots. In research [224] the scholars rely on confidential personal data being given and sufficient; this inevitably increases uncertainty and criticality due to the presence of external human identification

methods, including biometric reference identification and the “eyes-on” authentication of the patient’s identity.

The proposed prototype works as tools and processes allow time to be invested “up front” to approve and authenticate RFID tags. However, the read time requires just a few seconds and vital data can be retrieved easily in unfavourable circumstances, as shown by the device simulator.

Since this is a standard-based tool and does not need to be tied (allowing adapting, altering and modification of the NFSDE components), hopefully it will encourage users to use it to apply protection to their own low-power ventures or to embrace these ideas and to improve them beyond eHealth. They may modify and alter these security features provided by the proposed secure system to be suitable for many applications.

The eHealth scenario was chosen as it is extremely security vulnerable and demands the most robust protections due to privacy issues, data criticality and speed of access. The suggested secure framework addresses the “worst-case” situation for secrecy, integrity and flexibility without sacrificing the low-power requirements.

## **7.5 NFSDE device components**

This section presents the NFSDE components used by medical providers to allow them to read medical records quickly and in a secure manner. The most optimal components were chosen to make the NFSDE perform at full potential. However, these components are an example to illustrate the concept of the NFSDE device, and users are free to choose different components as long as they are compatible to work together in NFSDE and perform the required and defined functionalities. The selection of the NFSDE components was based on cost and performance including data access speed and an overall high level of protection. In addition, with advanced technology, NFSDE components can be modified and improved which prove the prototype is flexible for alteration and adaptation. Figure 7.3 shows the prototype contents.

### 7.5.1 NFSDE device components

The components were chosen to be cost efficient, and to be compatible with each other, however, they can be replaced with cheaper components as long as they still work as one unit, as advances in technology will result in more powerful hardware at lower price, so the flexibility of NFSDE is a goal itself. The security protocol and NFSDE work together as a secure cryptosystem, and the NFSDE is a flexible prototype to be used as a guide for implementation based on users' specific requirements.



Figure 7. 3 Core components of NFSDE

#### Storage – Secure USB drive

The most challenging aspect of any security protocol is storage and delivery of the secret keys. A regular storage device usually is not secure as it is possible to access the device physically or using side channel frequencies analysis which may expose the stored keys. On the other hand, using external servers for key storage and exchange is insufficient due to no internet connection. However, the key updates and backup can be optional through cloud based storage services such as provided by AWS.

The second challenge is tag authorisation and authentication which is a record keeping job. Thus, it is essential to have lightweight storage and portable devices.

These challenges were overcome in this thesis proposed NFSDE device by using an encrypted USB device that uses passwords for access, and uses AES 256-bit for encryption at a low cost of around \$10 [267]. It is necessary to have policies that arrange the key storage in the USB and not stored in the RFID reader if not attached to the USB, and the

USB encryption must be updated based on a regular scheduled time. Regularly updating the keys rotation and the USB encryption will mitigate possible attacks and security issues. One key recovery will not affect other keys as discussed in the attacks analysis section.

Other storage devices include EEPROM [268]. Assuming EEPROM is internal not external, it may be used for the keys storage, however, it is not immune from the side-channel attack [269]. Other possible tools are Physically Unclonable Functions (PUF) [250], [251]. The PUF is not currently cost efficient, however in the future it can be adapted within NFSDE.

### **CPU – Raspberry Pi**

The central processing unit (CPU) is the heart of the NFSDE. Raspberry Pi [270] can act as the computer within the NFSDE. Raspberry Pi is low cost and consumes minimum power [271]. Thus, it is considered the optimal solution for resource constrained devices, and it is also used for cryptographic implementation in blockchain [272].

Although small components were chosen for the NFSDE device, the potential implementation of the secure protocol does not require all the power the Raspberry Pi can provide, as it only computes the simple bit shift and XORing with few ANDs operations. NFSDE only needs a single Raspberry Pi and a few NFSDEs are needed by medical practitioners, which will reduce the overall cost, as most components are the tags which are very cheap. Every medical provider and patient needs one RFID tag.

Raspberry Pi can handle decryption, and can process the operations needed for the RFID reader and the fingerprint scanner. A low cost option that has enough power to do the previous process is Adafruit PiTFT (320×240, 2.8) which can be accompanied by a touchscreen for manual input [273], [274].

### **Fingerprint reader – Hamster Pro 10**

Hamster Pro 10 is a relatively new portable fingerprint reader device that is able to create a 500 byte template that is enough for the proposed system, which is a great option for patient fingerprint reading [275]. Hamster Pro 10 detects if the fingerprint scan matches the stored reference fingerprint template. While a fingerprint is a type of biometric data, some adjustment is required.

The system setting can be altered for multi-fingerprint scan recognition if the primary finger is injured, taking into consideration that each finger needs 500 bytes of reference template stored in the RFID card. Different biometrics may be used instead of a fingerprint, however, the fingerprint is faster for identification.

### **RFID reader – MFRC522**

The MFRC522 is a very low-cost RFID reader and a very suitable choice for prototype functionality. It is able to process the write and read operation on the standard tags as well as MIFARE tags [276].

### **Tags**

MIFARE tags with 4K storage capacity [277] were considered sufficient storage for the prototype device. The contactless property can be in wristbands or key fobs, which makes MIFARE suitable for patients, as well as swap cards like ID cards for medical practitioners. Other forms may be used for specific medical providers or patients. The functionality of the device can be adapted to support different forms of RFID tags.

In future CRFID tags that use PUF [250] can be used for patients as well as providers, however the current cost is not feasible, as every patient and provider needs a minimum of one RFID tag which will impact the overall cost. However, in the future with advances in technology and cost reduction, CRFID tags could be used in this device, which shows the flexibility of the proposed system.

## **7.6 Tag creation and implementation process**

This section provides details of the tag formation phase to both the provider and the patient, also describes the development steps. Figure 7.4 shows the record keeper rules and provider tag creation.

### **7.6.1 Creation of the medical provider tag**

This section provides a step by step demonstration of the creation of medical provider processes, illustrating the data flow and process flow, and explaining the authentication, authorisation and encryption/decryption process.

## Use of MICKEY 2.0

The MICKEY 2.0 algorithm is used for two separate purposes inside the provider tag. The first application of MICKEY encoding on the provider tag is to encrypt the “provider identification” and “authorisation” sectors. The common assumption is that a practitioner needs only a fraction of the patient’s data to be processed. Once the authorised field is decrypted by the NFSDE reader, the software on the NFSDE machine is designed to show only the data that the provider is allowed to access depending on the validity of the permission area.  $IV_d$ , which is the IV for the provider, is unique to the provider as well as the RFID tag and can be calculated by generating the PIN chosen by the provider and the specific tag ID. In addition, the  $K_0$  hidden key is still used.  $K_0$  is a variable over time. MICKEY 2.0,  $IV_d$  and  $K_0$  are used to encrypt the permission and identification data before being placed on the tag.

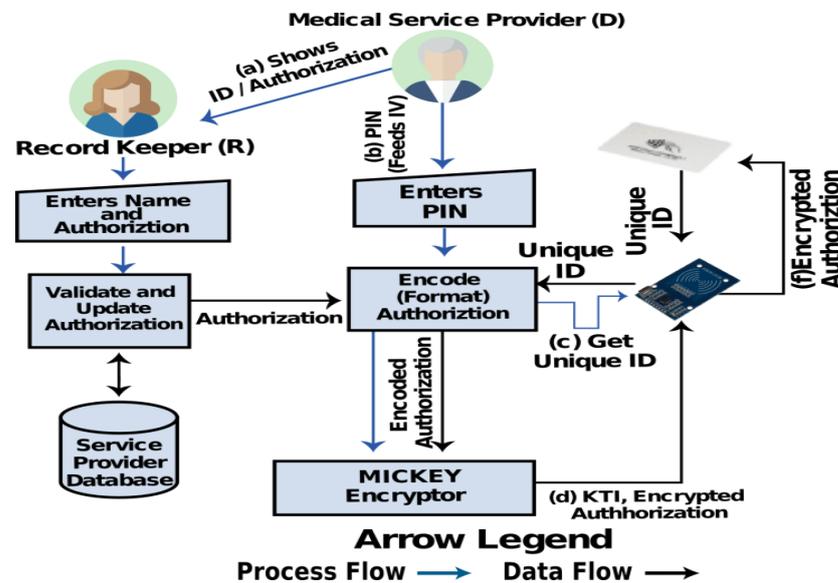


Figure 7. 4 Processes for provider tag creation

Once data is authenticated, a 32-bit cyclic redundancy check (CRC) [278] is computed for use in the next step. While 32 bits are not powerful enough to construct a cryptographically efficient hash, it can be used to avoid a collision attack [279]. This restriction is solved by encrypting the (hash) by MICKEY 2.0.

The decryption method with MICKEY 2.0,  $K_0$ , and  $IV_d$  first completes the activation of the RFID reader. The second main work of the MICKEY 2.0 algorithm is to authenticate it, as integrity is one of the foundations of information protection. This is necessary to recognise that the details are derived from the intended source and have not been tampered

with. As the world is resource-restricted, lightweight signatures must be used without losing credibility. In order to ensure that the data has not been tampered with, the 32-bit CRC of the authenticated authorisation information is used. To ensure the validity of the name, the second 32-bit CRC string “common salt” +  $IV_0$  + specific provider ID + KTI (key time index) is used.

An authorisation is validated using special attributes of all bodies to ensure that the “relevant object” has been approved by “this record keeper”. The  $IV_{rd}$  is IV which is generated by hashing the hidden  $IV_0$  and the generic RFID unique ID. This fourth key is special to the tag and the record holder. In fact, the  $K_1$  hidden key is used.  $IV_0$  and  $K_1$  differ over time, as measured by the KTI. The 32-bit CRC of the cryptographic authorisation is concatenated to a signature composed of a 4-byte (32-bit) CRC of “common salt” +  $IV_0$  + special ID + KTI. This authentication string is protected from intrusion by encrypting a concatenated string with MICKEY 2.0,  $IV_{rd}$  and  $K_1$  that has two implications on it. The first concept is data consistency, where the first CRC ensures the validity of the identity and authorisation areas, while the second CRC ensures the legitimacy of THIS code and this record keeper. The record keeper can authenticate the validity of the authorisation through using two lightweight functions. This signature sequence is placed in a specific data field on the patient data document, and the two values of the CRC guarantee the consistency of other essential fields stored on the provider tag, offering low power [222], [280] yet efficient security against interference. Once the tag is interpreted by the NFSDE system, the signature is decrypted with  $IV_{rd}$  and  $K_1$  and then checked for matching with the predicted value of the CRC (“shared salt” +  $IV_0$  + specific ID+KTI). In fact, the CRC of the authenticated authorisation is determined. The tag is assumed to be true if the decrypted area fits all CRC calculations.

### **Creation steps and scenario**

Usually, the development of an RFID card to enable the NFSDE system will be part of the on-board phase when a care practitioner (paramedic, nurse, EMT, physician, etc.) enters the medical organisation. The “record manager (or keeper)” may be a worker in the HR section. Figure 7.4 demonstrates the method of developing a healthcare provider tag.

The method of authentication is as follows:

1. Engagement between both the provider (D) and the record keeper (R), involving confirmation of the identification and authority of the provider to access relevant patient records as shown in Figure 7.4(a).
2. The provider shall insert the PIN as seen in Figure 7.4(b) of the situation. (See stage 4 for usage of the PIN).
3. The special tag ID token is read and accepted by the RFID reader as seen in Figure 7.4(c).
4. The PIN and the special ID on the card are hashed to construct the specific  $IV_d$  ( $IV_d = h(\text{PIN}, \text{ID})$ ).
5. Verification and authentication are encrypted using the MICKEY 2.0 algorithm, and using  $K_0$  and  $IV_d$ .
6. The latest KTI is stored on the RFID tag by an RFID reader, as seen in Figure 7.4(d).
7. The encrypted authenticated Identification and Authorisation fields are listed on the RFID tag as seen in Figure 7.4(e).
8. The CRC hash of the encrypted sector is determined to identify if the information has been manipulated.
9. The encryption field is authenticated using the two CRC values mentioned above to ensure both reliability and manipulate resistance.
10. The authentication area is stored on the RFID tag.

### **7.6.2 Creation of the patient tag**

This section describes the patient tag creation process.

#### **Use of MICKEY 2.0**

The patient tag includes vital patient health care records that may be helpful while the patient is in a place where internet communications are not possible, such as a rural environment or even an emergency scenario. The use of MICKEY 2.0 includes four different uses under the patient unique tag. The first goal is to ensure that the tag has been approved by the record keeper by using specific security features of the record keeper and patient. The  $IV_{tp}$  is generated by a secret  $IV_0$  and a specific RFID tag ID. Each  $IV_{tp}$  is exclusive to the record keeper and the patient tag. In fact, the hidden key  $K_1$  is used.  $K_1$  differs every time used.  $K_1$  is stored on the USB and indexed by the KTI, which is stored on the patient tag in the field.

The second task of MICKEY 2.0 is to encrypt the fingerprint reference info. The General Data Protection Regulation (GDPR) allows biometric data to be called privacy identifying information for protection purposes. When the tag is formed, the reference fingerprint is scanned and summarised in the 500-byte ISO IEC 19794-fingerprint template [281]. This prototype is authenticated with MICKEY 2.0,  $IV_{rp}$  and  $K_1$  and is placed on the RFID tag.

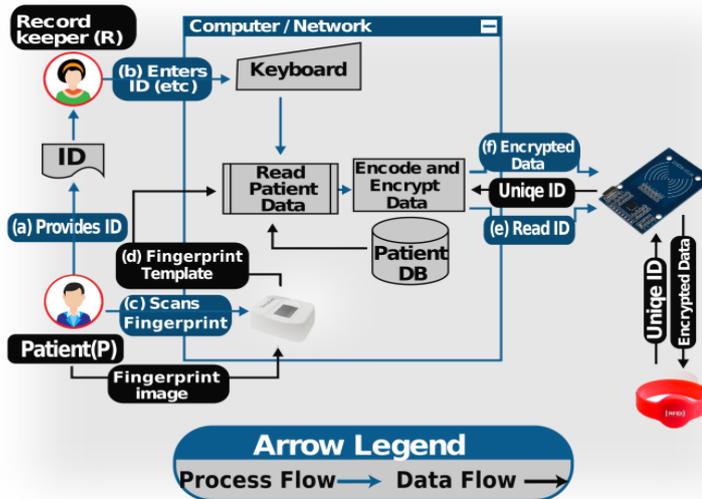
The third rule of MICKEY 2.0 is to encrypt and decrypt medical info. The  $IV_P$  needs to be special to the individual (patient) and its unique tag.  $IV_P$  is computed from an unencrypted 500-byte reference fingerprint prototype and a specific tag. In addition, a hidden key  $K_0$  is used, which differs over time and is indexed to the KTI. MICKEY 2.0,  $IV_P$ , and  $K_0$  are used to encrypt patient details until the data is placed on the tag. After the medical data is authenticated, a 32-bit CRC hash is computed for the next phase.

Authentication is the fourth objective of MICKEY 2.0. Two CRC values are calculated: one for the authenticated medical details and the other for the authentication string (“shared salt” +  $IV_0$  + specific ID + KTI). The two fields are concatenated and authenticated with MICKEY 2.0,  $IV_{rp}$ , and  $K_1$  respectively.

After validation of the tag as genuine,  $IV_{rp}$  and  $K_1$  are used to decode the comparison fingerprint file. Then  $IV_P$  hash is determined from the reference fingerprint example and the special tag UID. The fingerprint scanner matches the reference fingerprint to the freshly scanned fingerprint. If the fingerprints meet the ISO standard, the medical details will be decrypted and shown to the provider.

### **Creation steps and scenario**

The steps to build a patient tag explain the operation. It will normally arise when a patient or clinic visitor signs “out”. The measures available to create and authenticate a patient tag are shown in Figure 7.5. Appendix 7.1 provides the details for these steps.



**Figure 7.5 Processes for patient tag creation**

1. The patient is introduced to the registrar (or medical record keeper) to review the patient Identification (i.e. hospital sign-out), as seen in Figure 7.5(a).
2. After verifying the personal ID, data is retrieved as seen in Figure 7.5(b).
3. As shown in the chart, the patient's fingerprint is checked as in Figure 7.5(c).
4. The logger scans the biometric identification.
5. When the biometrics are the same as the individual, the signature is summed up in the signature prototype ISO/IEC 19794-2. It is the fingerprint "template", as in Figure 7.5(d). If this biometric does not follow the criteria of the patient registry, it will take necessary action. See Appendix 7.1 for a comprehensive scheme.
6. The RFID reader checks the single tag ID, and produces  $IV_P$  ( $IV_P = h(\text{TagID}, b)$  where  $b = \text{biometric}(\text{fingerprint})$ ) ( $IV_P$ ). See Figure 7.5(e).
7. MICKEY 2.0,  $K_0$  and  $IV_P$  are used to encrypt the medical information.
8. For the authenticated medical information, a CRC hash is determined.
9. For GDPR: Practitioner (GDPRP) [282], enforcement test and written on the RFID tag ( $IV_{rp} = \text{Hash}(\text{Tag ID}, IV_0)$ ), the code is encrypted with  $IV_{rp}$  and  $K_1$ .
10. For the "Common Salt" string +  $IV_0$  + Special ID+KTI a CRC is determined. This is encrypted and written on the name, as shown in Figure 7.5(f), to the previous CRC.
11. The actual KTI will be saved on the tag.

## 7.7 Key generation, storage and distribution

The NFSDE tool contains two of the IVs and also two keys, with one of them  $IV_0$  used to produce one of these IVs, and the values should be modified on a periodic basis. The pace of such transition depends on the institutional strategy. Every series of keys and IVs ( $IV_0$ ,  $K_0$ ,  $K_1$ ) has to be created randomly at a regular time and allocated to the KTI. The preservation of those three parameters is essential to the security of the system. Once a new collection of the keys and the IVs is created, then modified values must be saved to the USB as seen in Table 7.2 as an illustrative example. The “then” KTI is stored on the tag, also the KTI is for checking for the correct keys and the  $IV_0$  at the point of verification and decryption.

**Table 7. 2 Key generation, storage and time index update**

Key Time Index (KTI)	$K_0$	$K_1$	$IV_0$
Feb 1	OzQjPQQckv	mz9YlgUgvB	mfE6c2IJrt
Feb 2	fD3Er9NyF7	ZozT7OGzPv	fZpM9ZP4tm

The contact between the medical provider and the record holder consists of two parts: 1) the initial development of the provider’s ID tag, and 2) the frequently scheduled updating of the USB comprising the latest KTI, the main keys group and  $IV_0$ . The timing of the keys update is a policy issue, although for logistics reasons, all USB changes will not take effect until the latest key sets are required to encrypt patient records.

Interaction between the patient and the record keeper takes place immediately and if the tag has to be changed. It will be easier and more practical, for logistics considerations, to give a fresh tag to the individual once the documents are changed. The older tag must be totally erased in a cryptographically safe manner.

The record keeper needs to perform the following tasks:

1. Create the  $K_0$ ,  $K_1$  and  $IV_0$ .
2. View patient records in plain text format.
3. Confirm the provider’s degree of authorisation.
4. Create and upgrade the stable USB hard drive.
5. Confirm the patient’s ID.
6. Identify assurance of the provider.

7. Provide identification of tags (both medical practitioner and the patient).

Clinical record holders will allow physical access to the following:

1. Secure USB
2. Tags
3. Tag reader / writer
4. Fingerprint scanner (providing patient tags)
5. PIN pad (providing medical provider tags).

Authorisation to view the patient information log and provider information system will be given to the medical record keeper.

### **Key rotation**

The simple and main features of NFSDE protection rely on the use of a safe physical USB. For security purposes, keys will be changed on a daily basis according to a defined timetable. Key file storage could be based on an authenticated external cloud server (i.e., AWS Secrets Management System) and the user can access out-of-band. The certified medical provider could retrieve a key file from the cloud and then save it on a protected USB drive.

## **7.8 Device processes**

This section provides an overview of the NFSDE device activation, and the process for the patient data extraction.

### **7.8.1 Device activation (unlock)**

The NFSDE device must be enabled (unlocked) by an accredited healthcare professional to display the patient details. Figure 7.6 demonstrates the method of opening the NFSDE device. The provider has its own special RFID tag, which has been cryptographically authenticated by the record holder. The provider also brings a secure USB drive with time-indexed keys and  $IV_0$ .

The provider will also have its own PIN to access the machine. Figure 7.6 demonstrates the activation phase. The NFSDE enabling (unlocking) phases will be as follows.

1. Provider (D) inserts a password on the protected Flash drive that is attached to the Raspberry Pi on the NFSDE machine. This input is only needed once for every session. Please see Figure 7.6(a).
2. Figure 7.6(b) shows that after the provider scans the RFID tag, the device validates whether the tag belongs to the provider and if it has a correct KTI.
3. The NFSDE machine confirms the authorisation by interpreting  $K_1$  and  $IV_0$  of the USB. Provides  $IV_{rd}$  ( $IV_{rd} = h(IV_0, \text{provider UID})$ ) and, by using MICKEY,  $K_1$  and  $IV_{rd}$ , decodes that signature and verifies the verification code and the CRC.
4. In case the tag is verified, the provider must insert their own PIN as seen in Figure 7.6(c). The NFSDE evaluates the degree of authorisation by decrypting the identification and authorisation fields by using MICKEY,  $IV_d$  ( $IV_d = h(\text{unique ID}, \text{PIN})$ ) and  $K_0$ .
5. The NFSDE is now able to show the data that the provider is allowed to access. (The real activation process is similar to withdrawing cash from an ATM.)

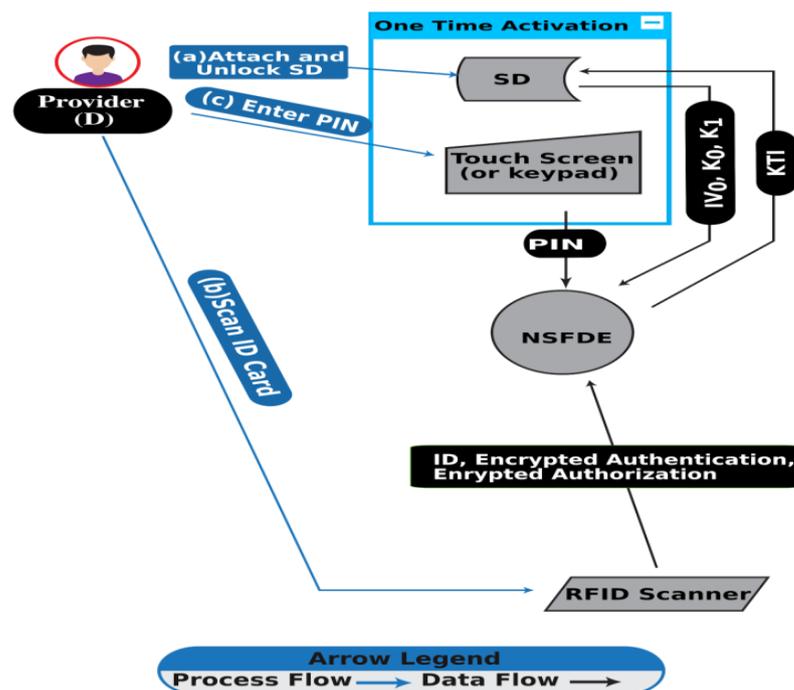


Figure 7. 6 NFSDE activation and unlocking processes

### 7.8.2 Procedure for reading the patient medical record

Figure 7.7 demonstrates the framework for the procedure of how to use the NFSDE device.

1. As in Figure 7.7(a), the patient RFID tag is checked.
2. Reference fingerprints ( $IV_{rp}$ ,  $K_1$ , MICKEY 2.0) are decrypted and issued to the fingerprint scanner as shown in Figure 7.7(b).
3. After testing the patient's fingerprint, if the patient's fingerprint matches the reference fingerprint, the medical information will be decrypted, as seen in Figures 7.7(c) and 7.7(d) respectively.
4.  $IV_P$  is calculated as  $IV_P = h(\text{reference fingerprint template, specific tag ID})$ . It is then possible to use  $IV_P$ ,  $K_0$  and MICKEY 2.0 to decrypt medical information.
5. The NFSDE displays the medical information corresponding to the degree of authorisation specified during NFSDE activation as seen in Figures 7.7(f) and 7.7(e) respectively.

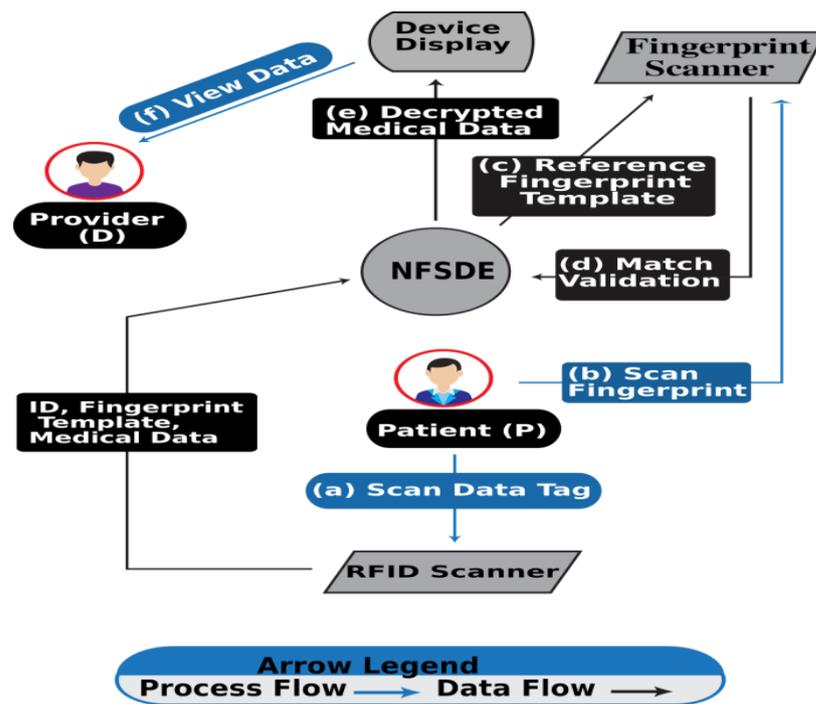
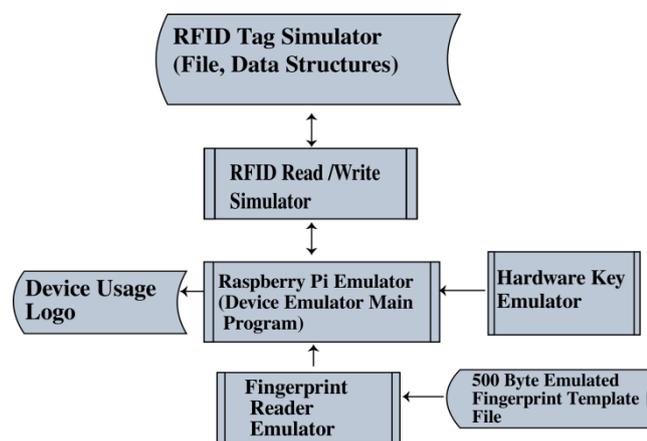


Figure 7. 7 Display the patient data process

## 7.9 The emulation processes for NFSDE and testing

### 7.9.1 The emulation processes for NFSDE

To create a software emulation, a C-language emulator of the NFSDE system was developed to illustrate the main processes and components, as seen in Figure 7.8. The key parts are a single card processor, a USB, a fingerprint sensor, a RFID reader and a RFID writer. Main procedures involve the development (and encryption) of both the provider and the patient tags, the validation of these identifiers, the activation of the NFSDE by the provider, the decryption of medical information and showing the medical data by the provider. Device-level approval is registered (for authentication purposes) in a file called device-log.txt. If the simulator generates both patient and provider tag recreations, the process will not be logged as, in reality, that would be performed on different systems, which are likely to have their own logging procedures.



**Figure 7. 8 NFSDE device software emulation**

The ISO/IEC fingerprint representation is 500 bytes long. Three files of 500 hex bytes each were used to mimic the fingerprint scanner. Every file represents the ISO/IEC fingerprint representation of the user. These values were arbitrary. The number of bytes is an essential feature of the emulator; thus, the size of the RFID data frame must be accurate and the calculation of the encryption speed should be precise. Standard file actions (fread, fwrite) were used to simulate the reader and writer of the RFID tag. Data structures were built to comply with the 4 K MIFARE requirement. Such data structures have been developed in order to be interpreted from an ordinary file. Command line software was developed that contains the following options: (1) build patient tag, (2) build provider tag, (3) trigger reader, (4) interpret patient tag, (5) unlock accessible drive, and (6) modify key number (time stamp emulator). Every choice requests input throughout the simulation of the

element. The rationale of every procedure followed those steps mentioned above. All of the functions mentioned above are executed or modelled.

### **7.9.2 Testing and running the emulator**

The simulator is a command line interface that shows a numerical screen. Every menu element reflects a step in the cycle of construction or validation. Appendix 7.2 presents step-by-step guidance for how to operate the simulator. Appendix 7.3 has specifications of the menu. Appendix 7.2 presents step-by-step guidance for how to operate the simulator.

Testing the NFSDE performance is important, as medical providers attending patients need to be as fast as possible. By using a program to simulate the device speed, the two main processes are how long it takes to unlock the device, and how much time is needed to read the patient data.

#### **Time for NFSDE unlock**

A software emulator was used to test the whole process for the NFSDE devices. It starts with NFSDE unlock time, which will occur only one time per session (for example when the medical provider starts their shift), which is done by scanning the provider ID. Second, the time needed for NFSDE for key 4-digit manual entry varies from provider to provider, so it is run multiple times with different people and then the average calculated with a total time of about 5 seconds as shown in Table 7.3.

**Table 7. 3 Time for NFSDE unlock**

<b>Event</b>	<b>Time</b>
Scan the ID Card (D)	1 second, see [283]
Insert and Unlock the SD	2 seconds
Enter the PIN	2 seconds
Total time to Unlock	5 seconds

#### **Time to read the patient data**

To calculate the patient data reading by the NFSDE:

1. Scan the patient ID card, then 2) scan the patient fingerprint; 3) calculate the time for the decryption for the 4K data on the patient tag. Table 7.4 shows the decryption

using MICKEY 2.0 is a very short time in microseconds. The total time is 2.07 seconds to show the encrypted data.

**Table 7. 4 Time to read encrypted patient data**

<b>Event</b>	<b>Time</b>
Scan the ID card (P)	1 second, see [283]
Scan the fingerprint	1 second, see [284]
Decode 4K data	66,291 microseconds
Total time to read the patient data	2 seconds

Other than decryption time which is easily measured, the time spent by people varies, so by taking the average for multiple entries a relatively good estimation can be made. However, as seen in Table 7.3 and Table 7.4, the whole process only takes a few seconds – around 5 seconds or less for NFSDE unlocking, and 2 seconds or less for reading patient data.

## **7.10 Attacks analysis**

This section explains how the system is resistant against possible attacks to test protocol security.

In this eHealth scenario, MICKEY 2.0 was used as an example of a possible lightweight encryption method. MICKEY 2.0 is replaceable, by any lightweight ciphers, as well as the thesis proposed cipher MICKEY 2.0.85, to encrypt the values (parameters) on the provider and patient tags.

The parameters can be encrypted as following:

- a) Patients' medical data
- b) Patients' IDs
- c) Patients' fingerprints
- d) The record keeper authentication located on the patients' tags
- e) The providers' authorisation level and identity
- f) The record keeper authentication is located on the providers' tags.

These parameters need to be addressed for any kind of cryptanalysis and attack resistance. To encrypt these parameters  $IV_0$ ,  $K_0$  and  $K_1$  need to be secreted to ensure the system security.  $IV_0$  value is provided by the record keeper, and the other non-secret  $IV_S$  ( $IV_P$  for patient and  $IV_d$  for provider) with varying keys. Revealing one of the secret  $IV_S$  or different keys will not result in recovering the other values.

#### **IVs calculations:**

$IV_{rd}$  used to encrypt and decrypt the provider authentication and

$$IV_{rd} = h(IV_0, \text{provider' tag UID}), h = \text{hash function}$$

$IV_{rp}$  used to encrypt and decrypt the patient authentication and

$$IV_{rp} = h(IV_0, \text{patient tag UID}), \text{ as the fingerprint data in the patient tag}$$

$IV_d$  used to encrypt and decrypt the provider level of authorisation and

$$IV_d = h(\text{PIN}, \text{provider tag UID})$$

$IV_p$  used to encrypt and decrypt the patient medical data and

$$IV_p = h(\text{patient fingerprint reference}, \text{Patient Tag UID})$$

#### **Keys calculation:**

$K_0$  used with  $IV_d$  for encryption and decryption of provider authorisation level and identity, also  $K_0$  used with  $IV_p$  for encryption and decryption of patient medical information.

$K_1$  used with  $IV_{rd}$  for encryption and decryption of provider authentication, and also  $K_1$  used with  $IV_{rp}$  for encryption and decryption of a patient's fingerprint and authentication.

### **7.10.1 Known plaintext attacks**

When an attacker gets information about the plaintext (P) (or portion of it) and the corresponding ciphertext (C) for a given message, this is called a known plaintext attack [12], [16]. Then the attacker will attempt to reveal the key based on the calculation of the mapping F between the P and C:

$$F:P \rightarrow C.$$

In the security protocol and the NFSDE device design it is not feasible for any attacker to perform this kind of attack, as for example, a patient's tag containing the biometric associated with the keystream is unique for every patient, so revealing one patient's data will not result in revealing another patient or provider's keystream. Nevertheless a patient

may compute their own  $IV_p$ . However, using MICKEY 2.0 as the encryption method will prevent the secret key recovery from known IV/plaintext [168], [248], and that is valid for all different combinations. In addition, plaintext used for field authentications will not be known as all parameters are encrypted by MICKEY 2.0 using hash functions, and not the original values. Furthermore, if the fingerprint data (which uses the ISO standard) for any two scans is not identical, the fingerprint will not be revealed as it is not possible to recover keys and IVs by using the known plaintext.

### **7.10.2 Brute force attack**

A brute force attack which is conducted by running extremely large computations in order to reveal the secret key can be considered feasible if it can be achieved in a reasonable amount of time [12]. In the NFSDE device and the developed security protocol, this kind of attack requires implementing a trial and error technique, in order for an attacker to guess the keys and IVs. For the secret parameters  $IV_0$ ,  $K_0$  and  $K_1$  already stored in the encrypted USB, performing this attack needs to mount the USB by computer to simulate the stored encrypted data. This also assumes having an authorised tag and legitimate scan of the fingerprint. Having said that, the attacker will be required to produce the 80-bit of ( $IV_0$ ,  $K_0$  and  $K_1$ ) parameters correctly, thus is  $2^{240}$  bits. Assuming this was done successfully (which is not feasible), it will only reveal a single KTI value. Thus, this kind of attack is not reasonable from a computational perspective.

### **7.10.3 Chosen IV attack**

The attacker tries to find some flaws in the IV, to gain some information about the secret key. The complexity of this attack is about how many bits are needed to extract the key [285]. For the stream cipher the IV is initialised with a secret key in the cipher which works as a function to generate a pseudo-random keystream. It is not practical to reuse the same IV with key as it is unsafe practice, and the lightweight stream ciphers may be vulnerable to such an attack. To avoid this vulnerability, choice of the IV is not allowed. There are four separate IVs, and all the IVs were computed using the hash functions.  $IV_d$  and  $IV_p$  are the unique values of the ID assigned at the time of the production process by the RFID manufacturer. On the other hand, the values of  $IV_{tp}$  and  $IV_{rd}$  as well as the  $IV_0$  (secret value) are all part of the hashing process. Thus it is not possible to select an IV value that will result in an attack due to using the hash signature as a data integrity tool.

#### **7.10.4 Two-time pad/reused key**

A two time attack can also be defined as a reused key attack [286]. The attack occurs when the attacker can get two different ciphertexts which were encrypted using the same key. Assume  $C_1$  and  $C_2$  the ciphertexts for  $m_1$  and  $m_2$ (messages), and  $P_1$  &  $P_2$  the plaintexts, then the attacker calculates  $C_1 \text{ XOR } m_1 = P_1$  and  $C_2 \text{ XOR } m_2 = P_2$ , and  $K$  is the same key used for both  $m_1$  and  $m_2$ , then  $K$  can be obtained by applying analysis of frequency.

The proposed system has four different (keys, IVs) pairs for five different steps. In addition, by implementing the keys rotation using timed KTI, the two IVs for the patient's tag and the two IVs for the provider's tag, each IV is obtained by two (at least) authentication factors as following:

1. RFID tag + PIN for the provider
2. RFID tag + Biometric (fingerprint) for the patient.

Additionally, three out of five encrypted parameters cannot be calculated by the frequency analysis as they were encrypted with hash functions. To be more specific the encrypted fingerprint on the patient's tag cannot be subject to frequency analysis, as the reused key cannot be used for encrypted parameters with a hash function.

#### **7.10.5 Denial-of-service attack**

In a denial of service attack the attacker tries to prevent or interrupt the system from doing the usual task [243], such as by sending a large number of requests to overwhelm the system and make it freeze or crash. The attacker can make the system slow in response for authorised users. For this system, the attacker needs to be physically close to the system as it assumes the internet connection is not present. The attacker could use nearby physical sources to overwhelm the service by electromagnetic interference, or use any method to corrupt or damage the device. In denial of service attacks, the system security information was located on, for example a USB, so it is assumed policies are implemented on use of the device components.

### **7.10.6 Insider attack**

In an insider attack, the attacker is an authorised person who has an access privilege to the system, and can use the system in unauthorised or malicious activities [287]. Insiders in the system include the provider and the record keeper. To secure all activities such as login and authorisation, the login is a timed process and is not reproducible. By applying the three As of data security, the insiders cannot deny their activities. Tag creation and key management and distribution tasks rely on the record keeper who has the same non-reproducible login activities. Furthermore, the key rotation and the key storage update the USB, and it also uses a cloud external service such as AWS for the key update and storage to update the USB.

### **7.10.7 Impersonation attack**

An impersonation attack occurs when the attacker successfully guesses the authorised user authentication which allows the attacker to gain access to confidential information [288]. The proposed system contains an authorisation process for the three entities: record keeper, provider and patient. Key and tag creation and management is the responsibility of the record keeper, and there should be a role and policies in place to ensure the security guidelines provided by the organisation are followed. Thus, multi-factor authentication procedures assure resistance against this kind of attack, as the provider tag UID and PIN, and the patient fingerprint with tag UID are authentication elements, and the record keeper should follow the organisation's security guidelines.

### **7.10.8 Man in the middle attack**

A man in the middle attack occurs when the attacker can interrupt the message between the sender and the receiver, and gain some secret information about the encryption keys [289]. In the system the encrypted USB is a physical object to store the keys. When the record keeper shares the USB with the provider, the record keeper must follow the organisation's administrative procedures, and physical delivery methods such as locks or robust boxes should be used. The USB storage update provided by a cloud service, with key delivery in a secure manner, can use AWS secret manager. Furthermore, a man in the middle attack on the USB is not feasible as it is not connected to other NFSDE physical components.

### 7.10.9 Side channel attacks

In general, side channel attacks work by capitalising on the flaws in the system hardware or software features, not on the system encryption method, in order to extract some secret information [290]. There are two major types of side channel attacks: differential power analysis and challenge/response attack.

- A. A differential power analysis is a popular kind of side channel attack which targets the embedded system like the NFSDE [291]. The aim is to recover the secret keys by applying the statistical analysis of the device power. If the keys are stored in the device, this will make it more vulnerable to DPA attack, therefore, it is possible to find the key by gaining access to the device memory (or portion of it) which contains the keys. The NFSDE device needs to be physically close to the resources of the DPA for a long period of time, and also requires a large sample of data in order for a DPA attack to be sufficient. This is avoided in NFSDE by using the external encrypted USB as a secret keys storage tool, as well as updating and rotating the keys on a regular basis to avoid reusing the same set of keys.
  
- B. Challenge/response side channel attack can be more applicable to RFID-based systems explained in detail in [292]. It is also a kind of statistical analysis that measures the correlation of the power analysis in the RFID domain to categories in the challenge/response protocol in the RFID system. The challenge/response protocol is not used in the NFSDE device and the security protocol. As the keys are not shared via RFID tags, rather by a physical device which is attached to the system, and the hash functions are used in the authentication instead of the challenge/response method, this kind of attack is not applicable in the proposed system.

## **7.11 Overall analysis and discussion**

Earlier IoT and RFID enthusiasts have discovered that low-power devices have insufficient security [222]. This chapter suggests a low-cost, standard-based security solution that remains practical in the worst case situations without internet connectivity to low-power secure devices.

The approach involves key management and key-update solutions for a protected RFID reader which is neither based on the internet nor on wireless connectivity, as described below. Specifically, an example of a real-life framework for patients in locations lacking secure connectivity (including Wi-Fi, 4G or 5G networks), in emergency circumstances and in rural regions is suggested and emulated. The service can include access to patient health information under challenging conditions. The suggested cryptosystem offers a low-cost, efficient and security protection alternative to the CIA triad usually offered by the public key infrastructure. By taking a completely new strategy and using applications and off-the-shelf equipment, a more complex custom-hardware asymmetric solution is avoided as in [231], [232].

### **7.11.1 Providing multi-factor authentication without connectivity**

The proposed security solution is a mix of “what you do” (i.e., a PIN and an USB passcode), “what you have” (i.e. an RFID tag and an USB) and “what you are” (i.e., biometrics such as fingerprint). For conventional key control, the whole key is placed on “anything you have” (i.e. a mutual key storage drive) or is “anything you remember” (password or PIN). Once the keys are reset, the keys must be redirected and kept, or the password or the PIN must be entered. The key and the PIN must be protected against interruption. Using NFSDEs without using public-key encryption, this weakness is minimised by using USB.

The USB will not include the full collection of necessary keys, but rather includes the parameters which use it to determine the keys. Such criteria are protected by a passcode that can be exclusive to every USB. It is wise to request healthcare providers to take proper diligence to secure the USB or to change it on a calendar basis. The modification of the USB security parameters is a question of routine practices rather than a technological problem.

The USB may be modified by transferring it to a protected location or using a protected method such as the AWS Secrets Manager. A debit-card similar delivery scheme may be used where the USB is physically distributed in one box by a courier and the passcode can be provided remotely through a program such as “One Time Secret passcode” [293].

In any scenario, the administration is no more complicated than any other main delivery method. When accessing private records, a minimum of six variables must be jointly authenticated. These three forms of variables are used at least once: anything you have – the patient’s RFID card, the provider’s RFID card and the USB, anything you know – the provider’s PIN and the USB card password, and anything you are – the patient’s fingerprint. Decryption keys are not usable until all variables are identified and collectively validated. Even if the USB is corrupted or exposed, exposure to private data is not feasible without four external security parameters. Thus, using the USB to store the security parameters is more reliable than the standard key transfer and no more difficult than using a debit card with a new PIN.

### **7.11.2 Providing privacy by design**

The fourth theory of “privacy by design” [265] is that secrecy ought to be “fundamental to the framework, without reducing functionality”. Rapid access to essential data is a practical necessity in emergency circumstances. Thus, a framework was built and mimicked in which verification, access and accounting would be no more difficult than the withdrawal of cash from an ATM. This thesis has been able to minimise or eradicate documented cryptographic attacks by implementing the concepts of defence in depth [261]. Through using a lightweight cryptosystem and a low-cost, efficient key delivery method, the framework proved able to deliver the CIA triad in the worst-case scenario of life and death.

### **7.11.3 Providing key distribution without connectivity**

Public key encryption is a growing method for key distribution. Generally, public key encryption is being used for security, secrecy and integrity; this includes PKI (usually communication needed for access to the Certifying Authority) and substantial CPU power. They are still not needed in this situation. Therefore authentication, anonymity and integrity are achieved without public key infrastructure by using the MICKEY 2.0 stream

encryption, hardware-based protection, and a security protocol with specific procedures. The innovative solution proposed for this is to measure each of four IVs and keys pairs using a mix of hardware-secure values, mutual values, and embedded RFID tags.

In standard symmetric cryptography, the sender must build a key and pass it to the recipient. The number of keys needed is considered to be  $n(n - 1)/2$ , where  $n$  is the number of parties that need to be communicated. The possibility of a key being captured by a third party and the need to establish a specific key for each sender/receiver pair is recognised as the “key delivery issue” [294].

Take the following scenario. If Alice, Bob, Charlie and Dave decide to share protected texts, six keys will have to be produced. Every key should be transferred from the sender to the recipient such that the adversary, say Eve, will not have the ability to replicate or capture the key. The key-distribution question to be solved is fairly straightforward:

- 1) How would Alice establish exclusive keys for Bob, Charlie, and Dave?
- 2) How would she send the keys to Bob, Charlie, and Dave without having adversary Eve intercept them?

A solution is proposed for the key distribution issue, where separately encrypted off-the-shelf hardware systems are used to store the security parameters that are used to determine the mutual symmetric security for each sender/receiver pair [261]. The threat review by analysing the possible attacks found that the same approach had mitigated specific RFID-based attacks. As the system is explicitly configured not to be wired or connected to the internet, it may also be used for out-of-band authentication (and probably authorisation).

The aim of the lightweight cryptosystem including the NFSDE prototype with security protocol is to encourage innovations in eHealth technology. Therefore, the minimal-power, strong-quality protection suggested is specifically relevant to RFID protection as well as security in general.

#### **7.11.4 Other applications**

This subsection explores several potential possibilities for RFID security applications with a brief summary of each.

##### **Two-person rule**

In high-security contexts, for example, when approving major organisational spending, it is always important for two workers to approve action [295]. This proposed security system would facilitate the multi-factor “offline” to implement the two-person law.

##### **Human courier scenario**

Because the proposed prototype does not need connection to the internet, it is an optimal solution to protect data from hackers. This is especially important where several individuals need to approve access to the asset. Parties needing very safe contact usually use human couriers rather than, for example, efficient cloud-based sharing of information. If the parties involved do not want to share confidential data through the internet, the cloud can also be a secure key exchange tool [294]. NFSDE, with a secure protocol, can be implemented as out-of-band authentication.

Other possible applications involve multi-factor authentication, such as smart wallets, cold storage of blockchain keys, master encryption keys, and safe data transfer by military organisations, where control of physical access to systems and equipment is required.

##### **Other common applications**

The most important use of RFID tags is to monitor items in transit and to compile inventory. Encrypted data can be placed on the monitoring sticker. The proposed secure system is immune to breaching and side channel breach. No unencrypted data has to be revealed. RFID tags are common for animal monitoring [296]. The approach can be used by scientists to easily store the data on tags that are attached to animal bodies for tracking and monitoring.

## 7.12 Conclusion

The chapter presented a practical application for lightweight encryption with a prototype device to secure sensitive data when internet connection is absent or not reliable. Previous chapters identified the importance of practical and efficient lightweight synchronous stream cipher implementations, that were achieved by introducing tools for security evaluation, cipher optimisation and practical application in mobile cloud computing and RFID technology.

Considering advances and expansion of RFID technology, as well as new applications, security remains a challenging task that needs to be addressed and improved. The proposed security system including NFSDE device and the security protocol fills the gap in this important area.

The approach bypasses the current solutions which focus on hardware solutions. However, the solution can be worked as a framework and software solution, which is flexible in regard to hardware, software and lightweight encryption, as they can be modified and tailored according to the user's needs without compromising the security. The NFSDE device components can be replaced, and the MICKEY 2.0 cipher can be replaced with other lightweight ciphers. Using eHealth as a practical example of efficient application, the lightweight security system solved and tested the tags' and parties' identification, authorisation and confidentiality. The following chapter provides an overall discussion of the thesis achievements with implications for advancing the field of lightweight synchronous stream ciphers.

## **Chapter 8: Discussion and Conclusion**

### **8.0 Chapter overview**

This chapter presents the discussion and overall analysis of the thesis findings and contributions, as well as possible future research directions emerging from this work. This chapter has the following structure. Section introduces the chapter, Section 8.2 reviews the thesis rationale and summarises outcomes; Section 8.3 discusses the unique window size and d-monomial tests; Section 8.4 discusses the proposed neural network models; Section 8.5 discusses the proposed MICKEY 2.0.85 cipher; Section 8.6 discusses the proposed FEATHER lightweight security protocol; Section 8.7 discusses the proposed lightweight cryptosystem with NFSDE prototype device; Section 8.8 shows how the thesis findings contribute to the current literature; and Section 8.9 summarises the thesis contributions. Finally, Section 8.10 presents possible future research directions resulting from this work to further improve security.

### **8.1 Introduction**

The thesis focuses on lightweight synchronous stream cipher analysis and applications. Data analysis, including statistical and mathematical analysis, is essential to work as a cryptanalysis method for lightweight synchronous stream ciphers, using pseudo-randomness statistical tests such as ANF-based tests, which translate the binary sequence that comes from the cipher and work as a keystream to translate their algebraic normal form, and then apply d-monomial based tests. Statistical pseudo-randomness tests, such as the suite of NIST tests, combine mathematical concepts with statistical analysis to assess the strength of a cipher. These tests help to optimise existing ciphers and develop new techniques and security implementations in essential applications. Pseudo-random number generators are essential for generating a pseudo-random number binary sequence as a keystream to ensure that a sequence has a pseudo-random appearance and does not cause any biases that may leave them vulnerable to attacks.

It is important, therefore, to test randomness using multiple different randomness tests, including ANF-based tests (e.g. d-monomial test), UWS tests, and prediction models such as multilinear regression predicting model, as well as superior prediction models, such as

neural network models. The standard NIST randomness tests provide a standard evaluation for the sequences generated by pseudo-random number generators. In addition, cryptanalysis methods apply different known attacks to test the resistance of ciphers to such attacks. Using different cryptanalysis and pseudo-randomness tests and adapting them for each particular cipher is not straightforward. The optimisation of lightweight encryption methods is currently achieved using lightweight synchronous stream ciphers, by proposing lighter, faster and power-efficient ciphers, which are feasible for lighter security cryptosystems, including smaller devices. Implementing lightweight encryption in real-world applications, such as mobile cloud computing, will improve IoT security in general, and will also be useful in RFID technology. These technologies both have important applications in areas that require a high level of security, such as eHealth care.

## **8.2 Discussion of thesis rationale and overview of outcomes**

This thesis adapts randomness analysis for chosen lightweight stream ciphers in order to introduce new cryptanalysis methods, including an optimised version of MICKEY 2.0 called MICKEY 2.0.85 and a secure lightweight protocol for mobile cloud computing called FEATHER. The thesis also provides a lightweight cryptosystem based on RFID systems suitable for authentication and security without needing an internet connection. Research findings were published in [221], [123], [297], [23].

The thesis tests and proves the following hypotheses:

1. Adapting pseudo-randomness tests is possible, regardless of the implementation effort of adaptation, and the tests can be tailored to specific ciphers.
2. Prediction modelling, especially neural network models, is effective for testing and measuring the pseudo-randomness of a binary sequence, which is the most important element of cipher security.
3. Optimising a successful, secure, and popular lightweight stream cipher is possible, and it is possible to have a secure lighter version.
4. Mobile cloud computing can benefit from lightweight stream ciphers. Lightweight encryption can provide security, faster performance, and longer battery life.
5. It is possible to provide a lightweight cryptosystem based on RFID technology, with no internet connection.

### **8.3 Using unique window size and d-monomial tests as randomness tests**

The unique window size test (UWS) is based on the maximum order complexity test, as discussed in Chapter 3. The goal of this test is to determine if the keystream is random enough for it to be very hard to find a function that can generate a similar keystream binary sequence. The SG and SSG lightweight synchronous stream ciphers were tested using this test to determine if they were the proper result of pseudo-random numbers generators. In this test, the bigger the UWS the better and more secure the cipher. If a cipher has a small UWS, it means for SG the two LFSRs combinations may need to be modified to have a larger UWS, and for SSG it implies the LFSR primitive polynomial needs to be changed. The results of this test provide users with a better choice of the LFSRs to ensure that the ciphers have sufficient pseudo-randomness properties.

The d-monomial tests are used to detect monomials of a certain degree  $d$ . If these monomials follow a normal distribution, it implies that there are no biases in the keystream. Another similar test is the maximal monomial test, which finds the highest  $d$  in the keystream. These tests are pseudo-randomness tests. The d-monomial tests on both the SG and SSG ciphers found that SG was weaker than SSG. A multilinear regression model was established for SG with degree 20 for UWS prediction. These results will help users to choose the SG LFSRs combination that results in less predictability.

### **8.4 Developing proposed novel neural network-based prediction models**

The neural network prediction model in Chapter 4 was designed to predict UWS for both SG and SSG. UWS as a pseudo-randomness tool is an important valid indicator of a cipher's security.

For the SG cipher, four neural network prediction UWS models were established for degree 20, 21, 23 and 24, with a learning rate of 0.0001 for degrees 20, 22 and 24, and a learning rate of 0.001 for degree 21. The learning rate needed to be adjusted for different degrees of the model to optimise learning for more accurate predictions; accuracy was approximately 95% and mean squared errors (MSE) of  $MSE < 0.008$  and  $MSE = 0.0019$ . UWS24 has the largest dataset, and hence, the model is better able to learn.

For SSG, there were five neural network predictions, with one for each UWS model of degrees 21, 22, 23, 24 and 25, with a learning rate of 0.0001. The models have four hidden layers, with 100, 50, 20 and 10 nodes, respectively, with an accuracy of 96.66%, 89.61%, 90.14%, 97.01%, and 97.14%, respectively, and an MSE of 0.0014, 0.016, 0.0046, 0.0098 and 0.0052, respectively. These are low error rates and indicate high model accuracy. In addition, to show how the model is able to learn with multiple UWS in the same model, one model was established for degrees 4 to 20 as one dataset, with a learning rate of 0.0001, and four hidden layers, with 100, 50, 20 and 10 nodes, respectively. For this model, the MSE = 0.0012, which is very low, and the accuracy was 96.05%.

Although the d-monomial and UWS tests established that the SG is weaker than SSG, the neural network prediction models showed high predictability for both ciphers, which introduced a new measurement tool for randomness and nonlinear complexity measurement. It was also observed that the models were able to learn better with larger datasets.

## **8.5 Developing and testing the proposed MICKEY 2.0.85 cipher**

MICKEY 2.0.85 was achieved by reduction and testing for pseudo-randomness after multiple experiments. Thus, the standard pseudo-randomness tests are necessary to ensure the viability of the new version. The suite of NIST tests is a standard measurement for randomness and uses a large number of binary sequences generated by MICKEY 2.0.85 for both the keystream and ciphertext. This cipher was compared to MICKEY 2.0 and MICKEY 1.0 to establish whether it has better pseudo-randomness. As expected, MICKEY 1.0, the older version, failed 14 of the 15 tests and only passed the linear complexity test. The stronger version, MICKEY 2.0, appeared to have a good passing rate on the NIST tests. The lighter version MICKEY 2.0.85 has a slightly better NIST test passing rate than MICKEY 2.0, as shown in Chapter 5.

MICKEY 2.0.85 was achieved by reduction and testing for pseudo-randomness after multiple experiments. Thus, the standard pseudo-randomness tests are necessary to ensure the viability of the new version. The NIST tests suite, as a standard measurement for randomness, uses a large number of binary sequences, which are generated by MICKEY 2.0.85 for both the keystream and the ciphertext. This cipher was compared to MICKEY 2.0 and MICKEY 1.0 to establish whether it has better pseudo randomness. It has

previously been established that this is a weak cipher and the NIST test confirmed this conclusion. The stronger version, MICKEY 2.0, appeared to have a good pass rate on the NIST tests, but the lighter version, MICKEY 2.0.85, has a slightly higher NIST test passing rate, as shown in Chapter 5.

The reduction methods for the MICKEY 2.0 cipher were carried out to maintain the randomness as much as possible by reducing the internal state bits number to achieve fewer gate equivalents. MICKEY 2.0.85 had 12.45% fewer gate equivalents than MICKEY 2.0. The reduction of gate equivalents will improve overall speed performance and reduce power consumption.

The power consumption estimator XPE, used for MICKEY 2.0.85, MICKEY 2.0, Trivium and Micro-Trivium, showed that MICKEY 2.0.85 had the lowest power consumption, with a reduction of 16.202% compared to MICKEY 2.0. The relationship between the number of gate equivalents and power consumption was described in two equations in Chapter 5.

The MICKEY 2.0.85 encryption speed is about 23% faster than MICKEY 2.0, which is important for computation in small devices, as more texts can be encrypted in less time and it consumes less power, which reduces overall costs in applications. The cryptanalysis in Chapter 5 showed that MICKEY 2.0.85 is slightly more attack resistant than MICKEY 2.0.

## **8.6 Developing new lightweight encryption method: FEATHER lightweight security protocol**

The FEATHER protocol was designed to be a lightweight encryption method for mobile cloud computing security. It secures communications between mobile devices, and between mobiles and a cloud server. It has many security parameters:

1. MICKEY 2.0 cipher – as a lightweight encryption/decryption method
2. Hash function – for securing security parameters
3. Timestamps – for time authentications
4. One time pad password – for mobile users to start the communication and be sent out of band
5. Secret keys and IVs pairs changed for every time used
6. Token with expiry time – for the server to verify the mobile

7. File ID – is unique for every file and used for file verifications
8. Mobile phone unique ID
9. Encrypted keystream – for keystream encryption in the cloud server
10. Username – unique to every user.

FEATHER is particularly secure for mobile users because of its series of steps. The process for communication by mobile users to request the keystream from the external server involves highly secure steps:

1. Register: The person registers an account on the external server by sending a message containing UID, phone ID and timestamp.
2. Update: The external server confirms the validity of the message by recomputing the signature, and then decrypts and stores the hashed password in the account. The response is either OK or ERROR.
3. Validate: The external server decodes the hashed password, recomputes the signature, and responds with OK or ERROR.
4. Generate: The external server generates a random MICKEY 2.0 key (20 bytes of Key+IV).
5. Upload: The external server stores the file and responds with OK or else ERROR if something goes wrong.
6. Request: The external server uses the token (or file-ID) to look up the requested data and sends it back to the mobile device.

Using five mobile devices to check the efficiency of the protocol, the cumulative time for keystream downloading, decoding and typing to memory was calculated. The FEATHER downloading is often time-consuming. However, compared to the CLOAK protocol [214], FEATHER is much faster. For example, for a file size of 8 MB, the total download time is 110 seconds using CLOAK and around 9.8 seconds using FEATHER. If more than two mobile devices need to interact at the same time, the external server generating the keystream using the FEATHER protocol is far faster than when using CLOAK. FEATHER would reduce the overall time because the decoding period is only running XOR operation on the keystream, which is fairly easy. Smartphone battery life is higher with FEATHER. This lightweight authentication mechanism can help to maintain anonymity, authorisation and protection for consumers. It also aims to reduce the power usage of smartphone devices to boost overall efficiency. The suggested lightweight protocol FEATHER was tested against possible established attacks and found to be secure and effective for

implementation. As MICKEY 2.0 was used as a pseudo-random number generator, the protocol is adaptable for use in other IV-based lightweight synchronous stream ciphers. Combined with the new MICKEY 2.0.85, presented in Chapter 5, it would be 23% faster to produce the secure keystream. This work is a contribution to advance mobile cloud computer security, technologies, IoT development, and security in general.

## **8.7 Developing proposed lightweight cryptosystem with NFSDE prototype device**

NFSDE and the secure protocol is a lightweight cryptosystem that proposes a prototype of a device called near-field secure data extractor, which allows secure RFID communications using a lightweight communication protocol without requiring stable internet connectivity. It offers extremely secure data exchange in the absence of the internet, which makes it useful in places with poor or no internet connectivity, such as remote and disaster-struck areas. The device demonstrates fast processing as well as robustness against several forms of attacks. An application of the proposed device and information exchange system is explained in the context of an eHealth scenario.

The proposed cryptosystem is a significant contribution to the literature because, unlike conventional solutions, the proposed solution is inexpensive and can be easily customised to various application scenarios because it uses standard commercially available components. This is a contribution to improving RFID-based security because it proposes a simple yet highly efficient solution for the storage and exchange of protected data in a secure manner without depending on internet connectivity. The proposed device has applications in multiple scenarios, such as:

- Two-person rule: In strong-security situations, for instance, when authorising major organisational spending, it is often necessary for two employees to authorise action.

This new protection framework would make it simpler for the multi-factor “offline” to enforce the two-person rule.

- Human courier scenario: As the current system will not need to be wired to the internet, it is an ideal way to secure data from hackers. If the parties involved do not wish to transmit sensitive data over the internet, the cloud may also be a protected key exchange tool.

Parties needing safe communication typically use human couriers rather than, for example, effective cloud-based exchange of knowledge.

- Other potential uses include multi-factor authentication, such as smart wallets, cold storage of blockchain keys, master encryption keys, and protected data sharing by military agencies where regulation of physical access to systems and facilities is needed.
- This solution may be used by scientists to conveniently store data on tags connected to animal bodies for tracking and control.
- An important application of RFID tags is to track products in transit and compile inventory.

Thus, the proposed cryptosystem implementing NFSDE devices can improve security in low-power or small-scale projects, where security is often compromised due to technical limitations or costs.

## **8.8 Thesis findings and contribution to the literature**

The thesis made key findings in four main areas, addressing issues in the existing literature. It identified flaws in shrinking generator and self-shrinking generator by proposing neural network-based prediction models for pseudo-randomness, proposed MICKEY 2.0.85 as a secure and lighter version of MICKEY 2.0, proposed a secure lightweight FEATHER protocol for mobile cloud computing security and proposed Near Field Secure Data Extractor (NFSDE) with lightweight secure encryption protocol for RFID security in the absence of internet connectivity.

**Contribution 1:** Identified flaws in shrinking generator and self-shrinking generator by proposing neural network-based prediction models for pseudo-randomness

In a study of randomness tests for binary sequences [298], researchers confirmed that, by calculating the longest period, the probability of obtaining zeros or ones is equal, but generalising this to different periods requires more research. This thesis addressed this issue by presenting a more holistic approach by using prediction models, specifically neural network models, to predict unique window size, which can be applied to any binary sequence [123], [297].

A study [299] on new statistical tests beyond NIST tests proposed using randomised tests for binary sequences based on limited patterns and is not comprehensive. The tests in this thesis provide an effective way to measure randomness to predict unique window size using neural network models, which is more comprehensive than the methods proposed in [299].

Previous studies [300] and [301] have focused on tests of whether the Boolean function was balanced and used a classification of weights with a specific and arranged framework, which is limited to weights classification. Maximum order complexity is preferred as a randomness test to an expansion test. These tests have been applied to the Thue–Morse and Rudin–Shapiro series, and the results confirm that maximum order complexity is a better test [302]. These results confirm the approach advanced in this thesis where unique window size is used as a form of maximum order complexity for randomness tests and neural networks are used for randomness prediction.

In conclusion, the approach advanced in this thesis, of using unique window size and neural network prediction models, provides accurate measurements with a very tiny error margin, and the approach can be generalised to any cipher that generates a binary sequence as a keystream.

**Contribution 2:** Proposed MICKEY 2.0.85 as a secure and lighter version of MICKEY 2.0

The most efficient and most used cipher is AES; however, it is a heavy encryption method, as AES requires relatively high power capability and considerable chip size. The thesis developed the proposed MICKEY 2.0.85 cipher [23] to overcome this problem. In general, non-lightweight ciphers are slower in performance, speed, and lower in throughput [303], [304].

RFID technology uses small devices as its components, especially RFID tags and RFID readers, and applications like sensor networks and IoT technology have small computation processor units, such as Raspberry Pi and Arduino. It is thus essential to provide security for these small components [305-307].

Lightweight ciphers such as Trivium were targeted to design multiple reduced versions, such as Micro-Trivium [189]; however, the proposed cipher, MICKEY 2.0.85, is more power efficient and needs fewer gate equivalents to work. More related optimised versions of different ciphers were introduced in Chapter 5 and also in [23].

Study [205] investigated using AES in mobile cloud computing and showed that the tiny RAM, low power supply, and small processors with small speed in mobile cloud computing meant AES was not feasible. Thus, lightweight encryption was introduced to handle the heavy tasks, such as file offload/download, with encryption methods based on pseudo-random permutation based on chaos systems.

**Contribution 3:** Developed secure lightweight FEATHER protocol for mobile cloud computing

A short study [308] based on cloud computing and mobile computing debated the importance of leaving the offloading tasks to be carried out in external applications using an external server. The authors proposed a mobile cloud computing enterprise that consists of four elements: mobile devices, wireless core, Wi-Fi access point, and regional information centres.

Another study [204] showed that mobile devices could save energy by offloading some tasks to the cloud server, such as battery life and wireless energy, which is used to transfer the data in some applications; however, some applications are not energy saver efficient [298]. In addition to the limited computational power in mobile devices, battery consumption due to heavy computation adds another challenge, which makes mobile cloud computing a good solution. As mobile devices have limited computation power, it is hard to address all security cryptosystem requirements.

CLOAK is a lightweight protocol based on the AES cipher, which enables two mobile devices to communicate with each other while leaving the keystream generation on an external server (AWS in their implementation) [214]. It can be compromised by fetching the keystream from an external server and from communication media as well. On the other hand, it can get the keystream from either trusted or untrusted external servers. However, the FEATHER protocol is a lightweight security cryptosystem and, as shown in Chapter 6, is faster and more power efficient.

**Contribution 4:** Developed Near Field Secure Data Extractor (NFSDE) with lightweight secure encryption protocol for RFID security in the absence of internet connectivity

The low cost of RFID tags means it is desirable for authentication and verification [249]. However, it requires internet connectivity, while the NFSDE cryptosystem published as part of this thesis does not require a connection to the internet [221].

The possibility of the key being intercepted by a third party and the requirement to establish a specific key for each sender–receiver pair is regarded as the “key delivery problem” [294]. In standard symmetric encryption, it is important for the sender to establish and transfer the key to the recipient. The number of keys needed is considered to be  $n(n - 1)/2$ , where  $n$  is the number of entities who have to be informed. Another study [231] favored a hardware approach to this issue and suggested specialised equipment named Recryptor. While asymmetric architectures can solve this issue they need more processing power than is ordinarily available in low-power devices. Another study [232] used a design processor named Fulmine for IoT near-sensor applications. This thesis proposed a cryptosystem implementation that includes NFSDE with MICKEY 2.0 to solve key distribution problems [221].

In situations with poor internet access, attacks by hackers can be much more aggressive [253] because the current framework has to be updated to resolve communication issues and to include authentication approaches that are suited to such scenarios. However, Study [253] offered solutions for open and untrusted networks, while the NFSDE based cryptosystem in this thesis offers protection where internet access is poor or non-existent [221].

## 8.9 Conclusion

In summary, the thesis analysis, randomness testing, neural network modelling, optimised cipher MICKEY 2.0.85, the FEATHER lightweight protocol for securing mobile cloud computing, and the NFSDE based lightweight cryptosystem all contribute to improved cryptographic and security analysis and applications. This work can inspire further security research and related technological developments to provide low-cost security solutions for small devices and small-scale projects. The main contributions of this thesis can be summarised as follows:

1. The thesis research identified flaws in the shrinking generator and self-shrinking generator ciphers. This is useful because it can be adapted for any other ciphers.
2. The thesis research designed a prediction neural network based model for randomness evaluation. This can be used for pseudo-randomness testing of any kind of ciphers and hash functions for example.
3. The thesis research proposed a lighter and more secure version of the MICKEY 2.0 cipher, called MICKEY 2.0.85. This is slightly better than the original in various security aspects and overall performance, and can be used for low scale security projects.
4. The thesis research tested MICKEY 2.0.85 for efficiency and found that it was 23% faster in encryption and consumes less power than MICKEY 2.0. This new version is useful for low-cost encryption methods such as mobile cloud computing and RFID low scale project. In addition, the testing methods work as a framework for designing new encryption methods for evaluation and testing.
5. The thesis research proposed the FEATHER security protocol for mobile cloud computing. This new lightweight cipher can be used for new small devices which are essential components of mobile cloud computing.
6. The thesis research introduced the non-internet connectivity dependent Near Field Secure Data Extractor (NFSDE) prototype device with security encryption method for RFID technology at the absence of internet connectivity. An example of implementation of this system was provided in an eHealth security context. However, it can be used in a wide range of applications such as facilitating the multi-factor “offline” to implement the two-person law, human courier scenario security and to monitor items in transit and to compile inventory.

## 8.10 Possible future research directions

Based on the thesis findings, there are several future research directions to further improve cryptosecurity for real-world applications:

1. Rather than applying the UWS test with neural network models to predict the keystream, it can be applied in internal cipher components, such as LFSR and NLFSR binary output, which can evaluate the strength of the cipher's internal components.
2. Adapting neural network prediction models in different ciphers and hash functions can test their pseudo-randomness and whether the keystreams are complex enough to resist possible attacks.
3. Neural network models for image and face recognition can be implemented by converting the raw data into binary data and then applying the models in Chapter 4, with some modifications. This is an interesting direction, to see how predicted data can result in accurate recognition.
4. The MICKEY 2.0 internal state size can be further reduced to less than 170-bit, but larger than 160-bit (the key size is 80-bit, so the internal state as security rule should be at least  $2 \times 80$ -bit), and then the new version and its parameters can be tested, as in Chapter 5.
5. Another reduction approach in the internal state size of MICKEY 2.0 is to make the S (nonlinear register) size more than R (linear register) size, while keeping the internal state size between 160-bit and 170-bit, and using the evaluation methods from Chapter 5.
6. For FEATHER, new ciphers other than MICKEY 2.0 can be implemented, then the overall process time and the power consumption can be calculated.
7. The cryptosystem (NFSDE + security protocol) can be adopted in other healthcare settings such as home visiting for patients with special needs, where it is difficult for them to visit hospitals and emergency cases, and other settings.
8. A casing can be designed to contain NFSDE components to make it more practical to use in a wide range of different environments (as practical application).
9. A holistic unit can be designed that does the NFSDE work while keeping the functionalities of all internal components.
10. Ciphers other than MICKEY 2.0 can be used in NFSDE for encryption.

## Appendices

The appendices provide additional results to those included in the thesis chapters. The following table of appetencies contents:

Appendices for chapter 3	<b>Appendix 3.1: Additional d-monomial test results</b>
	Appendix 3.2: Linear complexity results for SSG
	Appendix 3.3: Some statistical analysis for UWS4 to UWS24
	Appendix 3.4: Unique Window Size degree vs polynomial weight
	Appendix 3.5: Goodness of fit, plots and figures for different statistical distributions
	Appendix 3.6: Importance of variables based on univariate simple linear regression model R square
Appendices for chapter 5	Appendix 5.1: Counting GEs method
	Appendix 5.2: GE number comparison in MICKEY family algorithms
Appendix for chapter 3	Appendix 6: Example of data that represents the communication operation between mobile device and the cloud server
Appendices for chapter 7	Appendix 7.1: Detailed flow of the patient tag creation process
	Appendix 7.2: NFSDE device emulation
	Appendix 7.3: PUF and possibility of use within NFSDE

# Appendices for Chapter 3

## Appendix 3.1: Additional d-monomial test results

### First: For SSG

We performed d-monomial tests and applied chi-square tests with a degree of freedom of non primitive polynomials of degree 5 to 7.

The following summary table shows the results of the d-monomial test and chi-square test on primitive polynomials from degree 5 to 7. This thesis applied the d-monomial test and the chi-square test, with a degree of freedom n and confidence level  $\alpha=1\%$ ,  $5\%$  and  $10\%$ . (From the null hypothesis, all monomials have normal binomial distribution).

Table 3.1.1 Polynomial for LFSR with degrees from 5 to 7 and their strength

POLY	#TTF	#TFF	#FFF	TOTAL
$x^5+x^2+1$	0	1	0	1
$x^5+x^3+1$	1	0	0	1
$x^5+x^3+x^2+x+1$	1	0	0	1
$x^6+x+1$	0	1	0	1
$x^6+x^5+1$	0	2	0	2
$x^6+x^5+x^2+x+1$	0	0	1	1
$x^6+x^5+x^3+x^2+1$	0	2	0	2
$x^7+x+1$	0	1	1	2
$x^7+x^3+1$	2	0	1	3
$x^7+x^3+x^2+x+1$	0	0	1	1
$x^7+x^5+x^2+x+1$	0	0	1	1
$x^7+x^5+x^3+x+1$	0	0	1	1
$x^7+x^5+x^4+x^3+1$	1	0	0	1
$x^7+x^6+x^4+x^2+1$	1	0	0	1
$x^7+x^6+x^5+x^2+1$	0	0	1	1
$x^7+x^6+x^5+x^3+x^2+x+1$	0	0	1	1
$x^7+x^6+x^5+x^4+1$	1	0	1	2
$x^7+x^6+x^5+x^4+x^2+x+1$	1	0	1	2

T=true, F=false, so TFF (For example) shows the number of initial states for which the d-monomial tests passes at a significant level of 10% and fails at levels of 5% and 1%.

From the above table, we can see that  $x^7+x^3+1$  is a bad polynomial to use with the LFSR for SSG.

Here another example for degree 12. The study tested 144 primitive polynomials of degree 12. The following table shows, in decreasing order, the worst polynomials for LFSR.

Table 3.1.2: Polynomial for LFSR with degree 12 and their strength

Order	Poly	#FFF	#TFF	#TTF	Total
1	$x^{12}+x^{10}+x^2+x+1$	6	5	6	17
2	$x^{12}+x^6+x^5+x^3+1$	4	1	11	16
3	$x^{12}+x^8+x^2+x+1$	3	4	8	15
4	$x^{12}+x^7+x^4+x^3+1$	2	1	10	13
5	$x^{12}+x^{11}+x^{10}+x^8+x^2+x+1$	3	2	6	11
6	$x^{12}+x^9+x^7+x^6+1$	3	1	7	11
7	$x^{12}+x^7+x^6+x^4+1$	2	2	7	11
8	$x^{12}+x^{11}+x^{10}+x^2+1$	3	1	6	10
9	$x^{12}+x^8+x^7+x^2+1$	2	1	7	10

### Second: For SG results

By investigating the sg output and applied the d-monomial and chi-square test on the primitive polynomials, with degrees of freedom of  $n$  and  $n+1$ .

As our primary result, we investigated primitive polynomials for LFSR<sub>1</sub> and LFSR<sub>2</sub> of degree 4 to 7. In this investigation, we used three cases:

1. Fixing LFSR<sub>1</sub> and LFSR<sub>2</sub>.
2. Fixing the LFSR<sub>1</sub> and varying the LFSR<sub>2</sub>.
3. Fixing LFSR<sub>2</sub> and varying LFSR<sub>1</sub>.

From the results of our investigation, sg does not pass our d-monomial tests and chi-square test.

Our primary observation suggests that LFSR<sub>2</sub> (the controlling function) insures the nonlinearity of outputs. We also found that the greatest possible LFSR<sub>2</sub> length is most useful in order to gain the nonlinear property of SG outputs.

## Appendix 3.2: Linear complexity results for SSG

### An investigation of the association between degree and LC (Linear Complexity)

#### 3.2.1 Introduction

This report investigates the association between degree and LC, and if such an association exists, we attempt to find a formula relating degree and LC.

#### 3.2.2 Methodology

Firstly the LC data was grouped using the following coding, for both the chunks and profile data:

*Table 3.2.1: Coding of LC values*

LC value	Code
[0,10]	1
(10,20]	2
(20,30]	3
>30	4

This is required in order to carry out a Chi-square test for association. A Chi-square test for association is then carried out for both the chunks and profile data, individually. If an association is found, we will attempt to quantify the relationship between degree and association using a simple linear regression.

#### 3.2.3 Results

##### Results for the chunks data

Firstly, examine a contingency table.

*Table 3.2.2: Contingency table of degree versus grouped LC value*

		LC grouped			
		[0,10]	(10,20]	(20,30]	>30
Degree	4	32	0	0	0
	5	288	0	0	0
	6	685	83	0	0
	7	4165	1149	366	80
	8	7032	2441	669	2146

The Chi-square test for association is highly significant,  $\chi^2(12)=1460.034$ ,  $p<0.001$ . therefore conclude there is an association between degree and LC. From Table 3.1.1, it is clear that most polynomials have a high degree and low LC (e.g. degree 8 and LC between 0 and 10 inclusive), and very few have low degree and high LC (e.g. degree 4 and LC above 30).

To determine the nature of the relationship between the variables, we examine the scatter plot and fit a simple linear regression.

Please note that in general there is a lower known bound on LC, of the form:  $LC \geq c \cdot e^{(\theta n)}$ , (c and  $\theta$  are positive constants  $\in (0,1)$ ), but this bound is not achieved for all polynomials, which led us to consider this question. Our goal is to see what bound holds “an average” for all polynomials.

Figure 3.2.1: Scatter Plot of LC versus degree

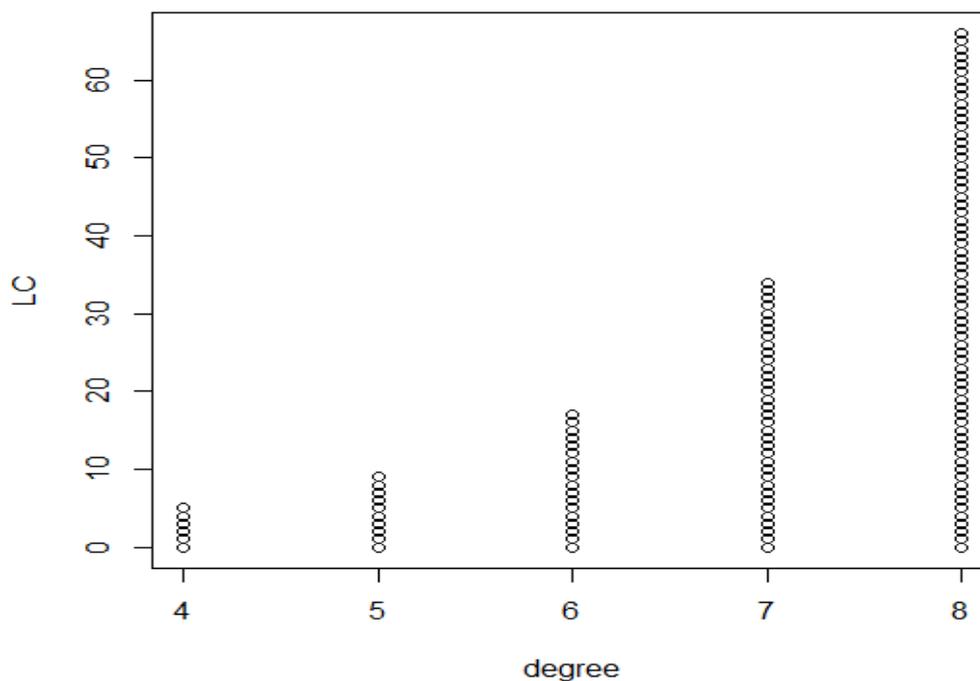


Figure 3.2.2: Scatter Plot of logarithm of LC+1 versus degree

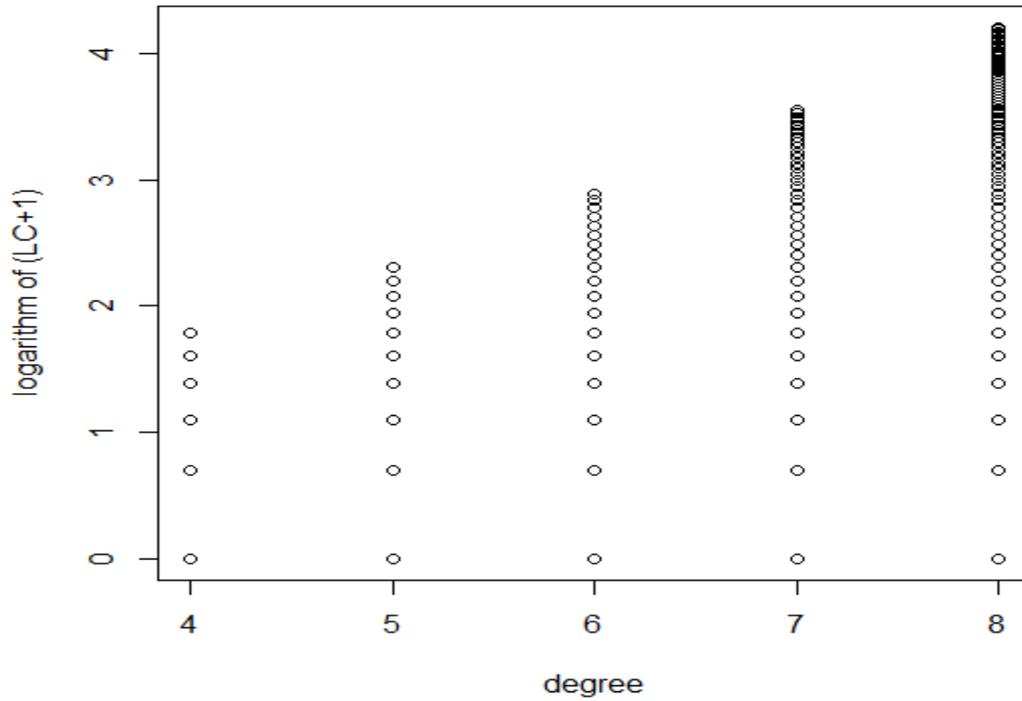


Table 3.2.3: Simple linear regression of degree versus LC+1 value

	Estimate	Std. Error	t	p-value
(Intercept)	-0.32985	0.072745	-4.534	5.82E-06
degree	0.328899	0.009577	34.342	2.00E-16

Thus the relationship between degree and LC is estimated to be

$$\ln(LC + 1) = -0.33 + 0.33 \times \text{degree}$$

This can be rewritten as

$$LC = 0.72e^{0.33 \times \text{degree}} - 1$$

## Results for the profile data

Firstly, we examine a contingency table.

Table 3.2.4: Contingency table of degree versus grouped LC value

	LC grouped			
	[0,10]	(10,20]	(20,30]	>30
4	16	0	0	0
5	96	0	0	0
degree 6	113	79	0	0
7	357	355	360	80
8	311	330	322	1085

The Chi-square test for association is highly significant,  $\chi^2(12)=1460.034$ ,  $p<0.001$ . Therefore that conclude there is an association between degree and LC. From Table 3.2.1, it is clear that most polynomials have a high degree and high LC (e.g. degree 8 and LC greater than 30), and very few have low degree and low LC (e.g. degree 4 and LC between 0 and 10 inclusive).

To determine the nature of the relationship between the variables, we examine the scatter plot and fit a simple linear regression.

Figure 3.2.4: Scatter Plot of LC versus degree

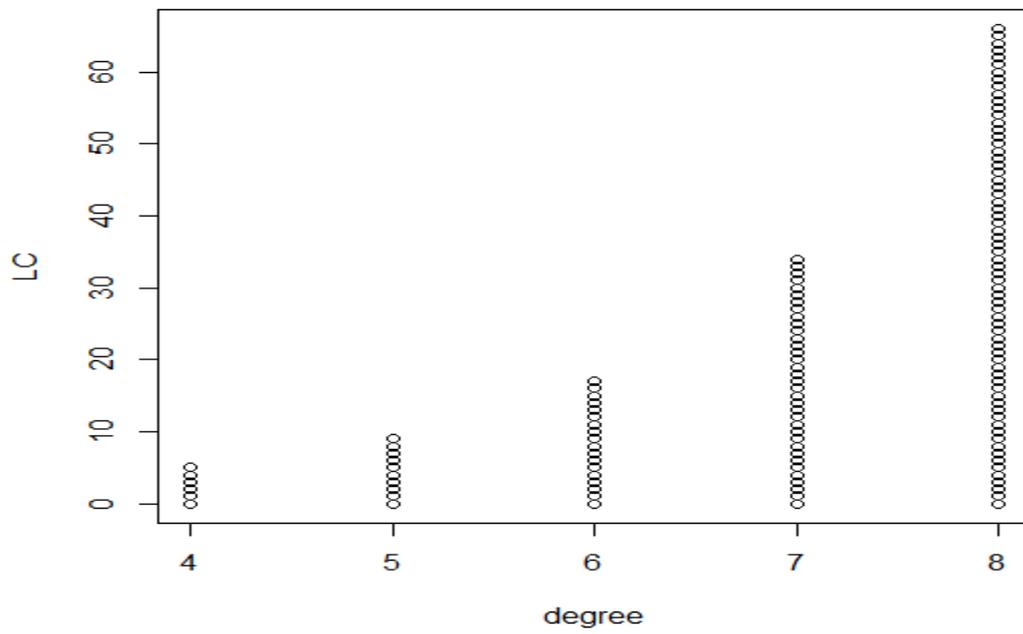


Figure 3.2.5: Scatter Plot of logarithm of LC+1 versus degree

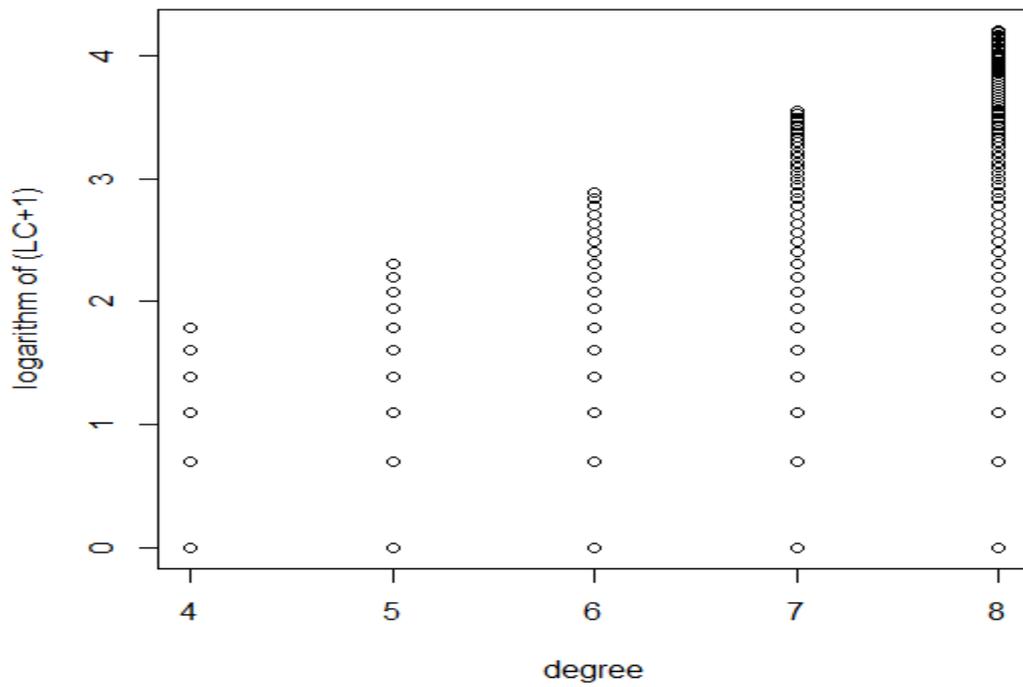


Table 3.2.2: Simple linear regression of degree versus LC+1 value

	Estimate	Std. Error	t	p-value
(Intercept)	-1.371	0.13074	-10.49	<2e-16
Degree	0.5785	0.01743	33.19	<2e-16

Thus the relationship between degree and LC is estimated to be

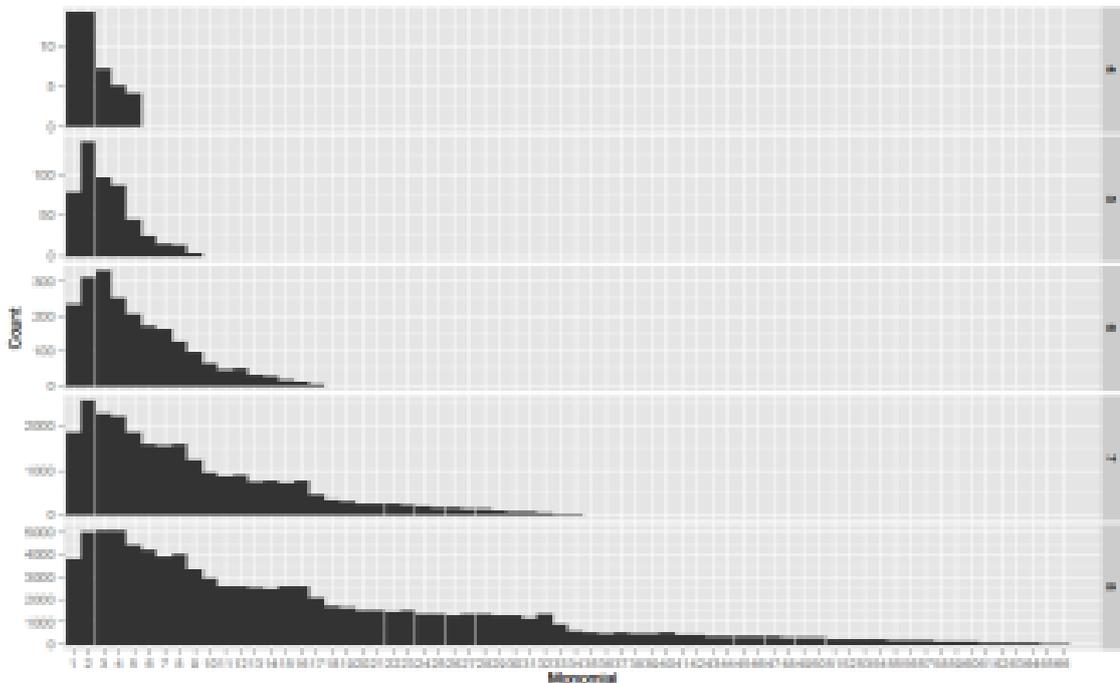
$$\ln(LC + 1) = -1.37 + 0.58 \times \text{degree}$$

This can be rewritten as

$$LC = 0.25e^{0.25 \times \text{degree}} - 1$$

The relationship appears to be exponential in nature, so the logarithm of LC+1 is taken prior to fitting a regression. Note that one needs to be added to LC to avoid taking the logarithm of zero, which is undefined.

In the following figure it represents linear complexity using chunk bits of self-shrinking generator output for all primitive polynomials of degrees 4 to 8. To calculate linear complexity, we took the first 4 chunk bits starting from bit 1, 2, 3...etc. bits of each ssg output and applied the Berlekamp-Massey algorithm. Then taking 8, chunk bits 16... etc.



### Appendix 3.3: Some statistical analysis for UWS4 to UWS24

Table 3.3.1: An overall statistical analysis of the UWS for shrinking generator with degrees from 4 to 24

Degree	Weight mean	UWS mean	Weight variance	UWS variance	Weight standard deviation	UWS standard deviation	UWS Weight Correlation
4	3	4	0	0	0	0	NA
5	4.3333	7.5	1.06666	4.3	1.032795	2.073644	0.747087
6	4.3333	7.6666	1.0666	1.06666	1.032795	1.032795	-0.25
7	5	11.055	1.88235	6.17320	1.37198	2.484593	-0.03451
8	5.5	13.437	0.8	4.12916	0.894427	2.032035	-0.27510
9	6.25	14.9166	1.63829	4.03546	1.27996	2.00884	-0.17377
10	6.533	17.766	2.35480	4.18192	1.53453	2.04497	0.169954
11	7.0227	18.931	2.33090	3.22961	1.52673	1.79711	-0.06191
12	7.3611	21.402	1.82672	4.07439	1.35156	2.01851	-0.0383
13	8.0476	23.325	2.7544	4.1626	1.6596	2.0402	0.0259
14	8.6031	25.284	2.88338	3.43027	1.69805	1.85209	-0.0171
15	8.9422	27.345	3.5986	3.8594	1.8970	1.9645	-0.0009
16	9.5117	29.354	3.5601	3.4237	1.8868	1.8503	0.0046
17	9.9644	31.284	4.0477	3.3906	2.0119	1.8413	-0.013
18	10.525	33.360	4.23499	3.65402	2.05790	1.91155	-0.0064
19	11.011	35.322	4.46157	3.53126	2.11224	1.87916	-0.0100
20	11.502	37.330	4.78885	3.51279	2.18834	1.87424	-0.0008
21	11.992	39.331	4.95370	3.5344	2.22569	1.88002	-0.0040
22	12.505	41.332	5.22508	3.55562	2.28584	1.88563	-0.0018
23	13.002	43.3346	5.51737	3.5444	2.34890	1.88268	0.00074
24	13.503	45.3346	5.7573	3.5067	2.3994	1.8726	0.0018

### Appendix 3.4: Unique Window Size degree vs polynomial weight

The following table shows all primitive polynomials of degree 9 and 10 with their weight and unique window size:

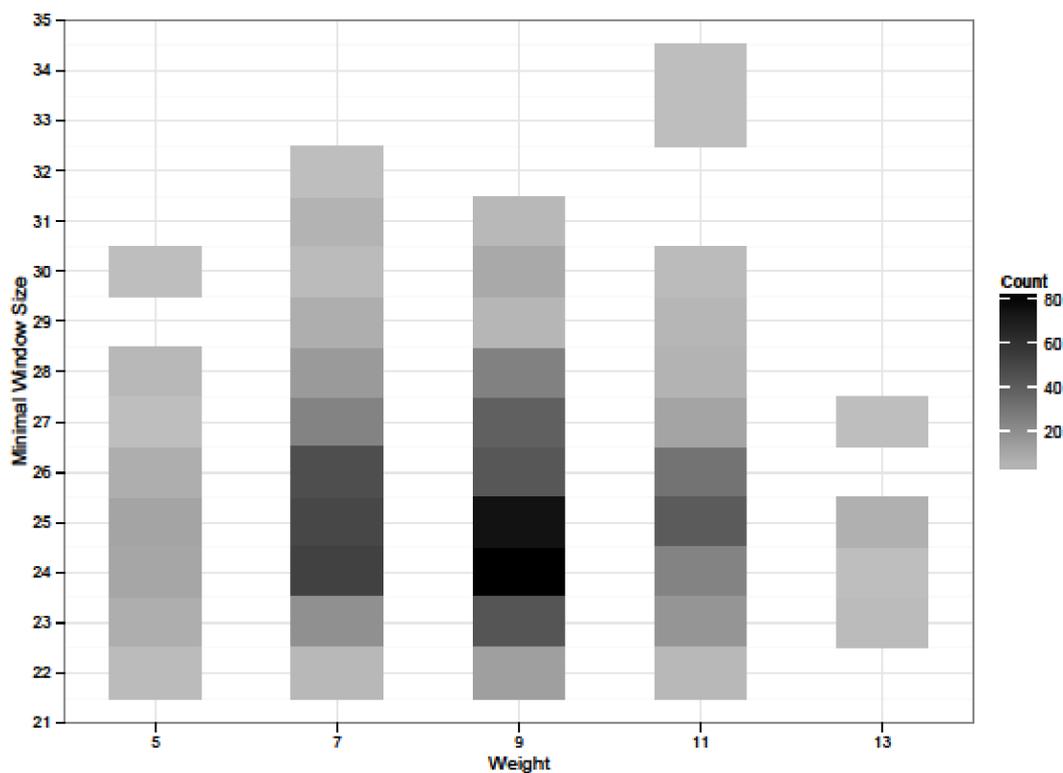
Table 3.4.1 weight of poly (degree 9) and Unique Window Size with (frequencies)

Weight	Unique Window Size with (frequencies)
3	16(1) 17(1)
5	12(1) 13(4) 14(4) 15(1) 16(2) 18(3) 20(1)
7	12(1) 13(6) 14(10) 15(3) 16(2) 17(3) 18(1) 19(2)
9	13(1) 15(1)

Table 3.4.2 weight of poly (degree 10) and Unique Window Size with (frequencies)

Weight	Unique Window Size with (frequencies)
3	15(1) 16(1)
5	15(2) 16(2) 17(9) 18(3) 19(2) 20(1) 23(1)
7	17(8) 18(4) 19(5) 20(2) 22(2) 24(1)
9	15(2) 16(1) 17(1) 18(3) 19(1) 20(1) 23(1)

The following histogram shows the calculations for primitive polynomials of degree 14, for weight versus unique window.



UWS21, weight =5

uws	count	p(uws)
36	2	0.012195122
37	21	0.12804878
38	38	0.231707317
39	36	0.219512195
40	32	0.195121951
41	13	0.079268293
42	16	0.097560976
43	5	0.030487805
46	1	0.006097561

UWS21 and all weights

uws	count	P(uws)
35	32	0.000377929
36	1522	0.017975246
37	9835	0.116154101
38	19746	0.233205782
39	20229	0.238910147
40	14697	0.17357568
41	8729	0.103091931
42	4790	0.05657124
43	2538	0.02997449
44	1237	0.014609316
45	639	0.007546769
46	355	0.004192649
47	160	0.001889645
48	75	0.000885771
49	32	0.000377929
50	28	0.000330688
51	13	0.000153534
52	6	7.0862E-05
53	2	2.3621E-05
56	3	3.5431E-05
58	1	1.181E-05
62	1	1.181E-05
63	1	1.181E-05
85	1	1.181E-05

### Appendix 3.5: Goodness of fit, plots and figures for different statistical distributions

This report includes the goodness of fit, as well as plots and figures for different statistical distributions comparison for SG with UWS2.

Table 3.5.1 Goodness of fit for possible probability distribution for SG with UWS20

<b>Distribution</b>	<b>Log-likelihood</b>	<b>AIC</b>	<b>BIC</b>	<b>Ranking</b>
Poisson	-207633.1	415268.3	415277.4	5
Negative Binomial	-207633.1	415270.3	415288.6	6
Geometric	-332299.8	664601.7	664610.8	7
Normal	-194858.1	389720.1	389738.4	3
Weibull	-207206.1	414416.3	414434.3	4
Gamma	-192544.2	385092.4	385110.7	2
Log normal	-191560.4	383124.7	383143	1

\*highest log likelihood, lowest AIC and BIC

Table 3.5.2 Test for randomness

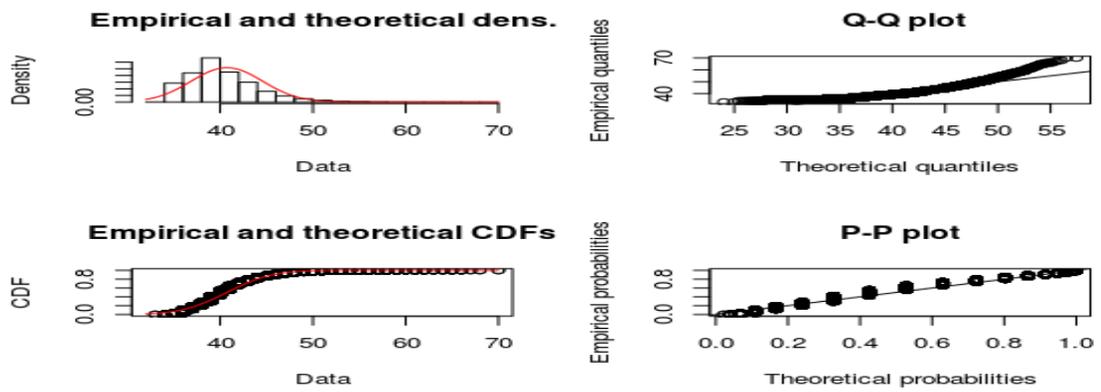
Test	Statistic	P value	Decision
Bartels rank test	-83.926	<0.001	Non-Random
Cox sturt test	8957	<0.001	Non-Random
Mann Kendal rank test	-139.68	<0.001	Non-Random

## Based on UWS20 variable for SG

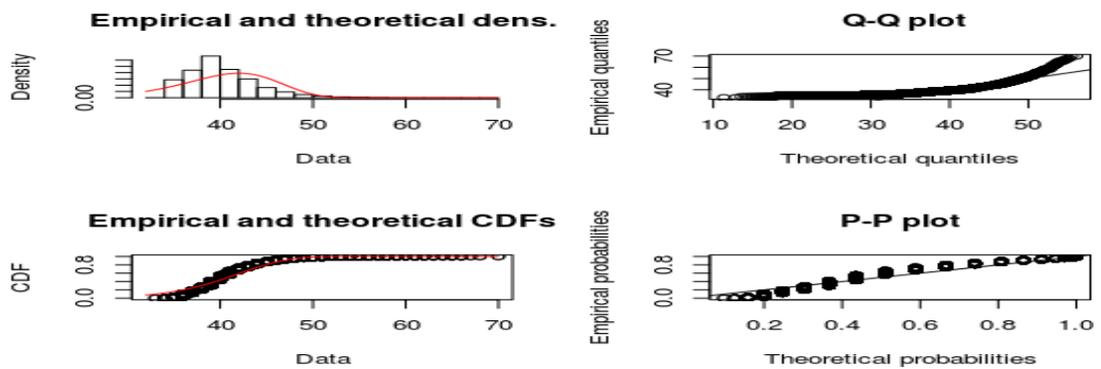
Observed and empirical distributions:

### First: Continuous distributions:

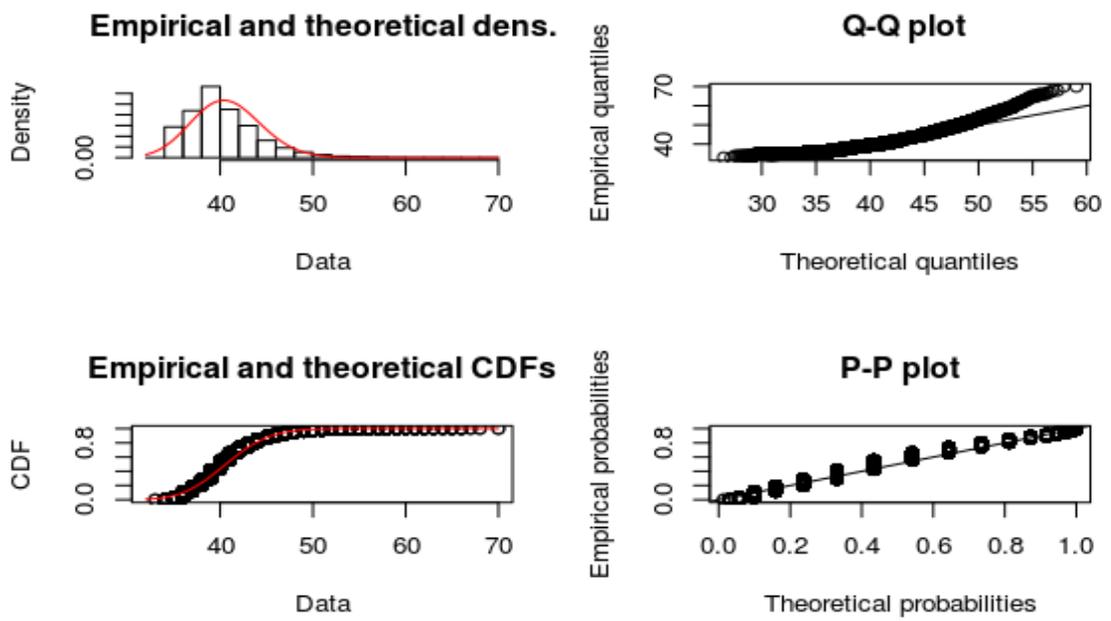
Normal plots:



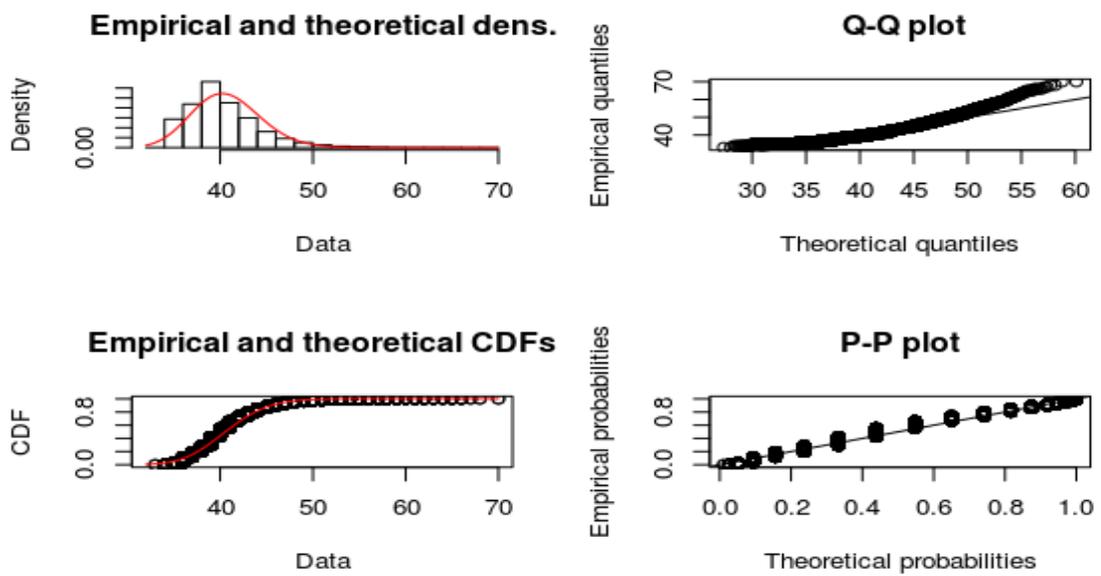
Weibull plots:



Gamma plots:

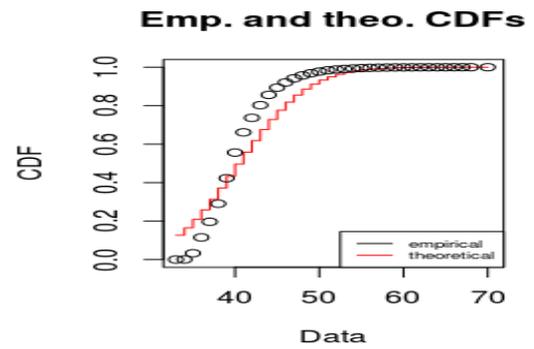
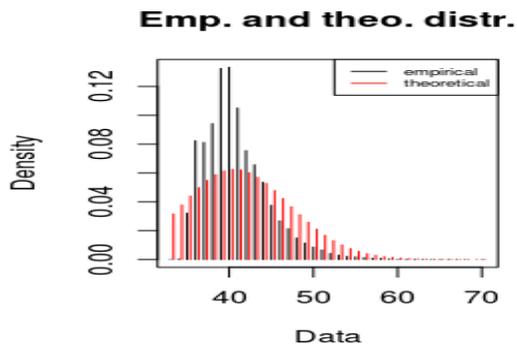


Log normal plots:

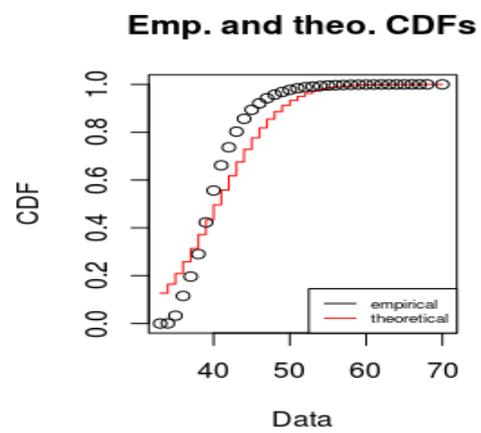
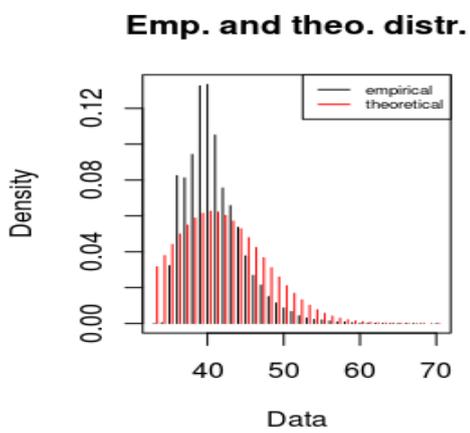


## Second: Discrete distributions:

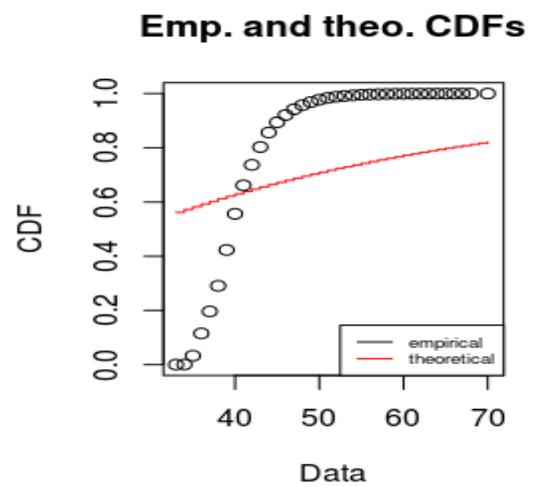
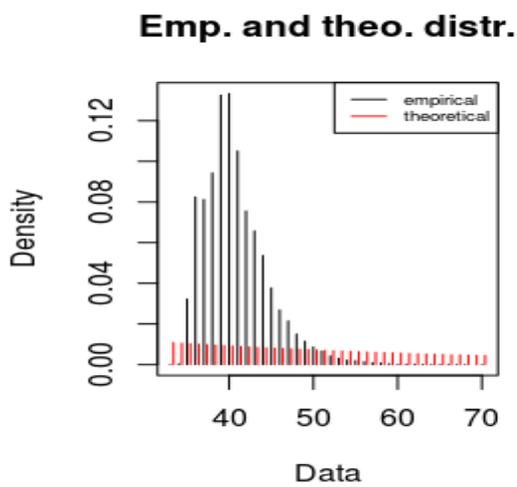
Poisson:



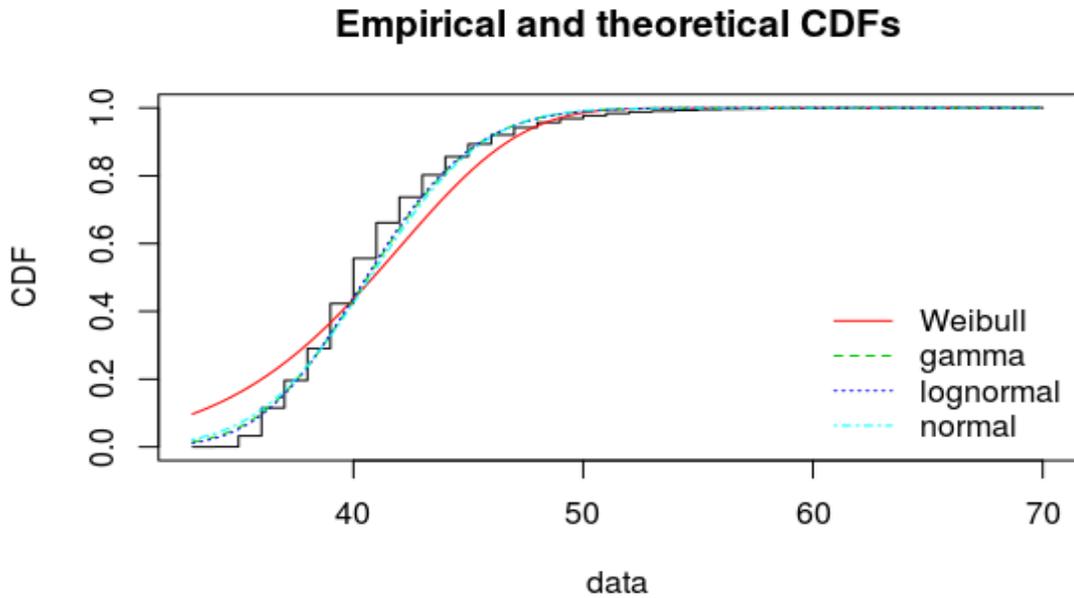
Negative Binomial:



Geometric:



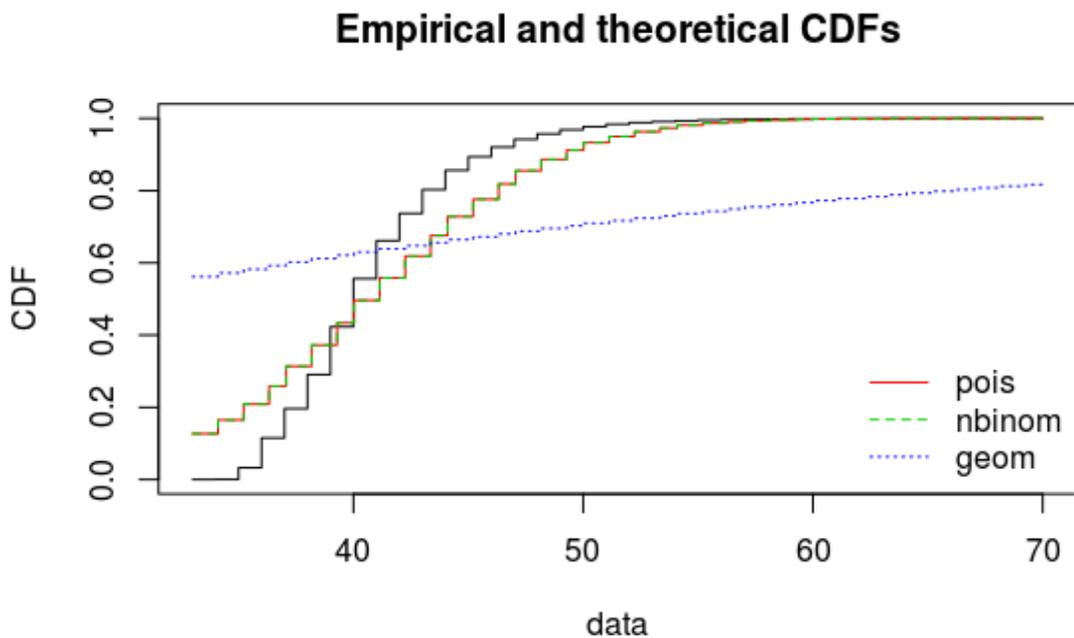
### Third: Comparisons between continuous distributions



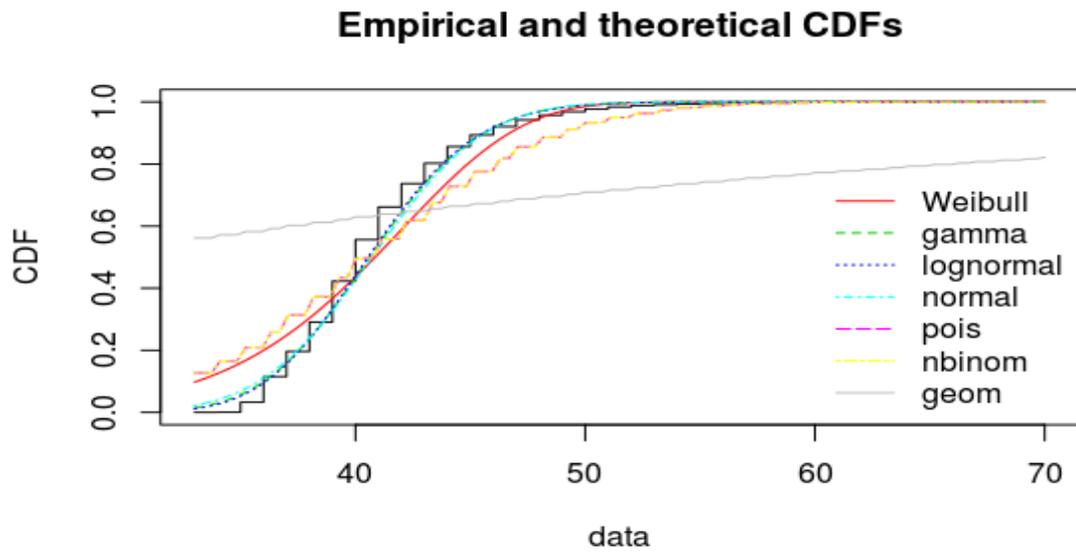
Weibull and gamma lines are on top of each other, hence only one red line visible.

### Fourth: Comparison between discrete distributions

Poisson and negative binomial lines are on top of each other, hence only one red line is visible.



**Fifth: Comparison between all (candidate) distributions**



**Appendix 3.6: Importance of variables based on univariate simple linear regression model R square**

<b>Variable</b>	<b>R Square</b>
<b>Input polynomial term of order 17</b>	<b>0.18</b>
<b>Input degree</b>	<b>0.127</b>
<b>Control degree</b>	<b>0.127</b>
<b>Control weight</b>	<b>0.10</b>
<b>Input weight</b>	<b>0.09</b>
<b>Input polynomial term of order 13</b>	<b>0.09</b>
<b>Input polynomial term of order 12</b>	<b>0.08</b>
<b>Input polynomial of order 14</b>	<b>0.08</b>
<b>Input polynomial term of order 15</b>	<b>0.08</b>
<b>Input polynomial term of order 16</b>	<b>0.08</b>
<b>Control polynomial term of order 9</b>	<b>0.078</b>
<b>Control polynomial term of order 11</b>	<b>0.068</b>
<b>Control polynomial term of order 8</b>	<b>0.057</b>
<b>Control polynomial term of order 13</b>	<b>0.054</b>
<b>Control polynomial term of order 10</b>	<b>0.041</b>
<b>Input polynomial term of order 10</b>	<b>0.04</b>
<b>Control polynomial term of order 4</b>	<b>0.038</b>
<b>Control polynomial term of order 5</b>	<b>0.036</b>
<b>Control polynomial term of order 6</b>	<b>0.036</b>
<b>Control polynomial term of order 3</b>	<b>0.032</b>
<b>Input polynomial term of order 11</b>	<b>0.018</b>
<b>Control polynomial term of order 12</b>	<b>0.018</b>
<b>Control polynomial term of order 7</b>	<b>0.015</b>
<b>Input polynomial term of order 8</b>	<b>0.014</b>
<b>Input polynomial term of order 7</b>	<b>0.008</b>
<b>Input polynomial term of order 9</b>	<b>0.002</b>
<b>Control polynomial term of order 17</b>	<b>0.001</b>
<b>Control polynomial term of order 14</b>	<b>0.0006</b>
<b>Input polynomial term of order 4</b>	<b>0.0005</b>
<b>Control polynomial term of order 16</b>	<b>0.0005</b>
<b>Control polynomial term of order 15</b>	<b>0.0004</b>
<b>Input polynomial term of order 2</b>	<b>0.00009</b>
<b>Control polynomial term of order 1</b>	<b>0.00009</b>
<b>Input polynomial term of order 1</b>	<b>0.00006</b>
<b>Input polynomial term of order 3</b>	<b>0.00006</b>
<b>Input polynomial term of order 6</b>	<b>0.00006</b>

## Appendices for Chapter 5

These appendices include explanations for the counting method of GEs as in Chapter 5, and comparison of GEs number in the MICKEY family algorithms.

### Appendix 5.1: Counting GEs method

The way to count the Gate Equivalents (GEs), which is the measurement unit to measure the electronic circuit complexity, is based on the operation used for computation as following:

**Table 5.1.** Number of GEs for a given logical gate; see [310].

Gate	Number of Gate Equivalents
NOT	1
AND	2
OR	2
XOR	3
NAND	1
NOR	1
XNOR	3
MUX	3

One of the objectives of this algorithm was to reduce the number of GEs. The following points summarise how the reduction was accomplished with a focus on those algorithm features used for reduction and indicates the kind of gate it represents. This enabled us to count the number of GEs.

The following part were published as a part of this thesis in [23]

CLOCK\_R 1: Initialization of the Internal Register (Single XOR)

CLOCK\_R 2: Loop (Conditional) Feedback Bit Logically Assigns Linear Register Bit (Within Loop)

MICKEY 2.0: for  $i = 0$  to 99

MICKEY 2.0.85: for  $i = 0$  to 84

CLOCK\_R 3: Linear (R\_MASK) Logic to Invert Bit (Single XOR) (Within Loop)

MICKEY 2.0: for  $i = 0$  to 99

MICKEY 2.0.85: for  $i = 0$  to 84

CLOCK\_R 4: Multiple Related Operation (Single MUX)—Conditionally executed based on control bit.

CLOCK\_R 5: Multiple Related Operation (Single MUX)—Conditionally executed based on feedback bit.

CLOCK\_S: 1 Initialization of Internal (Nonlinear) Register (Single XOR)

MICKEY 2.0: For  $i = 0$  to 99:

MICKEY 2.0.85: For  $i = 0$  to 84

CLOCK\_S: 2, CLOCK\_S: 3: Bitwise operations on internal structures 3 XORs and One AND (gates)

CLOCK\_S: 4: Conditional Logic on Feedback and Control Bit (Single MUX)

MICKEY 2.0: For  $i = 0$  to 99:

MICKEY 2.0.85: For  $i = 0$  to 84

CLOCK\_S: 5: Change Nonlinear Register (Single XOR)

CLOCK\_KG: 1–5: Simple Initializations: (4 XOR, 1 AND)

The IV and key were used along with the internal masks to initialize the registers in the function ECRYPT\_keysetup, ECRYPT\_ivsetup. The idea is that, by arbitrarily mixing the bits of the key and the IV, the initial state of both the linear and nonlinear registers will be unpredictable.

MICKEY 2.0: IV\_i 1: For  $i = 0$  to 79: Initialize on IV (Single MUX)

MICKEY 2.0: IV 2: For  $i = 0$  to 80: Initialize on Key (Single MUX)

Therefore, the MICKEY Algorithm works as follows:

MICKEY 2.0: Process (Single MUX to represent Logic):

1. Initialize the internal state using: IV, key, and CLOCK\_KG (which uses CLOCK\_R and CLOCK\_S) to mix in the IV and KEY bits based on the internal driver structures

(R\_MASK and COMP0, COMP1, FB0, FB1)

2. For each bit in the message invoke CLOCK\_KG

- a. CLOCK\_KG invokes CLOCK\_R, which advances the linear bit and masks it with R\_MASK to determine its final value.

- b. CLOCK\_KG also invokes CLOCK\_S, which may or may not advance the nonlinear bit depending on the linear position and the values of (COMP0, COMP1, FB0, and FB1)

- c. CLOCK\_KG determines the keystream bit by XORing the current linear and nonlinear registers and ANDs them with 1.

- d. Ciphertext Generation: The current plaintext message bit is XORed with the current Keystream bit, which becomes the ciphertext output.

## Appendix 5.2: GE number comparison in MICKEY family algorithms

The following part was published as a part of this thesis in [23]

The following table counts the number of GEs for both MICKEY 2.0 and MICKEY 2.0.85, considering the logical gates XOR, AND and MUX count. And that for internal state parts CLOCK\_R, CLOCK\_S, CLOCK\_KG, ECRYPT\_IVs setup and Encrypt\_process functions.

**Table 5.2.** GE Comparison between MICKEY Family Algorithms.

Function	Operation	GE Multiplier	MICKEY 2.0		MICKEY 2.0.85	
			Count Number	GE	Count Number	GE
CLOCK_R	XOR	3	401	1203	341	1023
	MUX	3	2	6	2	6
CLOCK_S	XOR	3	400	1200	340	1020
	AND	2	100	200	85	170
	MUX	3	2	6	2	6
CLOCK_KG	XOR	3	4	12	4	12
ECRYPT_IVs setup	MUX	3	160	480	160	480
Encrypt_process	MUX	3	8	24	8	24
Total GE					3131	2741

## Appendix for Chapter 6

### Appendix 6: Example of data that represents the communication operation between mobile device and the cloud server

#### Register

```
action=register&uid=431a53a9b877c8e7a1d00e485c4fbfa4&phone=%2B14165551212
&timestamp=1581955729&x=sig
s=OK&d=6178ebbcd400a83336c827e45c2d0ae3b8e96f1fc6daf2b5fe3b0b03cac9762b42
61e8166190499758717cb49e86985670077636e168ebdfa4f6e26ebf589a01&t=15819557
29
```

```
onetimepad =
6178ebbcd400a83336c827e45c2d0ae3b8e96f1fc6daf2b5fe3b0b03cac9762b4261e81661
90499758717cb49e86985670077636e168ebdfa4f6e26ebf589a01
```

#### Update

```
action=update&uid=431a53a9b877c8e7a1d00e485c4fbfa4&data=06c219e5bc8378f3a8a
3f83b4b7e4649&timestamp=1581955729&x=sig
s=OK&d=d8fafca7c6611a58e4e63894e99acdc606d43480639c8cfcc25bb3f8729b8b5996
7565f014af18d03600c497b6f9d0b1cca4f057086d40fac195c2700a450a88&t=158195572
9&x=f2053690b839f7f0bdfb6208f682fae0
```

```
sharedkeystream =
b982171b1261b26bd22e1f70b5b7c725be3d5b9fa5467e493c60b8fbb852fd72d4148de67
53f51476e71b823287f48e7bca38661e905ab256563201eb51d9089
```

#### Validate

```
action=validate&uid=431a53a9b877c8e7a1d00e485c4fbfa4&data=06c219e5bc8378f3a8
a3f83b4b7e4649&timestamp=1581955729&x=sig
s=OK&t=1581955729
```

#### Generate

```
action=generate&uid=431a53a9b877c8e7a1d00e485c4fbfa4&file=63158a3230a6ff1383c
8ef9f4790ca56&number=262144&expire=1581955759&timestamp=1581955729&x=sig
s=OK&t=1581955730
```

#### Request

**(download keystream)**

action=request&uid=431a53a9b877c8e7a1d00e485c4fbfa4&file=63158a3230a6ff1383c8ef9f4790ca56&timestamp=1581955730&x=sig

s=OK&d=c2b51d7e6ada70cb58b84de530350ca28a839ebb83ee5092f034c7e2e07ca5a1b276b53868c42feffa085357b76d5607bc02d8ce2df63f24b2bd098c354ca551210acd29f58c375d880

.....

keystream\_encrypted =

c2b51d7e6ada70cb58b84de530350ca28a839ebb83ee5092f034c7e2e07ca5a1b276b53868c42feffa085357b76d5607bc02d8ce2df63f24b2bd098c354ca551210acd29.....

keystream =

7b370a6578bbc2a08a9652958582cb8734bec52426a82edbcc547f19582e58d3666238de1dfb7ea891d13d1653099d87c763abed0bdac8d72e48f08676495adcab92bbc98d.....

**Upload**

**(upload encrypted file)**

action=upload&uid=431a53a9b877c8e7a1d00e485c4fbfa4&file=cfb57d776aed4b34e3d0f35440a925a4&data=4a34fe4762155e24e893312b610c2a7dcf089dc69eb2411cbcea51a69a263c21a0fab....

s=OK&t=1581955734

file\_contents = Shhh! Don't tell anyone. This is the secret message I am trying to send to my friend.

file\_inflated =

536868682120446f6e27742074656c6c20616e796f6e652e20546869732069732074686520736563726574206d657373616765204920616d20747279696e6720746f2073656e6420746f206d7920667269656e642e

file\_encrypted =

f3b6e95c7074ec4f3abd2e5bd4bbbed587135c6593bf43f55808ae95d2274c15374ee321e80ade4572331dc0039ff2908c1b9e4141874c94d74ee2802c9b0e614253b885331f43e5bd4b1a1592874ce523dff341f8e

file\_encrypted\_sk =  
4a34fe4762155e24e893312b610c2a7dcf089dc69eb2411cbcea51a69a263c21a0fabff8f59  
2b5104d406423118061ef7d1a6275f1716268118d081c7cad769d9cb99f4823958c30069f  
be299dc3097783c26f802b

## Upload

### (upload magic file)

action=upload&uid=431a53a9b877c8e7a1d00e485c4fbfa4&file=d4e3707271&data=b98  
2171c1261b26fd22e1f8fc380ad49c6b1a2c094b8926dc97e5a7a62c5605bf6d3c09e24d9a  
1a89c56b550&expire=1581955764&timestamp=1581955734&x=sig  
s=OK&t=1581955734

magic\_filename = magic

magic\_filename\_inflated = 6d61676963

magic\_filename\_encrypted = d4e370727

## Request

### (download magic file)

action=request&uid=431a53a9b877c8e7a1d00e485c4fbfa4&file=d4e3707271&timestamp=1581955734&x=sig  
s=OK&d=b982171c1261b26fd22e1f8fc380ad49c6b1a2c094b8926dc97e5a7a62c5605bf6  
d3c09e24d9a1a89c56b550&t=1581955734&x=04b4e10845b57dc9b4fbf5bb02d19bd4  
R0 = 7  
R1 = 4  
RN = 255

filename\_hashed = 76376a6c788cf95f31feec24f51ee281

token\_hashed = da979d2922c74d7851e6f0eff2270d73

## Request

### (download encrypted file)

action=request&uid=431a53a9b877c8e7a1d00e485c4fbfa4&file=cfb57d776aed4b34e3d0f35440a925a4&timestamp=1581955734&x=sig  
s=OK&d=4a34fe4762155e24e893312b610c2a7dcf089dc69eb2411cbcea51a69a263c21a0fabff8f592b5104d406423118061ef7d1a6275f1716268118d081c7cad769d9cb99f4823958c30069fbe299dc3097783c26f802b&t=1581955734&x=59a2859ad8dc9e68c4493fd941c7180f

file\_encrypted\_sk =  
4a34fe4762155e24e893312b610c2a7dcf089dc69eb2411cbcea51a69a263c21a0fabff8f592b5104d406423118061ef7d1a6275f1716268118d081c7cad769d9cb99f4823958c30069fbe299dc3097783c26f802b

file\_encrypted =  
f3b6e95c7074ec4f3abd2e5bd4bbed587135c6593bf43f55808ae95d2274c15374ee321e80ade4572331dc0039ff2908c1b9e4141874c94d74ee2802c9b0e614253b885331f43e5bd4b1a1592874ce523dff341f8e

file\_inflated =  
536868682120446f6e27742074656c6c20616e796f6e652e20546869732069732074686520736563726574206d657373616765204920616d20747279696e6720746f2073656e6420746f206d7920667269656e642e

file\_contents = Shhh! Don't tell anyone. This is the secret message I am trying to send to my friend.



## Appendix 7.2: NFSDE device emulation

The emulator functions as a command line program that displays a menu with the following options.

These are the main options suggested for display on the user interface and can be modelled according to user preferences.

MAIN MENU:

1. Create Patient Tag
2. Create Provider Tag
3. Activate Reader
4. Read Patient Tag
5. Unlock the USB
6. Change Key Number (Time stamp emulator)
7. Exit

### B. TEST RUN USING THE EMULATOR

1. To unlock the USB drive by using option 5, which makes the key available, the password is hard coded and displayed for convenience within the simulator. If the USB device is not unlocked,  $K_1$ ,  $K_0$ , and  $IV_0$  are not available to the simulator, and error messages are displayed. If the SD card is not “unlocked,” no creation, read, or activation can be performed.
2. Option 6 allows the simulation of a KTI rotation. For demonstration purposes, only two key sets were provided. This proves it is possible to encrypt the provider with one set of keys and the patient with another.
3. Creating a provider tag by using option 2 prompts the following process.
  - a. The provider’s identification, PIN, and authorization are entered.
  - b. A 7-byte unique ID is generated randomly.
  - c. Encryption is performed, and authentication code (as described above) is generated.
  - d. A file of the form “[uniqueid].enc” is used to simulate the tag (in this case, the provider tag.) This includes an unencrypted value of “2” in the tag type field to ensure that subsequent scans “know” this is a provider tag rather than a patient tag.
  - e. A plaintext file of the form “[uniqueid].txt” is also created for checking the accuracy of the decryption process.
4. Option 1 for creating a patient tag has two major features:

- a. Emulation of the reference fingerprint scan, which is performed by simply specifying one of the three hex files provided to serve as the reference fingerprint (we used three to enable us to emulate incorrect or failed scans).
  - b. Emulation of reading and encrypting the identity and “medical data.” We used a random name generator and a random string generator to emulate the patient identity and medical data.
  - c. A file of the form “[uniqueid].enc” is used to simulate the tag (in this case, the patient tag). This includes the unencrypted value of “1” in the tag type field to enable subsequent scans to “know” this is a patient tag rather than a provider tag.
  - d. A plaintext file of the form “[uniqueid].txt” is also created to verify the accuracy of the decryption process.
5. Option 3 (reader activation) begins by prompting the provider tag to be scanned. This scan is emulated by entering the filename of a provider tag that has already been created (“[uniqueid].enc”). If the file contains 2 (provider) in the tag type field, the authentication signature is decrypted and checked (including CRC). If the signature matches all acceptance criteria and it is assured that the data have not been tampered with, the device is “activated” and the authorization level of the provider is stored in the device memory.
6. Option 4 reads the patient tag. If the device has not been activated (Option 3), Option 4 fails immediately, prompting for activation. The RFID card is “scanned” by entering the filename of a previously created patient tag (“[uniqueid].enc”). If the file contains the patient value of 1 in the tag type field, the authentication signature is decrypted and checked (including CRC). Further, if the signature matches all acceptance criteria and it is assured that the data have not been tampered with, fingerprint scanning is performed in the next step. Fingerprint scanning is emulated by entering one of the fingerprint file numbers. A “good scan” is emulated by entering the same number as in the reference fingerprint template, whereas a “bad scan” is emulated by entering one of the other numbers. If the fingerprint matches, the medical data will be decrypted and displayed (based on the authorization level at activation.)

### **Appendix 7.3: PUF and possibility of use within NFSDE**

Physically Unclonable Functions (PUFs) have two major functional benefits: key generation and lightweight authentication [250], [251]. We do not need PUFs for key generation but could use them for lightweight authentication.

When an environmentally stable PUF becomes readily available for emergency use, the protocol we have developed should dove-tail into this technology. The emulator functions can be modified to use the protocol.

In a scenario that requires less frequent key rotation and in which potential problems caused by the environment are not life-threatening, off-the-shelf components could be replaced by commercially available special order CRFID tags. These tags, for providers only, could serve the same function as the SD.  $IV_0$ ,  $K_1$  and  $K_0$  could be encrypted and stored on the providers' CRFID cards and decrypted when his or her PIN is authenticated.

To pursue this line of inquiry, the software-based components of the emulator could be adopted to experiment on the best way to implement the protocol to guide the designer for specific hardware implementations without limiting designer creativity [309]. Adding a fuzzy extractor [251] to the emulator would definitely give us some better insights on implementation details necessary for PUFs.

For example, the RFID tag simulator component might be replaced by a CRFID with a PUF simulator component. The USB component may, for example, be used for storing expected responses to registered challenges or may be replaced or removed entirely.

## References

- [1] Fagbemi DD, Wheeler DM, Wheeler JC. The IoT Architect's Guide to Attainable Security and Privacy. CRC Press; 2019 Oct 8.
- [2] Le D-N, Bhatt C, Madhukar M. Security Designs for the Cloud, IoT, and Social Networking. John Wiley & Sons; 2019.
- [3] Jouini M, Rabai LB. A Computational Approach for Secure Cloud Computing Environments. In Modern Principles, Practices, and Algorithms for Cloud Security 2020 (pp. 129-144). IGI Global.
- [4] Poudel M, Shrestha S, Sarode RP, Chu W, Bhalla S. Query Languages for Polystore Databases for Large Scientific Data Archives [Internet]. 2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence). 2019. Available from: <http://dx.doi.org/10.1109/confluence.2019.8776972>
- [5] Mo J, Hu Z, Chen H, Shen W. An efficient and provably secure anonymous user authentication and key agreement for mobile cloud computing. Wireless Communications and Mobile Computing. 2019;2019.
- [6] Zhao R, Wang D, Zhang Q, Chen H, Huang A. CRH: A Contactless Respiration and Heartbeat Monitoring System with COTS RFID Tags. In 2018 15th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON) 2018 Jun 11 (pp. 1-9). IEEE.
- [7] He Y, Wang G, Li W, Ren Y. Improved Cube Attacks on Some Authenticated Encryption Ciphers and Stream Ciphers in the Internet of Things. IEEE Access. 2020 Jan 17;8:20920-30.
- [8] Pasqualini L, Parton M. Pseudo Random Number Generation: a Reinforcement Learning approach. Procedia Computer Science. 2020 Jan 1;170:1122-7.
- [9] Dinh HT, Lee C, Niyato D, Wang P. A survey of mobile cloud computing: architecture, applications, and approaches. Wireless communications and mobile computing. 2013 Dec 25;13(18):1587-611.
- [10] Want R. An introduction to RFID technology. IEEE pervasive computing. 2006 Feb 13;5(1):25-33.
- [11] Juels A. RFID security and privacy: A research survey. IEEE journal on selected areas in communications. 2006 Feb 6;24(2):381-94.
- [12] Van Tilborg HC, Jajodia S, editors. Encyclopedia of cryptography and security. Springer Science & Business Media; 2014 Jul 8.

- [13] Reactive.IO - Better Software, Faster [Internet]. [cited 2020 Apr 30]. Available from: <https://reactive.io/>
- [14] Hong J, Sarkar P. New applications of time memory data tradeoffs. In *International Conference on the Theory and Application of Cryptology and Information Security 2005* Dec 4 (pp. 353-372). Springer, Berlin, Heidelberg.
- [15] Hong J, Sarkar P. Rediscovery of Time Memory Tradeoffs. *IACR Cryptology ePrint Archive*. 2005 Mar;2005:90.
- [16] Menezes AJ, van Oorschot PC, Vanstone SA. *Handbook of Applied Cryptography* [Internet]. 2018. Available from: <http://dx.doi.org/10.1201/9780429466335>
- [17] Paar C, Pelzl J. *Understanding cryptography: a textbook for students and practitioners*. Springer Science & Business Media; 2009 Nov 27.
- [18] Buchanan WJ, Li S, Asif R. Lightweight cryptography methods. *Journal of Cyber Security Technology*. 2017 Oct 1;1(3-4):187-201.
- [19] Klein A. The eStream Project [Internet]. *Stream Ciphers*. 2013. p. 229–39. Available from: [http://dx.doi.org/10.1007/978-1-4471-5079-4\\_10](http://dx.doi.org/10.1007/978-1-4471-5079-4_10)
- [20] Omrani T, Rhouma R, Becheikh R. LICID: a lightweight image cryptosystem for IoT devices. *Cryptologia*. 2019 Jul 4;43(4):313-43.
- [21] Al\_Janabi S, Hussein NY. The reality and future of the secure mobile cloud computing (SMCC): survey. In *International Conference on Big Data and Networks Technologies 2019* Apr 29 (pp. 231-261). Springer, Cham.
- [22] Website [Internet]. [cited 2020 Apr 25]. Available from: Amazon Web Services, Inc. (2020). Amazon EC2. [online] Available at: <https://aws.amazon.com/ec2/>
- [23] Alamer A, Soh B, Brumbaugh DE. MICKEY 2.0. 85: A Secure and Lighter MICKEY 2.0 Cipher Variant with Improved Power Consumption for Smaller Devices in the IoT. *Symmetry*. 2020 Jan;12(1):32.
- [24] Coppersmith D, Krawczyk H, Mansour Y. The shrinking generator. In *Annual International Cryptology Conference 1993* Aug 22 (pp. 22-39). Springer, Berlin, Heidelberg.
- [25] Meier W, Staffelbach O. The self-shrinking generator. In *Communications and Cryptography 1994* (pp. 287-295). Springer, Boston, MA.
- [26] Gupta BB, Sheng QZ, editors. *Machine Learning for Computer and Cyber Security: Principle, Algorithms, and Practices*. CRC Press; 2019 Feb 5.
- [27] Rukhin A, Soto J, Nechvatal J, Smid M, Barker E. A statistical test suite for random and pseudorandom number generators for cryptographic applications. Booz-allen and hamilton inc mclean va; 2001 May 15.

- [28] He D, Zeadally S. An analysis of RFID authentication schemes for internet of things in healthcare environment using elliptic curve cryptography. *IEEE internet of things journal*. 2014 Sep 23;2(1):72-83.
- [29] Cohen EA. The Code Book: The Evolution of Secrecy from Mary, Queen of Scots to Quantum Cryptography. *Foreign Affairs*. 1999 Nov 1;78(6):148.
- [30] Rosenblatt AL. The code book: the evolution of secrecy from Mary Queen of Scots to quantum cryptography [Books]. *IEEE Spectrum*. 2000 Oct;37(10):10-4.
- [31] Singh S. The code book: the science of secrecy from ancient Egypt to quantum cryptography. Anchor; 2000.
- [32] Karzig T, Knapp C, Lutchyn RM, Bonderson P, Hastings MB, Nayak C, Alicea J, Flensberg K, Plugge S, Oreg Y, Marcus CM. Scalable designs for quasiparticle-poisoning-protected topological quantum computation with Majorana zero modes. *Physical Review B*. 2017 Jun 21;95(23):235305.
- [33] Chabaud F, Stern J. The cryptographic security of the syndrome decoding problem for rank distance codes. In *International Conference on the Theory and Application of Cryptology and Information Security* 1996 Nov 3 (pp. 368-381). Springer, Berlin, Heidelberg.
- [34] Englund H, Johansson T, Turan MS. A framework for chosen IV statistical analysis of stream ciphers. In *International Conference on Cryptology in India* 2007 Dec 9 (pp. 268-281). Springer, Berlin, Heidelberg.
- [35] Bellare M, Paterson KG, Rogaway P. Security of symmetric encryption against mass surveillance. In *Annual Cryptology Conference* 2014 Aug 17 (pp. 1-19). Springer, Berlin, Heidelberg.
- [36] Boyd C, Mathuria A, Stebila D. Authentication and key transport using public key cryptography. In *Protocols for Authentication and Key Establishment* 2020 (pp. 135-164). Springer, Berlin, Heidelberg.
- [37] Avoine G, Canard S, Ferreira L. Symmetric-key authenticated key exchange (SAKE) with perfect forward secrecy. In *Cryptographers' Track at the RSA Conference* 2020 Feb 24 (pp. 199-224). Springer, Cham.
- [38] Chen CM, Huang Y, Wang KH, Kumari S, Wu ME. A secure authenticated and key exchange scheme for fog computing. *Enterprise Information Systems*. 2020 Jan 12:1-6.
- [39] Salehi SA. Low-Cost Stochastic Number Generators for Stochastic Computing. Vol. 28, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*. 2020. p. 992–1001.

- [40] James F, Moneta L. Review of High-Quality Random Number Generators. Vol. 4, Computing and Software for Big Science. 2020.
- [41] Courtois NT. Fast algebraic attacks on stream ciphers with linear feedback. In Annual International Cryptology Conference 2003 Aug 17 (pp. 176-194). Springer, Berlin, Heidelberg.
- [42] Burman S, Mukhopadhyay D, Veezhinathan K. LFSR based stream ciphers are vulnerable to power attacks. In International Conference on Cryptology in India 2007 Dec 9 (pp. 384-392). Springer, Berlin, Heidelberg.
- [43] Feng GL, Tzeng KK. A generalization of the Berlekamp-Massey algorithm for multisequence shift-register synthesis with applications to decoding cyclic codes. IEEE Transactions on Information Theory. 1991 Sep;37(5):1274-87.
- [44] Lihua D, Yupu H. Weak generalized self-shrinking generators. Journal of Systems Engineering and Electronics. 2007 Jun;18(2):407-11.
- [45] Hu Y, Xiao G. Generalized self-shrinking generator. IEEE Transactions on Information Theory. 2004 Mar 30;50(4):714-9.
- [46] Mogos G. Quantum random number generator vs. random number generator. In 2016 International Conference on Communications (COMM) 2016 Jun 9 (pp. 423-426). IEEE.
- [47] Wang Y, Xiang S, Wang B, Cao X, Wen A, Hao Y. Time-delay signature concealment and physical random bits generation in mutually coupled semiconductor lasers with FBG filtered injection. Optics express. 2019 Mar 18;27(6):8446-55.
- [48] Irfan M, Ali A, Khan MA, Ehatisham-ul-Haq M, Mehmood Shah SN, Saboor A, Ahmad W. Pseudorandom Number Generator (PRNG) Design Using Hyper-Chaotic Modified Robust Logistic Map (HC-MRLM). Electronics. 2020 Jan;9(1):104.
- [49] Moggia E. Generalized Quasi-Random Lattice model for electrolyte solutions: Apparent and partial molal heat capacities. Fluid Phase Equilibria. 2020 Feb 1;505:112358.
- [50] Rezk AA, Madian AH, Radwan AG, Soliman AM. Reconfigurable chaotic pseudo random number generator based on FPGA. AEU-International Journal of Electronics and Communications. 2019 Jan 1;98:174-80.
- [51] Peetermans A, Rozic V, Verbauwhede I. A highly-portable true random number generator based on coherent sampling. In 2019 29th International Conference on Field Programmable Logic and Applications (FPL) 2019 Sep 8 (pp. 218-224). IEEE.

- [52] Schepers D, Ranganathan A, Vanhoef M. Practical Side-Channel Attacks against WPA-TKIP. In Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security 2019 Jul 2 (pp. 415-426).
- [53] Hinek MJ. Cryptanalysis of RSA and its variants. CRC press; 2009 Jul 21.
- [54] Erguler I, Anarim E. A new cryptanalytic time-memory trade-off for stream ciphers. In International Symposium on Computer and Information Sciences 2005 Oct 26 (pp. 215-223). Springer, Berlin, Heidelberg.
- [55] Jean J, Nikolić I, Peyrin T, Wang L, Wu S. Security analysis of PRINCE. In International Workshop on Fast Software Encryption 2013 Mar 11 (pp. 92-111). Springer, Berlin, Heidelberg.
- [56] Babbage SH. Improved “exhaustive search” attacks on stream ciphers. 1995 Jan 1;161–6.
- [57] Golić JD. Cryptanalysis of alleged A5 stream cipher. In International Conference on the Theory and Applications of Cryptographic Techniques 1997 May 11 (pp. 239-255). Springer, Berlin, Heidelberg.
- [58] Turan MS, Çalık Ç, Saran NB, Doğanaksoy A. New distinguishers based on random mappings against stream ciphers. In International Conference on Sequences and Their Applications 2008 Sep 14 (pp. 30-41). Springer, Berlin, Heidelberg.
- [59] Rueppel RA. Analysis and design of stream ciphers. Springer Science & Business Media; 2012 Dec 6.
- [60] Berbain C, Gilbert H. On the security of IV dependent stream ciphers. In International Workshop on Fast Software Encryption 2007 Mar 26 (pp. 254-273). Springer, Berlin, Heidelberg.
- [61] Pareschi F, Rovatti R, Setti G. Second-level NIST randomness tests for improving test reliability. In 2007 IEEE International Symposium on Circuits and Systems 2007 May 27 (pp. 1437-1440). IEEE.
- [62] Šýs M, Říha Z. Faster randomness testing with the NIST statistical test suite. In International Conference on Security, Privacy, and Applied Cryptography Engineering 2014 Oct 18 (pp. 272-284). Springer, Cham.
- [63] List JA, Shaikh AM, Xu Y. Multiple hypothesis testing in experimental economics. *Experimental Economics*. 2019 Dec 1;22(4):773-93.
- [64] Alekseychuk AN, Konyushok SN. On the Efficiency of the Probabilistic Neutral Bits Method in Statistical Cryptanalysis of Synchronous Stream Ciphers. *Cybernetics and Systems Analysis*. 2016 Jul 1;52(4):503-8.

- [65] The eSTREAM portfolio page [Internet]. [cited 2019 Aug 6]. Available from: <http://www.ecrypt.eu.org/stream>
- [66] Liu H, Wang B. Mitigating File-Injection Attacks with Natural Language Processing. In Proceedings of the Sixth International Workshop on Security and Privacy Analytics 2020 Mar 16 (pp. 3-13).
- [67] Ghafari VA, Hu H. A new chosen IV statistical distinguishing framework to attack symmetric ciphers, and its application to ACORN-v3 and Grain-128a. *Journal of Ambient Intelligence and Humanized Computing*. 2019 Jun 1;10(6):2393-400.
- [68] Sibleyras F. Generic Attack on Iterated Tweakable FX Constructions. In CT-RSA 2020-The Cryptographers' Track at the RSA Conference 2020 Feb 24.
- [69] Amine FM, Abdelkader G. Hybrid Approach of Modified AES. In *Cryptography: Breakthroughs in Research and Practice 2020* (pp. 129-141). IGI Global.
- [70] Maimut D, Ouafi K. Lightweight cryptography for RFID tags. *IEEE Security & Privacy*. 2012 Apr 3;10(2):76-9.
- [71] Sangariand S, Manickam L. A light-weight cryptography analysis for wireless based healthcare applications. *J Comput Sci*. 2014;10(10):2088-94.
- [72] Griotti M, Gandino F, Rebaudengo M. Transitory Master Key Transport Layer Security for WSNs. *IEEE Access*. 2020 Jan 23;8:20304-12.
- [73] Reggiani A, Romanelli R, Tritapepe T, Nijkamp P. NEURAL NETWORKS: AN OVERVIEW AND APPLICATIONS IN THE SPACE ECONOMY. IN: NEURAL NETWORKS IN TRANSPORT APPLICATIONS. Atmospheric Environment. 1998.
- [74] Chen D, Li S, Liao L. A recurrent neural network applied to optimal motion control of mobile robots with physical constraints. *Applied Soft Computing*. 2019 Dec 1;85:105880.
- [75] Li H, Huang Z, Fu J, Li Y, Zeng N, Zhang J, Ye C, Jin L. Modified weights-and-structure-determination neural network for pattern classification of flatfoot. *IEEE Access*. 2019 May 10;7:63146-54.
- [76] Yu B, Wang Z, Zhu R, Feng X, Qi M, Li J, Zhao R, Huang L, Xin R, Li F, Zhou F. The Transverse Ultrasonogram of Thyroid Papillary Carcinoma Has a Better Prediction Accuracy Than the Longitudinal One. *IEEE Access*. 2019 Jul 2;7:100763-70.
- [77] Fasoli D, Panzeri S. Mathematical studies of the dynamics of finite-size binary neural networks: A review of recent progress. *Math Biosci Eng*. 2019 Sep 4;16(6):8025–59.

- [78] Raghu S, Sriraam N, Hegde AS, Kubben PL. A novel approach for classification of epileptic seizures using matrix determinant. *Expert Systems with Applications*. 2019 Aug 1;127:323-41.
- [79] Raghu S, Sriraam N, Temel Y, Rao SV, Hegde AS, Kubben PL. Performance evaluation of DWT based sigmoid entropy in time and frequency domains for automated detection of epileptic seizures using SVM classifier. *Computers in biology and medicine*. 2019 Jul 1;110:127-43.
- [80] Raghu S, Sriraam N, Rao SV, Hegde AS, Kubben PL. Automated detection of epileptic seizures using successive decomposition index and support vector machine classifier in long-term EEG. *Neural Computing and Applications*. 2019 Jul 31:1-20.
- [81] Kimmel J, Brack A, Marshall W. Deep convolutional and recurrent neural networks for cell motility discrimination and prediction. *IEEE/ACM Trans Comput Biol Bioinform*. 2019 Jun 27.
- [82] Duong BP, Khan SA, Shon D, Im K, Park J, Lim DS, Jang B, Kim JM. A Reliable Health Indicator for Fault Prognosis of Bearings. *Sensors*. 2018 Nov;18(11):3740.
- [83] Basu S, Karuppiah M, Nasipuri M, Halder AK, Radhakrishnan N. Bio-inspired cryptosystem with DNA cryptography and neural networks. *Journal of Systems Architecture*. 2019 Mar 1;94:24-31.
- [84] Shaikh JR, Beniwal R, Iliev G. Cryptography and optimization-driven support vector neural network to mitigate DoS attacks in E-commerce. In *Applications of computing, automation and wireless systems in electrical engineering 2019* (pp. 551-561). Springer, Singapore.
- [85] Mell PM, Grance T. SP 800-145. *The NIST Definition of Cloud Computing*, National Institute of Standards & Technology, Gaithersburg, MD. 2011 Sep.
- [86] Qi Q, Tao F. A Smart Manufacturing Service System Based on Edge Computing, Fog Computing, and Cloud Computing. *IEEE Access*. 2019 Jun 19;7:86769-77.
- [87] Gulabani S. *Amazon Web Services Bootcamp: Develop a scalable, reliable, and highly available cloud environment with AWS*. Packt Publishing Ltd; 2018 Mar 30.
- [88] Soh J, Copeland M, Puca A, Harris M. *Microsoft Azure: Managing the Intelligent Cloud*. Apress; 2020.
- [89] Krishnan SP, Gonzalez JL. *Building your next big thing with google cloud platform: A guide for developers and enterprise architects*. Apress; 2015 May 22.
- [90] Chaka JG, Marimuthu M. Curtailing the Threats to Cloud Computing in the Fourth Industrial Revolution. In *Cloud Security: Concepts, Methodologies, Tools, and Applications 2019* (pp. 1-30). IGI Global.

- [91] Jouini M, Rabai LB. A security framework for secure cloud computing environments. In *Cloud security: Concepts, methodologies, tools, and applications 2019* (pp. 249-263). IGI Global.
- [92] Thota C, Sundarasekar R, Manogaran G, Varatharajan R, Priyan MK. Centralized fog computing security platform for IoT and cloud in healthcare system. In *Fog Computing: Breakthroughs in Research and Practice 2018* (pp. 365-378). IGI global.
- [93] Vijayakumar V, Priyan MK, Ushadevi G, Varatharajan R, Manogaran G, Tarare PV. E-health cloud security using timing enabled proxy re-encryption. *Mobile Networks and Applications*. 2019 Jun 15;24(3):1034-45.
- [94] Mohanty SP, Yanambaka VP, Kougianos E, Puthal D. PUFchain: A Hardware-Assisted Blockchain for Sustainable Simultaneous Device and Data Security in the Internet of Everything (IoE). *IEEE Consumer Electronics Magazine*. 2020 Feb 3;9(2):8-16.
- [95] Chaudhry SA, Kim IL, Rho S, Farash MS, Shon T. An improved anonymous authentication scheme for distributed mobile cloud computing services. *Cluster Computing*. 2019 Jan 16;22(1):1595-609.
- [96] Smith S. "Internet of Things" Connected Devices to Almost Triple to Over 38 Billion Units by 2020 [Internet]. [cited 2020 Apr 25]. Available from: <https://www.juniperresearch.com/press/press-releases/iot-connected-devices-to-triple-to-38-bn-by-2020>
- [97] Atre H, Razdan K, Sagar RK. Offloading Computation for Efficient Mobile Cloud Computing. *Indian Journal of Science and Technology*. 2016 Jun;9(22):1-6.
- [98] Seok B, Sicato JC, Erzhen T, Xuan C, Pan Y, Park JH. Secure D2D Communication for 5G IoT Network Based on Lightweight Cryptography. *Applied Sciences*. 2020 Jan;10(1):217.
- [99] Sarode RP, Bhalla S. Data Security in Mobile Cloud Computing. Available at SSRN 3352362. 2019 Mar 14.
- [100] Dey S, Ye Q, Sampalli S. A machine learning based intrusion detection scheme for data fusion in mobile clouds involving heterogeneous client networks. *Information Fusion*. 2019 Sep 1;49:205-15.
- [101] Sarkar A, Dey J, Bhowmik A, Mandal JK, Karforma S. Computational Intelligence Based Neural Session Key Generation on E-Health System for Ischemic Heart Disease Information Sharing. In *Contemporary Advances in Innovative and Applicable Information Technology 2019* (pp. 23-30). Springer, Singapore.
- [102] Matolcsy B, Zolomy A. Designing an Efficient Ultra Small Form Factor On-Chip Antenna for UHF RFID Application. *Radioengineering*. 2019 Jun 1;29(2).

- [103] Ali Z, Perret E, Barbot N, Siragusa R, Hély D, Bernier M, Garet F. Detection of Natural Randomness by Chipless RFID Approach and Its Application to Authentication. *IEEE Transactions on Microwave Theory and Techniques*. 2019 May 16;67(9):3867-81.
- [104] Singh P, Acharya B, Chaurasiya RK. A comparative survey on lightweight block ciphers for resource constrained applications. *International Journal of High Performance Systems Architecture*. 2019;8(4):250-70.
- [105] Carstensen-Opitz C, Fine B, Moldenhauer A, Rosenberger G. *Abstract Algebra: Applications to Galois Theory, Algebraic Geometry, Representation Theory and Cryptography*. Walter de Gruyter GmbH & Co KG; 2019.
- [106] Pieprzyk J, Wang H, Zhang XM. Möbius transforms, coincident Boolean functions and non-coincidence property of Boolean functions. *International Journal of Computer Mathematics*. 2011 May 1;88(7):1398-416.
- [107] Verma JP, Abdel-Salam AS. *Testing statistical assumptions in research*. John Wiley & Sons; 2019 Mar 4.
- [108] Pace L. *Beginning R: An introduction to statistical programming*. Apress; 2012 Nov 28.
- [109] Benjamini I, Schramm O, Wilson DB. Balanced boolean functions that can be evaluated so that every input bit is unlikely to be read. In: *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*. New York, NY, USA: Association for Computing Machinery; 2005. p. 244–50. (STOC '05).
- [110] KB S, Aithal G. Generation of pseudo random number sequence from discrete oscillating samples of equally spread objects and application for stream cipher system. *Concurrency and Computation: Practice and Experience*. 2020 Jan 10;32(1):e5181.
- [111] Golomb SW. *Shift register sequences*. Aegean Park Press; 1967.
- [112] Burdakov O, Ioannis C, Demetriou and Panos M. Pardalos (eds): *Approximation and Optimization: Algorithms, Complexity and Applications*. In *SN Operations Research Forum* 2020 Mar (Vol. 1, No. 1, pp. 1-5). Springer International Publishing.
- [113] Jansen CJ. The maximum order complexity of sequence ensembles. In *Workshop on the Theory and Application of Cryptographic Techniques* 1991 Apr 8 (pp. 153-159). Springer, Berlin, Heidelberg.
- [114] Erdmann D, Murphy S. An approximate distribution for the maximum order complexity. *Designs, Codes and Cryptography*. 1997 Mar 1;10(3):325-39.
- [115] Fontaine C. Synchronous Stream Cipher. *Encyclopedia of Cryptography and Security*. p. 603–603. Available from: [http://dx.doi.org/10.1007/0-387-23483-7\\_423](http://dx.doi.org/10.1007/0-387-23483-7_423)

- [116] De Cannière C. Trivium: A stream cipher construction inspired by block cipher design principles. In *International Conference on Information Security 2006* Aug 30 (pp. 171-186). Springer, Berlin, Heidelberg.
- [117] Hell M, Johansson T, Meier W. Grain: a stream cipher for constrained environments. *IJWMC*. 2007 May 1;2(1):86-93.
- [118] Yifang W, Rong Z, Yi C. A self-synchronous stream cipher based on composite discrete chaos. In *2009 8th IEEE International Conference on Cognitive Informatics 2009* Jun 15 (pp. 210-214). IEEE.
- [119] Jiang S, Gong G. On edit distance attack to alternating step generator. In *Mathematical Properties of Sequences and Other Combinatorial Structures 2003* (pp. 85-92). Springer, Boston, MA.
- [120] Jiang H, Li C, Fan J. Research on Pseudo-Random Characteristics of New Random Components. In *2019 International Conference on Artificial Intelligence and Advanced Manufacturing (AIAM) 2019* Oct 16 (pp. 163-167). IEEE.
- [121] Filiol E. A new statistical testing for symmetric ciphers and hash functions. In *International Conference on Information and Communications Security 2002* Dec 9 (pp. 342-353). Springer, Berlin, Heidelberg.
- [122] Saarinen M-JO. Chosen-IV Statistical Attacks on eStream Ciphers. In: *SECRYPT*. 2006. p. 260–6.
- [123] Alamer A, Soh B. Design and Implementation of a Statistical Testing Framework for a Lightweight Stream Cipher. *Engineering, Technology & Applied Science Research*. 2020 Feb 3;10(1):5132-41.
- [124] Golić JD. Correlation analysis of the shrinking generator. In *Annual International Cryptology Conference 2001* Aug 19 (pp. 440-457). Springer, Berlin, Heidelberg.
- [125] Zhang B, Wu H, Feng D, Bao F. A fast correlation attack on the shrinking generator. In *Cryptographers' Track at the RSA Conference 2005* Feb 14 (pp. 72-86). Springer, Berlin, Heidelberg.
- [126] Golic JD, Menicocci R. Statistical distinguishers for irregularly decimated linear recurring sequences. *IEEE transactions on information theory*. 2006 Mar 6;52(3):1153-9.
- [127] Ekdahl P, Meier W, Johansson T. Predicting the shrinking generator with fixed connections. In *International Conference on the Theory and Applications of Cryptographic Techniques 2003* May 4 (pp. 330-344). Springer, Berlin, Heidelberg.
- [128] Boztaş S, Alamer A. Statistical dependencies in the self-shrinking generator. In *2015 Seventh International Workshop on Signal Design and its Applications in Communications (IWSDA) 2015* Sep 14 (pp. 42-46). IEEE.

- [129] Zenner E, Krause M, Lucks S. Improved cryptanalysis of the self-shrinking generator. In *Australasian Conference on Information Security and Privacy 2001* Jul 11 (pp. 21-35). Springer, Berlin, Heidelberg.
- [130] Debraize B, Goubin L. Guess-and-determine algebraic attack on the self-shrinking generator. In *International Workshop on Fast Software Encryption 2008* Feb 10 (pp. 235-252). Springer, Berlin, Heidelberg.
- [131] Website [Internet]. [cited 2020 Apr 26]. Available from: " >Technologies, M., 2020. Easyfit - Distribution Fitting Software. [online] Mathwave.com. Available at: <<http://www.mathwave.com/>>
- [132] Website [Internet]. [cited 2020 Apr 26]. Available from: " >Cran.r-project.org. 2020. [online] Available at: <<https://cran.r-project.org/web/packages/fitdistrplus/fitdistrplus.pdf>>
- [133] Massey J. Shift-register synthesis and BCH decoding. *IEEE transactions on Information Theory*. 1969 Jan;15(1):122-7.
- [134] Akriotou M, Mesaritakis C, Grivas E, Chaintoutis C, Fragkos A, Syvridis D. Random number generation from a secure photonic physical unclonable hardware module. In *International ISCIS Security Workshop 2018* Feb 26 (pp. 28-37). Springer, Cham.
- [135] Hagan M, Demuth H, Beale M, De Jesus O. *Neural Network Design*, Boston.
- [136] Park DC, El-Sharkawi MA, Marks RJ, Atlas LE, Damborg MJ. Electric load forecasting using an artificial neural network. *IEEE transactions on Power Systems*. 1991 May;6(2):442-9.
- [137] Patel J, Shah S, Thakkar P, Kotecha K. Predicting stock market index using fusion of machine learning techniques. *Expert Systems with Applications*. 2015 Mar 1;42(4):2162-72.
- [138] Xiao Y, Wu J, Lin Z, Zhao X. A deep learning-based multi-model ensemble method for cancer prediction. *Comput Methods Programs Biomed*. 2018 Jan;153:1–9.
- [139] Zoph B, Vasudevan V, Shlens J, Le QV. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition 2018* (pp. 8697-8710).
- [140] Hertz J, Krogh A, Palmer RG, Horner H. Introduction to the theory of neural computation. *Physics Today*. 1991;44:70.
- [141] Nugraha AA, Liutkus A, Vincent E. Deep neural network based multichannel audio source separation. In *Audio Source Separation 2018* (pp. 157-185). Springer, Cham.

- [142] Coutinho M, de Oliveira Albuquerque R, Borges F, Garcia Villalba LJ, Kim TH. Learning perfectly secure cryptography to protect communications with adversarial neural cryptography. *Sensors*. 2018 May;18(5):1306.
- [143] Arvandi M, Wu S, Sadeghian A. On the use of recurrent neural networks to design symmetric ciphers. *IEEE computational intelligence magazine*. 2008 Apr 18;3(2):42-53.
- [144] Diffie W, Hellman M. New directions in cryptography. *IEEE transactions on Information Theory*. 1976 Nov;22(6):644-54.
- [145] Kinzel W, Kanter I. Interacting neural networks and cryptography. In *Advances in solid state physics 2002* (pp. 383-391). Springer, Berlin, Heidelberg.
- [146] Godhavari T, Alamelu NR, Soundararajan R. Cryptography using neural network. In *2005 Annual IEEE India Conference-Indicon 2005 Dec 11* (pp. 258-261). IEEE.
- [147] Ruttor A, Kinzel W, Kanter I. Neural cryptography with queries. *Journal of Statistical Mechanics: Theory and Experiment*. 2005 Jan 24;2005(01):P01009.
- [148] Yu W, Cao J. Cryptography based on delayed chaotic neural networks. *Physics Letters A*. 2006 Aug 14;356(4-5):333-8.
- [149] Guo, D., Cheng, L. & Cheng, L. A New Symmetric Probabilistic Encryption Scheme Based on Chaotic Attractors of Neural Networks. *Applied Intelligence* 10, 71–84 (1999). <https://doi.org/10.1023/A:1008337631906>
- [150] Huan J. Deep-Learning: Investigating feed-forward deep Neural Networks for modeling high throughput chemical bioactivity data. In *2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM) 2016 Dec 15* (pp. 5-5). IEEE.
- [151] Singh A. Aarti nandal, "Neural Cryptography for Secret Key Exchange and Encryption with AES." *International Journal of Advanced Research in Computer Science and Software Engineering*. 2013;3(5):376–81.
- [152] Thabtah F, Mohammad RM, McCluskey L. A dynamic self-structuring neural network model to combat phishing. In *2016 International Joint Conference on Neural Networks (IJCNN) 2016 Jul 24* (pp. 4221-4226). IEEE.
- [153] Özkaynak F. Cryptographically secure random number generator with chaotic additional input. *Nonlinear Dynamics*. 2014 Nov 1;78(3):2015-20.
- [154] Dubrova E, Hell M. Espresso: A stream cipher for 5G wireless communication systems. *Cryptography and Communications*. 2017 Mar 1;9(2):273-89.
- [155] Turan MS. On the nonlinearity of maximum-length NFSR feedbacks. *Cryptography and Communications*. 2012 Dec 1;4(3-4):233-43.
- [156] Brandstätter N, Winterhof A. Linear complexity profile of binary sequences with small correlation measure. *Periodica Mathematica Hungarica*. 2006 Jun 1;52(2):1-8.

- [157] Mérai L, Winterhof A. On the pseudorandomness of automatic sequences. *Cryptography and Communications*. 2018 Nov 1;10(6):1013-22.
- [158] Mérai L, Niederreiter H, Winterhof A. Expansion complexity and linear complexity of sequences over finite fields. *Cryptography and Communications*. 2017 Jul 1;9(4):501-9.
- [159] Johansson T. Reduced complexity correlation attacks on two clock-controlled generators. In *International Conference on the Theory and Application of Cryptology and Information Security* 1998 Oct 18 (pp. 342-356). Springer, Berlin, Heidelberg.
- [160] Dittmer S, Emily J, Maass P. Singular values for relu layers. *IEEE transactions on neural networks and learning systems*. 2019 Nov 5.
- [161] Imamverdiyev Y, Sukhostat L. Lithological facies classification using deep convolutional neural network. *Journal of Petroleum Science and Engineering*. 2019 Mar 1;174:216-28.
- [162] Géron A. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media; 2019 Sep 5.
- [163] Website [Internet]. [cited 2020 Apr 26]. Available from: TensorFlow. (n.d.). Retrieved January 1, 2019, from <https://www.tensorflow.org>
- [164] Website [Internet]. [cited 2020 Apr 26]. Available from: Python.org. (n.d.). Welcome to Python.org. [online] Available at: <https://www.python.org>
- [165]. Website [Internet]. [cited 2020 Apr 26]. Available from: Filezilla-project.org. (n.d.). FileZilla - The free FTP solution. [online] Available at: <https://filezilla-project.org/>
- [166] Website [Internet]. [cited 2020 Apr 26]. Available from: [www.amazon ec2](http://www.amazon.com/ec2)
- [167] Lydia A, Francis S. Adagrad-An Optimizer for Stochastic Gradient Descent [Internet]. *INTERNATIONAL JOURNAL OF INFORMATION AND COMPUTING SCIENCE*. May; 2019. Available from: <http://ijics.com/gallery/92-may-1260.pdf>
- [168] Banik S, Maitra S, Sarkar S. Improved differential fault attack on MICKEY 2.0. *Journal of Cryptographic Engineering*. 2015 Apr 1;5(1):13-29.
- [169] Lara E, Aguilar L, García JA, Sanchez MA. A Lightweight Cipher Based on Salsa20 for Resource-Constrained IoT Devices. *Sensors*. 2018 Oct;18(10):3326.
- [170] Li S, Song H, Iqbal M. Privacy and Security for Resource-Constrained IoT Devices and Networks: Research Challenges and Opportunities. *Sensors* [Internet]. 2019 Apr 25;19(8). Available from: <http://dx.doi.org/10.3390/s19081935>
- [171] Ertaul L, Woodall A. IoT security: Performance evaluation of grain, mickey, and trivium-lightweight stream ciphers. In: *Proceedings of the International Conference on*

Security and Management (SAM). The Steering Committee of The World Congress in Computer Science, Computer ...; 2017. p. 32–8.

[172] Gurdur D. ARCHITECTURAL ENERGY-DELAY ASSESSMENT OF ABACUS MULTIPLIER WITH RESPECT TO OTHER MULTIPLIERS. Middle East Technical University Northern Cyprus Campus, Mersin-10 Turkey [Internet]. 2013; Available from: <http://etd.lib.metu.edu.tr/upload/12616231/index.pdf>

[173] Bui DH, Puschini D, Bacles-Min S, Beigné E, Tran XT. Ultra low-power and low-energy 32-bit datapath AES architecture for IoT applications. In 2016 International Conference on IC Design and Technology (ICICDT) 2016 Jun 27 (pp. 1-4). IEEE.

[174] Bogdanov A, Knudsen LR, Leander G, Paar C, Poschmann A, Robshaw MJ, Seurin Y, Vikkelsoe C. PRESENT: An ultra-lightweight block cipher. In International Workshop on Cryptographic Hardware and Embedded Systems 2007 Sep 10 (pp. 450-466). Springer, Berlin, Heidelberg.

[175] Wang K, Wang H, Wang Z, Yin Y, Mao L, Zhang Y. Method for pigment spectral matching identification based on adaptive levenshtein distance. *Optik*. 2019 Feb 1;178:74-82.

[176] Dutta IK, Ghosh B, Bayoumi M. Lightweight Cryptography for Internet of Insecure Things: A Survey. In 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC) 2019 Jan 7 (pp. 0475-0481). IEEE.

[177] Xilinx Power Estimator (XPE) [Internet]. Xilinx. [cited 2020 May 1]. Available from: <https://www.xilinx.com/products/technology/power/xpe.html>

[178] Simion E, Burciu P. A Note On the Correlations Between NIST Cryptographic Statistical Tests Suite. *UNIVERSITY POLITEHNICA OF BUCHAREST SCIENTIFIC BULLETIN-SERIES A-APPLIED MATHEMATICS AND PHYSICS*. 2019;81(1):209–18.

[179] Dutta IK, Ghosh B, Bayoumi M. Lightweight Cryptography for Internet of Insecure Things: A Survey. In 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC) 2019 Jan 7 (pp. 0475-0481). IEEE.

[180] Ding L, Liu C, Zhang Y, Ding Q. A new lightweight stream cipher based on chaos. *Symmetry*. 2019 Jul;11(7):853.

[181] McGinthy JM, Michaels AJ. Lightweight internet of things encryption using Galois extension field arithmetic. In 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) 2018 Jul 30 (pp. 74-80). IEEE.

- [182] Qasaimeh M, Al-Qassas RS, Tedmori S. Software randomness analysis and evaluation of lightweight ciphers: the prospective for IoT security. *Multimedia Tools and Applications*. 2018 Jul 1;77(14):18415-49.
- [183] Verma G, Khare V, Kumar M. More precise FPGA power estimation and validation tool (FPEV\_tool) for low power applications. *Wireless Personal Communications*. 2019 Jun 30;106(4):2237-46.
- [184] Website [Internet]. [cited 2020 Apr 26]. Available from: 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>.
- [185] Hong J, Kim WH. Tmd-tradeoff and state entropy loss considerations of streamcipher mickey. In *International Conference on Cryptology in India 2005 Dec 10* (pp. 169-182). Springer, Berlin, Heidelberg.
- [186] Website [Internet]. [cited 2020 Apr 26]. Available from: Babbage, S.; Dodd, M. The stream cipher MICKEY 2.0, ECRYPTStream Cipher, EU ECRYPT Netw., Denmark, U.K., Tech. Rep., 2006. Available: <https://www.ecrypt.eu.org/stream/index.html>
- [187] Koppula V, Waters B. Realizing chosen ciphertext security generically in attribute-based encryption and predicate encryption. In *Annual International Cryptology Conference 2019 Aug 18* (pp. 671-700). Springer, Cham.
- [188] Hofheinz D, Kamath A, Koppula V, Waters B. Adaptively secure constrained pseudorandom functions. In *International Conference on Financial Cryptography and Data Security 2019 Feb 18* (pp. 357-376). Springer, Cham.
- [189] Zhang S, Chen G. Micro-Trivium: A lightweight algorithm designed for radio frequency identification systems. *International Journal of Distributed Sensor Networks*. 2017 Feb;13(2):1550147717694171.
- [190] Liu D, Chen X, Peng D. Some cosine similarity measures and distance measures between q-rung orthopair fuzzy sets. *International Journal of Intelligent Systems*. 2019 Jul;34(7):1572-87.
- [191] Armbrust M, Fox A, Griffith R, Joseph AD, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A, Stoica I, Zaharia M. A view of cloud computing. *Communications of the ACM*. 2010 Apr 1;53(4):50-8.
- [192] Fernando N, Loke SW, Rahayu W. Mobile cloud computing: A survey. *Future generation computer systems*. 2013 Jan 1;29(1):84-106.
- [193] Number of mobile phone users worldwide 2015-2020 | Statista [Internet]. Statista. [cited 2020 Apr 26]. Available from: <https://www.statista.com/statistics/274774/forecast-of-mobile-phone-users-worldwide/>

- [194] Desolda G, Ardito C, Jetter HC, Lanzilotti R. Exploring spatially-aware cross-device interaction techniques for mobile collaborative sensemaking. *International Journal of Human-Computer Studies*. 2019 Feb 1;122:1-20.
- [195] Buyya R, Yeo CS, Venugopal S. Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities. In: 2008 10th IEEE International Conference on High Performance Computing and Communications. 2008. p. 5–13.
- [196] Subramanian N, Jeyaraj A. Recent security challenges in cloud computing. *Computers & Electrical Engineering*. 2018 Oct 1;71:28-42.
- [197] Singh G. A study of encryption algorithms (RSA, DES, 3DES and AES) for information security. *International Journal of Computer Applications*. 2013 Jan 1;67(19).
- [198] Bogdanov A, Mendel F, Regazzoni F, Rijmen V, Tischhauser E. ALE: AES-based lightweight authenticated encryption. In: *International Workshop on Fast Software Encryption* 2013 Mar 11 (pp. 447-466). Springer, Berlin, Heidelberg.
- [199] Doukas C, Maglogiannis I. Bringing IoT and cloud computing towards pervasive healthcare. In: *2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing* 2012 Jul 4 (pp. 922-926). IEEE.
- [200] Eisenbarth T, Kumar S, Paar C, Poschmann A, Uhsadel L. A survey of lightweight-cryptography implementations. *IEEE Design & Test of Computers*. 2007 Dec 6;24(6):522-33.
- [201] Kitsos P, Sklavos N, Provelengios G, Skodras AN. FPGA-based performance analysis of stream ciphers ZUC, Snow3g, Grain V1, Mickey V2, Trivium and E0. *Microprocessors and Microsystems*. 2013 Mar 1;37(2):235-45.
- [202] Manifavas C, Hatzivasilis G, Fysarakis K, Papaefstathiou Y. A survey of lightweight stream ciphers for embedded systems. *Security and Communication Networks*. 2016 Jul 10;9(10):1226-46.
- [203] Bahl P, Han RY, Li LE, Satyanarayanan M. Advancing the state of mobile cloud computing. In: *Proceedings of the third ACM workshop on Mobile cloud computing and services* 2012 Jun 25 (pp. 21-28).
- [204] Kumar K, Lu YH. Cloud computing for mobile users: Can offloading computation save energy?. *Computer*. 2010 Apr 8;43(4):51-6.
- [205] Bahrami M, Singhal M. A light-weight permutation based method for data privacy in mobile cloud computing. In: *2015 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering* 2015 Mar 30 (pp. 189-198). IEEE.

- [206] Daemen J, Rijmen V. The design of Rijndael: AES-the advanced encryption standard. Springer Science & Business Media; 2013 Mar 9.
- [207] Osvik DA, Bos JW, Stefan D, Canright D. Fast software AES encryption. InInternational Workshop on Fast Software Encryption 2010 Feb 7 (pp. 75-93). Springer, Berlin, Heidelberg.
- [208] Yoshikawa M, Goto H. Security Verification Simulator for Fault Analysis Attacks. Int. J. Soft Comput. Softw. Eng.[JSCSE]. 2013 Mar;3(3).
- [209] Buchmann J. Introduction to cryptography. Springer Science & Business Media; 2013 Dec 1.
- [210] Robshaw M, Billet O, editors. New stream cipher designs: the eSTREAM finalists. Springer; 2008 Jun 19.
- [211] Kardas S, Çelik S, Bingöl MA, Levi A. A new security and privacy framework for RFID in cloud computing. In2013 IEEE 5th International Conference on Cloud Computing Technology and Science 2013 Dec 2 (Vol. 1, pp. 171-176). IEEE.
- [212] Canteaut A, Carpov S, Fontaine C, Fournier J, Lac B, Naya-Plasencia M, Sirdey R, Tria A. End-to-end data security for IoT: from a cloud of encryptions to encryption in the cloud. InProc. IEEE Conf.(Cesar) 2017 Nov (pp. 1-21).
- [213] Diedrich L, Jattke P, Murati L, Senker M, Wiesmaier A. Comparison of Lightweight Stream Ciphers: MICKEY 2.0, WG-8, Grain and Trivium [Internet]. Unpublished; 2016. Available from: <https://pdfs.semanticscholar.org/e95a/63046ccda05182e17be584a37bd87350c6f8.pdf>
- [214] Banerjee A, Hasan M, Rahman MA, Chapagain R. Cloak: A stream cipher based encryption protocol for mobile cloud computing. IEEE Access. 2017 Aug 25;5:17678-91.
- [215] Babbage S, Dodd M. The stream cipher MICKEY 2.0. ECRYPT Stream Cipher. 2006 Jun.
- [216] Turan MS, Doganaksoy A, Calik C. Statistical analysis of synchronous stream ciphers. SASC 2006: Stream Ciphers Revisited. 2006 Feb 2.
- [217] Al Hinai S, Batten LM, Colbert B. Mutually clock-controlled feedback shift registers provide resistance to algebraic attacks. InInternational Conference on Information Security and Cryptology 2007 Aug 31 (pp. 201-215). Springer, Berlin, Heidelberg.
- [218] Kazmi AR, Afzal M, Amjad MF, Abbas H, Yang X. Algebraic side channel attack on trivium and grain ciphers. IEEE Access. 2017 Oct 25;5:23958-68.
- [219] Anand S, Perumal V. EECDDH to prevent MITM attack in cloud computing. Digital Communications and Networks. 2019 Nov 1;5(4):276-87.

- [220] Labs G. GSam Battery Monitor - Apps on Google Play [Internet]. [cited 2020 Apr 27]. Available from: <https://play.google.com/store/apps/details?id=com.gsamlabs.bbm&hl=en>
- [221] Alamer A, Soh B, Alahmadi AH, Brumbaugh DE. Prototype Device With Lightweight Protocol for Secure RFID Communication Without Reliable Connectivity. *IEEE Access*. 2019 Nov 19;7:168337-56.
- [222] Kim J, Cho J, Park D. Low-power command protection using SHA-CRC inversion-based scrambling technique for CAN-integrated automotive controllers. In *2018 IEEE Conference on Dependable and Secure Computing (DSC)* 2018 Dec 10 (pp. 1-2). IEEE.
- [223] BBC News. Fridge sends spam emails. BBC [Internet]. 2014 Jan 17 [cited 2020 Apr 27]; Available from: <https://www.bbc.com/news/technology-25780908>
- [224] Kaur M, Sandhu M, Mohan N, Sandhu PS. RFID technology principles, advantages, limitations & its applications. *International Journal of Computer and Electrical Engineering*. 2011 Feb 1;3(1):151.
- [225] Xiao Y, Shen X, Sun BO, Cai L. Security and privacy in RFID and applications in telemedicine. *IEEE communications magazine*. 2006 May 15;44(4):64-72.
- [226] Wang P, Chaudhry S, Li L, Li S, Tryfonas T, Li H. The Internet of Things: a security point of view. *Internet Research*. 2016 Apr 4.
- [227] Alaba FA, Othman M, Hashem IA, Alotaibi F. Internet of Things security: A survey. *Journal of Network and Computer Applications*. 2017 Jun 15;88:10-28.
- [228] Chamekh M, Hamdi M, El Asmi S, Kim TH. Security of RFID based Internet of Things applications: Requirements and open issues. In *2018 15th International Multi-Conference on Systems, Signals & Devices (SSD)* 2018 Mar 19 (pp. 699-703). IEEE.
- [229] Stapleton JJ. *Security without obscurity: A guide to confidentiality, authentication, and integrity*. CRC Press; 2014 May 2.
- [230] Galbraith SD. Authenticated key exchange for SIDH. *IACR Cryptology ePrint Archive*. 2018 Mar 13;2018:266.
- [231] Zhang Y, Xu L, Dong Q, Wang J, Blaauw D, Sylvester D. Recryptor: A Reconfigurable Cryptographic Cortex-M0 Processor With In-Memory and Near-Memory Computing for IoT Security. *IEEE J Solid-State Circuits*. 2018 Apr;53(4):995–1005.
- [232] Conti F, Schilling R, Schiavone PD, Pullini A, Rossi D, Gürkaynak FK, Muehlberghuber M, Gautschi M, Loi I, Haugou G, Mangard S. An IoT endpoint system-on-chip for secure and energy-efficient near-sensor analytics. *IEEE Transactions on Circuits and Systems I: Regular Papers*. 2017 May 13;64(9):2481-94.

- [233] Babbage S, Dodd M. The MICKEY Stream Ciphers [Internet]. Lecture Notes in Computer Science. p. 191–209. Available from: [http://dx.doi.org/10.1007/978-3-540-68351-3\\_15](http://dx.doi.org/10.1007/978-3-540-68351-3_15)
- [234] Banik S. Some studies on selected stream cipher, analysis, fault attack & related results [Internet]. Indian Statistical Institute, Kolkata; 2015. Available from: <http://library.isical.ac.in:8080/jspui/bitstream/123456789/6639/1/TH434.pdf>
- [235] Banik S, Maitra S. A differential fault attack on MICKEY 2.0. In International Workshop on Cryptographic Hardware and Embedded Systems 2013 Aug 20 (pp. 215-232). Springer, Berlin, Heidelberg.
- [236] Su Y, Gao Y, Kavehei O, Ranasinghe DC. Hash functions and benchmarks for resource constrained passive devices: A preliminary study. In 2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops) 2019 Mar 11 (pp. 1020-1025). IEEE.
- [237] Pawłowicz B, Salach M, Trybus B. Infrastructure of RFID-based smart city traffic control system. In Conference on Automation 2019 Mar 27 (pp. 186-198). Springer, Cham.
- [238] Missalot SK, Salama R, Liyanapathirana R. Dual-Band RFID Antenna Design for Infrastructure Health Monitoring. In: 2019 International Conference on Electrical Engineering Research Practice (ICEERP). 2019. p. 1–6.
- [239] Newton GD. A Billion Little Pieces: RFID and Infrastructures of Identification: Frith, J.(2019). A Billion Little Pieces: RFID and Infrastructures of Identification. Cambridge, MA: The MIT Press. 321 pages.
- [240] Chawla K, McFarland C, Robins G, Thomason W. An accurate real-time RFID-based location system. International Journal of Radio Frequency Identification Technology and Applications. 2018;5(1):48-76.
- [241] Lee JY, Lin WC, Huang YH. A lightweight authentication protocol for internet of things. In 2014 International Symposium on Next-Generation Electronics (ISNE) 2014 May 7 (pp. 1-2). IEEE.
- [242] Biryukov A, Perrin LP. State of the art in lightweight symmetric cryptography.
- [243] Billet O, Etrog J, Gilbert H. Lightweight privacy preserving authentication for RFID using a stream cipher. In International Workshop on Fast Software Encryption 2010 Feb 7 (pp. 55-74). Springer, Berlin, Heidelberg.
- [244] Wu W, Zhang L. LBlock: a lightweight block cipher. In International Conference on Applied Cryptography and Network Security 2011 Jun 7 (pp. 327-344). Springer, Berlin, Heidelberg.

- [245] Karakoç F, Demirci H, Harmancı AE. Impossible differential cryptanalysis of reduced-round LBlock. In IFIP International Workshop on Information Security Theory and Practice 2012 Jun 20 (pp. 179-188). Springer, Berlin, Heidelberg.
- [246] Ahson SA, Ilyas M. RFID handbook: applications, technology, security, and privacy. CRC press; 2017 Dec 19.
- [247] "eStream 2019 Foreword," 2019 Open Conference of Electrical, Electronic and Information Sciences (eStream), Vilnius, Lithuania, 2019, pp. i-v. Available from: <http://dx.doi.org/10.1109/estream.2019.8732147>
- [248] Mikhalev V, Armknecht F, Müller C. On ciphers that continuously access the non-volatile key. IACR Transactions on Symmetric Cryptology. 2016:52-79.
- [249] Bendavid Y, Bagheri N, Safkhani M, Rostampour S. IoT Device Security: Challenging "A Lightweight RFID Mutual Authentication Protocol Based on Physical Unclonable Function". Sensors. 2018 Dec;18(12):4444.
- [250] Gao Y, Su Y, Yang W, Chen S, Nepal S, Ranasinghe DC. Building secure SRAM PUF key generators on resource constrained devices. In 2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops) 2019 Mar 11 (pp. 912-917). IEEE.
- [251] Gao Y, Su Y, Xu L, Ranasinghe DC. Lightweight (reverse) fuzzy extractor with multiple reference puf responses. IEEE Transactions on Information Forensics and Security. 2018 Dec 13;14(7):1887-901.
- [252] Takpor T, Atayero AA. Integrating Internet of Things and EHealth solutions for students' healthcare. In Proceedings of the World Congress on Engineering 2015 (Vol. 1). World Congress on Engineering, London, UK.
- [253] Hiller J, Pennekamp J, Dahlmanns M, Henze M, Panchenko A, Wehrle K. Tailoring onion routing to the Internet of Things: Security and privacy in untrusted environments. In 2019 IEEE 27th International Conference on Network Protocols (ICNP) 2019 Oct 8 (pp. 1-12). IEEE.
- [254] Rodrigues JJ, Segundo DB, Junqueira HA, Sabino MH, Prince RM, Al-Muhtadi J, De Albuquerque VH. Enabling technologies for the internet of health things. Ieee Access. 2018 Jan 4;6:13129-41.
- [255] Ma Y, Wu Y, Ge J, Jun LI. An architecture for accountable anonymous access in the Internet-of-Things network. IEEE Access. 2018 Feb 15;6:14451-61.
- [256] Mocrii D, Chen Y, Musilek P. IoT-based smart homes: A review of system architecture, software, communications, privacy and security. Internet of Things. 2018 Sep 1;1:81-98.

- [257] Ammar M, Russello G, Crispo B. Internet of Things: A survey on the security of IoT frameworks. *Journal of Information Security and Applications*. 2018 Feb 1;38:8-27.
- [258] Sweeney L. Replacing personally-identifying information in medical records, the Scrub system. In *Proceedings of the AMIA annual fall symposium 1996* (p. 333). American Medical Informatics Association.
- [259] Cavoukian A. Privacy by design... Take the challenge. Information and Privacy Commissioner of Ontario.
- [260] Perrin C. The CIA triad. *Dostopno na*. 2008 Jun 30.
- [261] Sourour M, Adel B, Tarek A. Ensuring security in depth based on heterogeneous network security technologies. *International Journal of Information Security*. 2009 Aug 1;8(4):233-46.
- [262] Sah SK, Shakya S, Dhungana H. A security management for cloud based applications and services with diameter-AAA. In *2014 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT) 2014 Feb 7* (pp. 6-11). IEEE.
- [263] Alrawi O, Lever C, Antonakakis M, Monrose F. Sok: Security evaluation of home-based iot deployments. In *2019 IEEE Symposium on Security and Privacy (SP) 2019 May 19* (pp. 1362-1380). IEEE.
- [264] Fernández-Caramés TM, Fraga-Lamas P, Suárez-Albela M, Castedo L. Reverse engineering and security evaluation of commercial tags for RFID-based IoT applications. *Sensors*. 2017 Jan;17(1):28.
- [265] Cavoukian A. Privacy by design: The 7 foundational principles. Information and privacy commissioner of Ontario, Canada. 2009 Aug;5.
- [266] Metz C. AAA protocols: authentication, authorization, and accounting for the Internet. *IEEE Internet Computing*. 1999 Nov;3(6):75-9.
- [267] Integral 16GB Secure 360 Encrypted USB3.0 Flash Drive (256-bit AES Encryption) [Internet]. Available from: [https://www.amazon.com/Integral-Secure-Encrypted-256-bit-Encryption/%20dp/B00TUBOTEI/ref=sr\\_1\\_6?qid=1561614555&refinements=p\\_n\\_feature\\_keywords\\_browse-bin%3A6813186011&s=pc&sr=1-6](https://www.amazon.com/Integral-Secure-Encrypted-256-bit-Encryption/%20dp/B00TUBOTEI/ref=sr_1_6?qid=1561614555&refinements=p_n_feature_keywords_browse-bin%3A6813186011&s=pc&sr=1-6)
- [268] Harari E, Norman RD, Mehrotra S. Flash eeprom system [Internet]. US Patent. 5297148, 1994 [cited 2020 Apr 28]. Available from: <https://patentimages.storage.googleapis.com/fc/19/cb/1d06a1fcccae8/US5297148.pdf>
- [269] Chen L, Cong K, Sultana S, inventors. Side-channel attack detection using hardware performance counters. United States patent application US 16/234,085. 2019 May 2.

- [270] Upton E, Halfacree G. Raspberry Pi® User Guide [Internet]. 2016. Available from: <http://dx.doi.org/10.1002/9781119415572>
- [271] Sforzin, A., Mármol, F.G., Conti, M. and Bohli, J.M., 2016, July. RPiDS: Raspberry Pi IDS—A fruitful intrusion detection system for IoT. In 2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCCom/IoP/SmartWorld) (pp. 440-448). IEEE.
- [272] Puthal D, Mohanty SP, Nanda P, Kougianos E, Das G. Proof-of-Authentication for Scalable Blockchain in Resource-Constrained Distributed Systems. In: 2019 IEEE International Conference on Consumer Electronics (ICCE). 2019. p. 1–5.
- [273] Wayland M, Landgraf M. A cartesian coordinate robot for dispensing fruit fly food. 2018; Available from: <https://www.repository.cam.ac.uk/handle/1810/285008>
- [274] Adafruit Industries. Adafruit PiTFT - 320x240 2.8" TFT+Touchscreen for Raspberry Pi [Internet]. [cited 2020 Apr 28]. Available from: <https://www.adafruit.com/product/1601>
- [275] Guides [Internet]. SecuGen. [cited 2020 Apr 28]. Available from: <https://secugen.com/guides/>
- [276] Semiconductors NXP. MFRC522 Standard performance MIFARE and NTAG frontend. Eindhoven: NXP Semiconductors. 2016;
- [277] MIFARE Classic® 4K Contactless Smart Card [Internet]. [cited 2020 Apr 28]. Available from: <http://www.stronglink-rfid.com/en/rfid-cards/mifare-4k.html/>
- [278] Xiong Z, Wu Y, Ye C, Zhang X, Xu F. Color image chaos encryption algorithm combining CRC and nine palace map. *Multimed Tools Appl.* 2019 Nov 1;78(22):31035–55.
- [279] Fuhr T, Leurent G, Suder V. Collision attacks against CAESAR candidates. In *International Conference on the Theory and Application of Cryptology and Information Security 2015* Nov 29 (pp. 510-532). Springer, Berlin, Heidelberg.
- [280] Farrell S, Toutain L, Yegin A, Ratilainen A, Anaya JC, Ponsard B, Crowcroft J, Gomez C, Heile B, Minaburo A, Paradells J. Low-power wide area network (lpwan) overview.
- [281] Chen TP, Yau WY, Jiang X. ISO/IEC standards for on-card biometric comparison. *International Journal of Biometrics.* 2013 Jan 1;5(1):30-52.

- [282] De Hert P, Papakonstantinou V. The new General Data Protection Regulation: Still a sound system for the protection of individuals?. *Computer law & security review*. 2016 Apr 1;32(2):179-94.
- [283] Hamster Pro 10 [Internet]. SecuGen. [cited 2020 Apr 28]. Available from: <https://secugen.com/products/hamster-pro-10/>
- [284] Standard performance MIFARE® and NTAG® frontend | NXP [Internet]. [cited 2020 Apr 28]. Available from: <https://www.nxp.com/products/rfid-nfc/nfc-hf/nfc-readers/standard-performance-mifare-and-ntag-frontend:MFRC52202HN1>
- [285] Maitra S. Chosen IV cryptanalysis on reduced round ChaCha and Salsa. *Discrete Applied Mathematics*. 2016 Jul 31;208:88-97.
- [286] Fluhrer SR. Cryptanalysis of ring-LWE based key exchange with key share reuse. *IACR Cryptology ePrint Archive*. 2016 Jan 30;2016:85.
- [287] Yang X, Xu C, Li C. A privacy model for RFID tag ownership transfer. *Security and Communication Networks*. 2017;2017.
- [288] Munilla J, Burmester M, Peinado A. Attacks on ownership transfer scheme for multi-tag multi-owner passive RFID environments. *Computer Communications*. 2016 Aug 15;88:84-8.
- [289] Brooks M, Yang B. A Man-in-the-Middle attack against OpenDayLight SDN controller. In *Proceedings of the 4th Annual ACM Conference on Research in Information Technology 2015 Sep 29* (pp. 45-49).
- [290] Zhang T, Zhang Y, Lee RB. Cloudradar: A real-time side-channel attack detection system in clouds. In *International Symposium on Research in Attacks, Intrusions, and Defenses 2016 Sep 19* (pp. 118-140). Springer, Cham.
- [291] Chen C, Eisenbarth T, Von Maurich I, Steinwandt R. Differential power analysis of a McEliece cryptosystem. In *International Conference on Applied Cryptography and Network Security 2015 Jun 2* (pp. 538-556). Springer, Cham.
- [292] Kasper T, Oswald D, Paar C. New methods for cost-effective side-channel attacks on cryptographic RFIDs. In *Workshop on RFID Security 2009 Jun*.
- [293] Share a secret - One Time [Internet]. [cited 2020 Apr 28]. Available from: <https://onetimesecret.com/>
- [294] Singh A. Centralized key distribution using quantum cryptography. *Int. J. Comput. Sci. Mobile Comput.*. 2017 Jul;6(7):208-13.
- [295] Kuppusamy TK, DeLong LA, Cappos J. Uptane: Security and customizability of software updates for vehicles. *IEEE Vehicular Technology Magazine*. 2018 Feb 1;13(1):66-73.

- [296] Krull CR, McMillan LF, Fewster RM, van der Ree R, Pech R, Dennis T, Stanley MC. Testing the feasibility of wireless sensor networks and the use of radio signal strength indicator to track the movements of wild animals. *Wildlife research*. 2019 Jan 15;45(8):659-67.
- [297] Alamer A, Soh B. A new neural-network-based model for measuring the strength of a pseudorandom binary sequence. *Int J Adv Eng Sci Appl Math*. 2020 Apr;7(4):29–38.
- [298] Uğuz M, Doğanaksoy A, Sulak F, Koçak O. R-2 composition tests: a family of statistical randomness tests for a collection of binary sequences. *Cryptography and Communications*. 2019 Sep 15;11(5):921-49.
- [299] Wang Y, Nicol T. On statistical distance based testing of pseudo random sequences and experiments with PHP and Debian OpenSSL. *Computers & Security*. 2015 Sep 1;53:44-64.
- [300] Liu J, Mesnager S. Weightwise perfectly balanced functions with high weightwise nonlinearity profile. *Designs, Codes and Cryptography*. 2019 Aug 15;87(8):1797-813.
- [301] Tang D, Liu J. A family of weightwise (almost) perfectly balanced boolean functions with optimal algebraic immunity. *Cryptography and Communications*. 2019 Nov 1;11(6):1185-97.
- [302] Sun Z, Winterhof A. On the maximum order complexity of the Thue-Morse and Rudin-Shapiro sequence. *arXiv preprint arXiv:1910.13723*. 2019 Oct 30.
- [303] Biryukov A. The design of a stream cipher LEX. In *International Workshop on Selected Areas in Cryptography 2006 Aug 17* (pp. 67-75). Springer, Berlin, Heidelberg.
- [304] Maitra S, Yelamarthi K. Rapidly Deployable IoT Architecture with Data Security: Implementation and Experimental Evaluation. *Sensors*. 2019 Jan;19(11):2484.
- [305] Rangra A, Sehgal VK, Shukla S. A Novel Approach of Cloud Based Scheduling Using Deep-Learning Approach in E-Commerce Domain. *International Journal of Information System Modeling and Design (IJISMD)*. 2019 Jul 1;10(3):59-75.
- [306] Chakraborty RS, Mathew J, Vasilakos A, editors. *Security and fault tolerance in internet of things*. Springer; 2019.
- [307] Randhawa RH, Hameed A, Mian AN. Energy efficient cross-layer approach for object security of CoAP for IoT devices. *Ad Hoc Networks*. 2019 Sep 1;92:101761.
- [308] Noor TH, Zeadally S, Alfazi A, Sheng QZ. Mobile cloud computing: Challenges and future research directions. *Journal of Network and Computer Applications*. 2018 Aug 1;115:70-85.

[309] Oberg JK, Valamehr J, Kastner R, Sherwood T, inventors; Tortuga Logic Inc, assignee. Generating hardware security logic. United States patent US 10,289,873. 2019 May 14.

[310] Kaeslin H. Digital integrated circuit design: from VLSI architectures to CMOS fabrication. Cambridge University Press; 2008 Apr 28.