PinTar: MESSAGE AUTHENTICATION CODE BASED ON QUASIGROUP
AND PERMUTATION COMPRESSION FUNCTION

ZAHRADDEEN ABUBAKAR PINDAR

A thesis submitted in fulfillment
of the requirement for the award of the
Degree of Doctor of Philosophy

Faculty of Computer Science and Information Technology
Universiti Tun Hussein Onn Malaysia

JUNE 2017
I dedicate this work to my mother Iya Fatima Umar, may her soul rest in peace and to my father Mallam Garba Pindar.
ACKNOWLEDGMENT

In the name of Allah, Most Gracious, Most Merciful. All Praise be to Allah, the Cherisher and Sustainer of the worlds. Oh Allah, send Grace and Honour on Prophet Muhammad (S.A.W) and on the family and true followers of Prophet Muhammad (PBUH).

I would like to express my appreciation and gratitude to my down to earth supervisor Associate Professor Sapiee Jamel for his motivation and support through securing me the funding to pursue my Ph.D. I would also like to thank him for the various trips he has taken me across Malaysia and also the fishing and scuba diving lessons. I am indeed very grateful. I would also like to thank my family for their support and motivation through the difficult times during my Ph.D. journey. And to the faculty members especially Faculty of Computer Science and Information Technology (FSKTM) postgraduate student’s society, I say thank you. Thank you for the good company and friendship you have provided me during my Ph.D. journey.

Finally, I would like to thank the Office for Research, Innovation, Commercialization and Consultancy (ORICC), Universiti Tun Hussein Onn Malaysia for providing the funds to conduct my Ph.D. research.
ABSTRACT

In the area of Information Security, Message Authentication Codes (MACs) are used to verify the integrity of messages that are sent or received over an insecure channel. Traditionally, MAC algorithms are constructed by using existing block ciphers as underlying compression function due to the challenges and difficulties faced in constructing new MACs from the scratch. However, block ciphers requires key scheduling for encryption and this process impacts the performance of the MAC. As a result, permutation compression functions based on permutations and XOR were proposed as alternatives to block ciphers. Consequently, these constructions of have been shown certain limitations. This research proposes an efficient and secure compression function based on the theory of quasigroup and permutation, as a remedy for the draw backs of permutation based compression functions. In addition, a new variable length MAC (128 bits, 256 bits and 512 bits) has been proposed based on the proposed compression function. The security of the proposed compression function and MAC algorithm was analyzed based on statistical analysis. Average correlation assessment of 0.094 was obtained for the proposed compression function. Randomness of compression function was analyzed and average $P-value$ of 0.4354 was obtained for randomness test using NIST statistical test tool. Furthermore, PinTar MAC algorithm is shown to have a key space of $2^{19}$ factorial ($2^{19}$!), which makes the MAC highly resistant against key exhaustive search attack, compared to DES and AES block ciphers which have key space of $2^{56}$ and $2^{128}$ respectively. Avalanche result of 97%, 94%, 94% and 88% was obtained for 128 bits of PinTar MAC, MD5, MD4 and MD2 respectively. PinTar MAC has also been shown to be resistant against adaptive chosen text attack due to the very large key space. Finally, it can be concluded that PinTar MAC and its compression function are efficient, and can serve as alternative algorithms in designing security systems.
ABSTRAK

Di dalam bidang keselamatan maklumat, Kod Mesej Pengesahan (MAC) telah digunakan untuk mengesahkan integriti mesej yang dihantar atau diterima melalui saluran yang tidak dijamin keselamatannya. Secara tradisinya, algoritma MAC telah dibina dengan menggunakan blok sifer yang telah tersedia sebagai fungsi mampatan asas kerana terdapat permasalahan dan kesukaran yang terpaksa dihadapi dalam pembinaan MAC yang baru daripada mula. Walau bagaimanapun, blok sifer memerlukan kunci penjadualan untuk penyulitan dan proses ini memberi kesan kepada prestasi MAC. Hasilnya, fungsi pilihan antar mampatan dan XOR telah dicadangkan sebagai alternatif kepada blok sifer. Namun, pembinaan ini mempunyai kelemahan tertentu. Kajian ini telah mencadangkan fungsi rawak berasaskan fungsi mampatan yang lebih cekek dan selamat sebagai penyelesaian kepada kekurangan pilihan antar berasaskan fungsi mampatan. Selain itu, pembolehubah MAC yang baru (128 bit, 256 bit dan 512 bit) telah dicadangkan berasaskan fungsi mampatan yang telah dicadangkan. Keselamatan fungsi mampatan dan algoritma MAC yang dicadangkan dianalisis berdasarkan analisis secara statistik. Penilaian purata korelasi sebanyak 0.094 telah diperolehi bagi fungsi mampatan yang telah dicadangkan. Kerawakan terhadap fungsi mampatan telah dianalisis dan purata $P-value$ sebanyak 0.4354 telah diperolehi melalui ujian kerawakan dengan menggunakan alat ujian statistik NIST. Begitu juga, algoritma PinTar MAC telah menunjukkan mempunyai ruang kunci sebanyak $2^{19}$ factorial ($2^{19!}$) yang telah menjadikan MAC tahan kepada serangan kunci carian yang menyeluruh berbanding blok sifer AES dan DES. Keputusan avalanche sebanyak 97%, 94%, 94% dan 88% telah diperolehi untuk 128 bit PinTar MAC, MD5, MD4 dan MD2. PinTar MAC juga telah menunjukkan ketahanannya terhadap serangan teks yang dipilih secara penyesuaian kerana ia mempunyai ruang kunci yang sangat besar. Akhirnya, dapat disimpulkan bahawa MAC dan fungsi mampatan yang dicadangkan sangat cekek dan berupaya untuk dijadikan sebagai algoritma alternatif kepada rekabentuk sistem keselamatan.
CONTENTS

TITLE i
DECLARATION ii
DEDICATION iii
ACKNOWLEDGMENT iv
ABSTRACT v
ABSTRAK vi
LIST OF TABLES xii
LIST OF FIGURES xiv
LIST OF APPENDICES xv
LIST OF SYMBOLS AND ABBREVIATIONS xvi
LIST OF PUBLICATIONS xvii

CHAPTER 1 INTRODUCTION 1
1.1 Background of Study 1
1.2 Background of the Problem 3
1.3 Problem Statement 4
1.4 Objectives 5
1.5 Contributions of Study 5
1.6 Significance of Study 7
1.7 Scope 8
1.8 Thesis Organization 9
CHAPTER 2 LITERATURE REVIEW

2.1 Introduction

2.2 Preliminaries
   2.2.1 Binary Operations
   2.2.2 Groups
      2.2.2.1 Abelian and Non-Abelian Groups
   2.2.3 Permutations
      2.2.3.1 Generating Permutations
      2.2.3.2 Properties of Permutations
   2.2.4 Quasigroups (Latin squares)
      2.2.4.1 Generating Quasigroups
   2.2.5 Quasigroups String Transformation

2.3 Overview of Message Authentication Code (MAC)

2.4 Properties of MAC

2.5 Overview of Design Components for MAC
   2.5.1 Initial Vector (IV)
   2.5.2 Message Pre-processing and Generating Hash Digest
   2.5.3 Key Scheduling
   2.5.4 Compression Function

2.6 Methods of Constructing MAC
   2.6.1 Block Cipher as Compression Function for MAC
      2.6.1.1 Cipher-based MAC (CMAC)
      2.6.1.2 Parallelizable MAC (PMAC)
   2.6.2 Hash Function as Compression Function for MAC
      2.6.2.1 Merkle-Damgard Construction Scheme
      2.6.2.2 Hash Function based MAC (HMAC)
   2.6.3 MAC based on Algebraic Structures
      2.6.3.1 Denes and Keedwell MAC Scheme
      2.6.3.2 Meyer MAC Scheme (QMAC)

2.7 Overview of Cryptographic Hash Functions
   2.7.1 Whirlpool Hash Function
   2.7.2 MDx Family Hash Function
2.7.3 SHAx Family Hash Function 43
2.8 Permutation Based Compression Functions 44
  2.8.1 Rogaway Compression Function 45
  2.8.2 Mennink Compression Function 46
  2.8.3 Comparison of Compression Function based on Permutation 48
2.9 Summary of Chapter 49

CHAPTER 3 RESEARCH METHODOLOGY 50
  3.1 Introduction 50
  3.2 Research Framework 50
    3.2.1 Literature Review 51
    3.2.2 Identification of Research Problem 52
    3.2.3 Proposed Methodology 52
    3.2.4 Security Analysis Compression Function 53
      3.2.4.1 Correlation Assessment 53
      3.2.4.2 Randomness Test 54
      3.2.4.3 Collision Resistance 55
      3.2.4.4 Pre-image Resistance 56
    3.2.5 Security Analysis of MAC Algorithm 58
      3.2.5.1 Brute force Attack (Key Exhaustive Search) 58
      3.2.5.2 Avalanche Property 59
    3.2.6 Result and Discussion 61
    3.2.7 Summary of Chapter 61

CHAPTER 4 DESIGN OF PinTar MAC ALGORITHM 63
  4.1 Introduction 63
  4.2 Overview of MAC Algorithm 64
  4.3 Design of PinTar MAC Algorithm 65
    4.3.1 Initialization Vector (IV) 67
    4.3.2 Key Generation 67
    4.3.3 Message Pre-processing 69
    4.3.4 Proposed 2n-to-n bit Compression Function (CF) 70
4.3.4.1 Pseudorandom Function \((T_F)\) 70
4.3.5 Hash Digest Reduction Function 75
4.4 Simulation of Proposed MAC Algorithm 76
4.5 Summary of Chapter 81

CHAPTER 5 SECURITY ANALYSIS OF \textit{PinTar} MAC ALGORITHM 83
5.1 Introduction 83
5.2 Security Analysis of \(2n - to - n\) bit Compression Function 84
  5.2.1 Correlation Test for \(2n - to - n\) bit Compression Function 84
  5.2.2 Randomness Test for \(2n - to - n\) bit Compression Function 85
  5.2.3 Collision Resistance of \(2n - to - n\) bit Compression Function 86
  5.2.4 Pre-image Resistance of \(2n - to - n\) bit Compression Function 87
5.3 Security Analysis of \textit{PinTar} MAC Algorithm 88
  5.3.1 Brute force Attack (Key Exhaustive Search Attack) 88
  5.3.2 Avalanche Property 89
    5.3.2.1 Avalanche Test of Four 128 bits Hash Functions 90
    5.3.2.2 Avalanche Test of Four 256 bits Hash Functions 91
    5.3.2.3 Avalanche Test of Four 512 bits Hash Functions 92
5.4 Summary of Chapter 93

CHAPTER 6 CONCLUSION AND FUTURE WORK 95
6.1 Introduction 95
6.2 Research Contributions 96
6.3 Future work 97

REFERENCES 99
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Example of Abelian Group</td>
<td>15</td>
</tr>
<tr>
<td>2.2</td>
<td>Example of Non-abelian Group</td>
<td>16</td>
</tr>
<tr>
<td>2.3</td>
<td>Number of Non-commutative Quasigroups (Damm, 2003)</td>
<td>17</td>
</tr>
<tr>
<td>2.4</td>
<td>Generating Quasigroup of order 6 based on Permutations</td>
<td>21</td>
</tr>
<tr>
<td>2.5</td>
<td>Result from Table 2.4</td>
<td>21</td>
</tr>
<tr>
<td>2.6</td>
<td>Generating Quasigroup of order 7 using Linear Mapping</td>
<td>23</td>
</tr>
<tr>
<td>2.7</td>
<td>Quasigroup of order 7</td>
<td>24</td>
</tr>
<tr>
<td>2.8</td>
<td>Hash Function and their Initial Vectors (Rivest, 1991; NIST, 2012)</td>
<td>28</td>
</tr>
<tr>
<td>2.9</td>
<td>Comparison of Hash Function variants and Block, Message and Digest size (Rivest, 1991; NIST, 2012; FIPS, 2015)</td>
<td>28</td>
</tr>
<tr>
<td>2.10</td>
<td>Quasigroup of order 5</td>
<td>38</td>
</tr>
<tr>
<td>2.11</td>
<td>Quasigroup of order 7</td>
<td>40</td>
</tr>
<tr>
<td>2.12</td>
<td>Comparison of Single-setting Permutation based Compression Functions</td>
<td>48</td>
</tr>
<tr>
<td>2.13</td>
<td>Comparison of Multiple-setting Permutation based Compression Functions</td>
<td>48</td>
</tr>
<tr>
<td>3.1</td>
<td>Collision Attack Scheme</td>
<td>56</td>
</tr>
<tr>
<td>3.2</td>
<td>Pre-image Attack Scheme</td>
<td>58</td>
</tr>
<tr>
<td>3.3</td>
<td>Key Exhaustive Search (Stallings, 2006)</td>
<td>59</td>
</tr>
<tr>
<td>4.1</td>
<td>SHA2 512 bits Initial Vector</td>
<td>67</td>
</tr>
<tr>
<td>4.2</td>
<td>Permutation of order 256</td>
<td>68</td>
</tr>
<tr>
<td>4.3</td>
<td>Randomly Generated Permutations $\pi(x)$ and $g(x)$</td>
<td>72</td>
</tr>
<tr>
<td>4.4</td>
<td>Computing two inputs in Transformation $T_F$</td>
<td>73</td>
</tr>
<tr>
<td>4.5</td>
<td>Legends of Figure 4.6</td>
<td>74</td>
</tr>
<tr>
<td>4.6</td>
<td>Algorithm of PinTar MAC</td>
<td>77</td>
</tr>
</tbody>
</table>
5.1 Security Analysis on Security Strength of Proposed $2n - to - n$ bit Compression Function and PinTar MAC

5.2 Correlation Assessment Test for 16 bits, 32 bits and 64 bits

5.3 $P$-value Frequency, Block Frequency and Runs Test for 16, 32 and 64 bits Inputs

5.4 $P$-value Frequency, Block Frequency and Runs Test for 128, 256 and 512 bits Inputs

5.5 Collision Resistance of Proposed $2n - to - n$ bit Compression Functions

5.6 Pre-image Resistance of Proposed $2n - to - n$ bit Compression Functions

5.7 Comparison of Key Space

5.8 Avalanche Property of PinTar 128 bits MAC

5.9 Comparison of four 128 bits Hash Functions

5.10 Avalanche Result of PinTar 256 bits MAC

5.11 Comparison of four 256 bits Hash Functions

5.12 Avalanche Result of PinTar 512 bits MAC

5.13 Comparison of four 512 bits Hash Functions
LIST OF FIGURES

2.1 Taxonomy of Literature Review 11
2.2 Communication between Alice and Bob using Message Authentication Code (Bellare & Rogaway, 2005) 26
2.3 Taxonomy of Cryptographic Hash Functions 31
2.4 Illustration of the two cases of MAC Generation using CMAC (Dworkin, 2005) 32
2.5 Parallelizable MAC (Black, 2002) 33
2.6 Function for Merkle–Damgård construction scheme 34
2.7 Merkle–Damgård construction scheme 35
2.8 Algorithm of Rogaway and Steinberger $2^n - to - n$ bit Compression Function (Rogaway & Steinberger, 2008a) 46
2.9 Framework of Rogaway and Steinberger $2^n - to - n$ bit Compression Function (Rogaway & Steinberger, 2008a) 46
2.10 Framework of Mennink and Preneel $2^n - to - n$ bit Compression Function (Mennink & Preneel, 2012) 47
3.1 Research Framework 51
4.1 Taxonomy of Cryptographic Hash Functions showing Research Contribution 63
4.2 Signing Message Using PinTar MAC 64
4.3 Verifying Message Using PinTar MAC 65
4.4 Proposed PinTar MAC Algorithm 66
4.5 Algorithm of $2^n - to - n$ bit Compression Function 74
4.6 Framework of $2^n - to - n$ bit Compression Function 75
# LIST OF APPENDICES

<table>
<thead>
<tr>
<th>Appendix</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Test Data for Randomness Test and Correlation Analysis</td>
<td>108</td>
</tr>
<tr>
<td>B</td>
<td>Avalanche Results for Cryptographic Hash Functions</td>
<td>113</td>
</tr>
<tr>
<td>C</td>
<td>Correlation Assessment Program</td>
<td>117</td>
</tr>
<tr>
<td>D</td>
<td>Implementation of <em>PinTar</em> MAC Algorithm</td>
<td>119</td>
</tr>
</tbody>
</table>
LIST OF SYMBOLS AND ABBREVIATIONS

* Binary operation
⊕ Exclusive OR operation (XOR)
PinTar Variable MAC Algorithm (128 bits, 256 bits, 512 bits)
AES Advanced Encryption Standard (The Rijndael Encryption Algorithm)
DES Data Encryption Standard
HiSea The Hybrid Cube Encryption Algorithm
KECCAK The SHA 3 Hash Algorithm
TWOFISH The Twofish Encryption Algorithm
∀ a, b For all a and b
π (x) Permutation of x
a, b ∈ S Elements a, b in set S
LIST OF PUBLICATIONS

Journal:


Proceeding:


CHAPTER 1

INTRODUCTION

1.1 Background of Study

Integrity is one of the most fundamental properties of information security. Intentional or unintentional unauthorized changes to information usually violate its integrity (Pindar, 2014). Cryptographic hash functions are mechanisms used to verify the integrity of information by detecting whether or not the information has undergone modification. A cryptographic hash function takes in information as input and processes it using mathematical transformations to produce an output known as the message digests. Protection is achieved by creating a unique relationship between the information (input) and the message digest (output), such that it is difficult to find the same message digest for distinct inputs (Bellare et al., 1996; Bellare & Rogaway, 2005).

Cryptographic hash functions can be grouped into unkeyed and keyed function. Unkeyed hash functions do not require keys to generate message digests. They are also known as hash functions. These functions can also be further categorized into two: dedicated hash functions and block cipher based hash functions. Dedicated hash functions dominate the development of existing hash functions. They are designed specifically for hashing purposes. Examples of dedicated hash functions are Secure Hash Algorithm (SHA) family (Hansen, 2001; FIPS, 2015) and Message Digest Algorithm (MD) family (Rivest, 1991). On the other hand, block cipher based hash functions are designed to use block ciphers as underlying compression function. Example of block cipher based hash function is Whirlpool hash function (Barreto
et al., 2000), which utilizes the Advanced Encryption Standard (AES) block cipher as its underlying compression function. However, this category of hash functions is less effective than dedicated hash function (Black et al., 2005).

Keyed hash functions, also known as Message Authentication Codes (MACs), are constructed on the bases of existing hash functions, and they require keys to generate the message digests. This construction method is also called Hash MAC (HMAC) (NIST, 2008). In this case, the underlying hash function can be a dedicated hash function or block cipher based hash function and the strength of the MAC is derived from the security of the underlying hash function. Another method of constructing MAC is from the scratch (Aumasson & Bernstein, 2012; Mouha et al., 2015). This method does not require the use of a hash function as the underlying compression function, rather it is based on arithmetic and algebraic operations. MACs developed by Keedwell (1992) and Meyer (2006) are typical example of MAC that do not require hash functions as underlying compression function. In practice, the output of a keyed hash function or MAC must be easy to compute with knowledge of the secret key, but difficult to compute without the knowledge of the key. It is required that the number of potential keys must be large enough to prevent key exhaustive search attacks on the MAC (Meyer, 2006).

Hash functions have been used as primitives for constructing MACs (NIST, 2008), pseudo random number generators, key derivation functions and also in designing systems which provide security for transfer of confidential information over an insecure channel (Hani et al., 2006). This arguably suggests the need for continuous development of secure and efficient hash functions. Similarly, the need for a secure cryptographic hash algorithm was further motivated by the weaknesses found in the widely used MD4, MD5, SHA-0 and SHA-1 hash algorithms (Dobbertin, 1998; Wang et al., 2005). In 2007, the United States National Institute of Standards and Technology (NIST) called for a public competition to develop a publicly disclosed hash algorithm (SHA-3) that is capable of protecting sensitive information for decades (NIST, 2007). In 2015, KECCAK hash algorithm was selected as the new SHA-3 hash algorithm to replace MD4, MD5, SHA-0, SHA-1 and SHA 2 (FIPS, 2015). One of the advantages of publicly disclosing the algorithm is to overcome the problem of obscurity through secrecy. Also, the underlying mathematical properties
of the algorithms can be properly examined by cryptographers to determine their correctness, strength and efficiency (Naor & Yung, 1989).

1.2 Background of the Problem

Constructing a secure and efficient MAC algorithms is a difficult and challenging task and for this reason, many MAC constructions uses existing block ciphers as their underlying compression function (Lai & Massey, 1993; Preneel et al., 1994; Xun & Kwok, 1997; Rogaway, 2004; Dworkin, 2005; Black et al., 2005; Rogaway & Steinberger, 2008a). The advantage of this method is that the MAC algorithm does not have to be constructed from scratch. Therefore, the security of the MAC algorithm is dependent on the security of the underlying block cipher.

However, this method of construction has certain disadvantages. A broken block cipher will result to the insecurity of the entire MAC algorithm. Similarly, many block ciphers such as Advanced Encryption Standard (AES) and Data Encryption Standard (DES) are covered by patents and subject to export control (NIST, 2001). This sets a limit to the extent of using the algorithms especially when application of block cipher involves modifying the components of the algorithm. Accordingly, block ciphers use key scheduling algorithms to generate the keys for encryption and decryption processes. This procedure of generating keys significantly affects the performance of the MAC algorithm (Black et al., 2005). For instance, the HiSea block cipher is known to incur computational cost during key scheduling and encryption procedure (Jamel et al., 2011b). In addition, Black et al. (2005) showed that a compression function that is based on fixed keys also impacts the performance of the entire MAC algorithm. As a result, Black et al. (2005) proposed the idea of constructing a simple compression function, using fewer components of a block cipher such as permutations and Boolean operators. The compression function should be a $2n - to - n$ bit compression function. That is, it should take in two inputs ($2n$ bits) and map them to single output ($n$ bit).
1.3 Problem Statement

Rogaway & Steinberger (2008a) proposed a $2n \rightarrow n$ compression function based on three (3) random permutations with arithmetic operation in finite field of $\mathbb{F}_2^{128}$. Security analysis of their work shows that the compression function is only secure against collision and pre-image attack if the permutations are organized in a multi-permutation setting\(^1\). While using single-permutation setting\(^2\) would reduce the increase the simplicity of a compression function by reducing the computation cost of generation different permutations, their work shows that a single-permutation setting is not resistant against collision and pre-image attacks. In addition, their proposed constructions of have shown to have ineffective pseudorandom property and also does not support keys, which them not very suitable for compression functions for a MAC algorithm.

Another $2n \rightarrow n$ bit compression function was proposed by Mennink & Preneel (2012) based on three (3) random permutations with XOR operator. The improvement of their compression function is due to the adoption of XOR operator over finite field of $\mathbb{F}_2^{128}$. This is because XOR operation is an already pre-defined function is computer systems, while functions for finite fields have to be defined before application. The adoption of Exclusive OR (XOR) operator in their construction has reduced the computational cost of defining and using functions such as finite fields. Mennink & Preneel's compression function requires that the permutations must be organized in a multi-permutation setting in order to be secure against collision and pre-image attack. Another weakness is that their compression function is shown to have ineffective pseudorandom property.

This research proposes a secure and efficient $2n \rightarrow n$ compression function using a new primitive that is based on a single-setting, which supports large keys and has efficient pseudorandom property. The single-setting construction adopted in this research is based on the concept of using three (3) pseudorandom functions ($T_{F1}$, $T_{F2}$, $T_{F3}$), such that all three functions are equal ($T_{F1} = T_{F2} = T_{F3}$). Using three equal pseudorandom functions simplifies the construction of the compression function, by

\(^1\) The multi permutation setting is when all the three permutations are different as ($\pi_1 \neq \pi_2 \neq \pi_3$).
\(^2\) The single permutation setting is when all three permutations are the same ($\pi_1 = \pi_2 = \pi_3$)
eliminating the challenges and computational cost of constructing three different pseudorandom functions where all the functions are different \( T_{F_1} \neq T_{F_2} \neq T_{F_3} \). The pseudorandom function is constructed based on the theory of quasigroups and permutation. This is due to reason that quasigroups have been shown to be very efficient in development of encryption algorithms (Battey & Parakh, 2013) and in error detection codes (Belyavskaya et al., 2003). This research aims to utilize efficiency of permutations and quasigroups in the development of a secure and efficient \( 2n \rightarrow to \rightarrow n \) compression function for a MAC algorithm.

1.4 Objectives

The objective of this research is to achieve the followings:

(i) To propose a \( 2n \rightarrow to \rightarrow n \) bit compression function based on pseudorandom primitive constructed using permutations and quasigroups.

(ii) To propose a MAC based on the \( 2n \rightarrow to \rightarrow n \) bit compression function proposed in (i).

(iii) To analyze the strength of the compression function using statistical analysis such as correlation assessment, randomness test, collision and pre-image resistance. Similarly, the proposed MAC will be analyzed using brute force attack and avalanche property.

1.5 Contributions of Study

The major objective of the research presented in this thesis is the development of a secure MAC algorithm (PinTar MAC). However, there are several security challenges involved in the construction of compression functions for MAC algorithms as identified in the problem statement. Some of these challenges included simplifying the construction of the compression function by using few components as possible. The simplified construction must be secure against collision and pre-image attacks. The construction must be able to support strong cryptographic keys, and lastly it
should have very efficient pseudorandom property. The contributions of this research are directed towards addressing the mentioned challenges. Three contributions have been presented in the thesis.

The first research contribution involves using a different underlying primitive for the proposed $2n - to - n$ compression function. This was done in order to address the issue of weak random property and key support which is found commonly found in the existing $2n - to - n$ compression functions. The underlying primitive proposed in this research is a pseudorandom function ($T_F$), which is constructed based on the theory of very large quasigroups of order $256$ ($2^{19}$ bits) and permutations of order $256$ ($2^8$ bits). The pseudorandom function ($T_F$) primitive operates in a way such that very similar inputs will have significantly different outputs. Theoretical analysis of the pseudorandom primitive in Section 4.3.4.1 has shown to be very effective. Very large quasigroups of order $256$ ($2^{19}$ bits) are adopted in the design as keys, to address the concern of key support. In comparison to the key space found in standard block cipher such as the AES in Section 5.3.1, quasigroups of order $256$ have been shown to have a significantly larger key space of key of $2^{19}$ factorial ($2^{19!}$). The key space of very large quasigroups makes them very resistant against key exhaustive search. This makes very large quasigroups as good candidate for cryptographic keys.

The second contribution is the development of a $2n - to - n$ bit compression function, which utilizes the pseudorandom function ($T_F$) as its underlying primitive. As mentioned in the problem statement, earlier constructions of the $2n - to - n$ bit compression function are based on permutations, XOR operators or finite fields of $F^{128}_2$. The existing $2n - to - n$ compression functions are required to have a multi-permutation setting ($\pi_1 \neq \pi_2 \neq \pi_3$) in order to be secure against collision and pre-image attack. On the contrary, this thesis proposes a compression function which uses a single-setting pseudorandom function ($T_{F1} = T_{F2} = T_{F3}$). Single-setting are more advantageous over multi-setting, due to the reason that single-setting saves computational cost of defining different functions. Analysis of the proposed $2n - to - n$ bit compression function shows that the proposed function is secure against collision and pre-image attack, and it is shown to have very effective pseudorandom property.

The third contribution discussed in this thesis involves integrating the proposed
2n-to-n bit compression function into the framework of a MAC algorithm (PinTar MAC). PinTar MAC is designed to generate hash digest of 128 bits. Hash digests of 256 bits and 512 bits can also be generated for security applications where longer hash digests are required. PinTar MAC uses very large quasigroups of order 256 as keys (2^{19} bits). PinTar MAC has been analysed based on its security against key exhaustive search attack and the avalanche test. The results shows that the proposed PinTar MAC is efficient and can be used as alternative for other cryptographic hash functions.

1.6 Significance of Study

Encryption algorithms plays a significant role in protecting confidentiality of information from unauthorized individuals. For instance when two parties (Alice and Bob) communicate over an insecure channel using a secure encryption algorithm such as Advanced Encryption Standard (AES), the confidentiality of their communication is protected against intruders and eavesdroppers (Eve). Consequently, encryption algorithms do not protect the integrity of the information transmitted in the medium (Bellare & Rogaway, 2005). Hence, eavesdroppers such as Eve can modify the content of the information, and the integrity of the received information cannot be easily verified. Therefore, this necessitates the need to find alternative solutions to determine the integrity of information that is sent or received over an insecure channel. This lead to the development of cryptographic hash function, particularly MAC algorithms (Bellare & Rogaway, 2005; Goldwasser & Bellare, 2008). As such, when Alice sends a message to Bob through an insecure channel, Bob can verify the authenticity of the message by using a MAC algorithm.

Constructing a secure and efficient MAC algorithms is a difficult and challenging task and for this reason, many MAC constructions use existing encryption algorithms as their underlying compression function (Black, 2002; Rogaway, 2004; Dworkin, 2005). The advantage of this method of constructions is that the MAC algorithm does not have to be constructed from scratch. The MAC algorithm enjoys the security of the underlying encryption algorithm. If the security of the underlying encryption algorithm is broken, the entire MAC can no longer be secure. Also, many encryption algorithms such as the popularly known Advanced Encryption Standard
(AES) and Data Encryption Standard (DES) are covered by patents and subject to export control. This sets a limit to the extent of using the algorithms especially when application of encryption algorithm involves modifying the components of the algorithm. Encryption algorithms use key generation algorithms to generate keys that will be used in the encryption process. This procedure of generating keys significantly affects the performance of the MAC algorithm. Based on the identified issues, researchers are compelled to construct secure and efficient compression functions for MACs algorithms that are based on few components of the encryption algorithm such as permutations and other Boolean operators (Rogaway & Steinberger, 2008a; Shrimpton & Stam, 2008; Mennink & Preneel, 2012).

This research draws its significance from other notable researches that have been proposed in the literature by constructing efficient compression functions for MACs and hash functions. This research shows that algebraic structures such as quasigroups can be used to construct new and secure compression functions for cryptographic algorithms. The methodology of constructing the new compression function and entire MAC algorithm can be viewed as a new contribution to the pool of existing methods of constructing cryptographic hash functions.

1.7 Scope

The scope of this research focuses on the development of a new $2n - 10 - n$ bit compression function, which is used as underlying compression function for a MAC algorithm. The security of the compression function and MAC algorithm is analyzed using statistical methods of analysis. The attack schemes adopted in the analysis are based on heuristic and generic attack models for analyzing compression functions and MAC algorithms. The proposed MAC will serve as an alternative cryptographic function to existing keyed hash functions and MACs.
1.8 Thesis Organization

The organization of this thesis is as follows:

Chapter 1 of this thesis is categorized into six sections. The first section discusses the introduction and background study of Message Authentication Codes (MACs). The current research issues and challenges that are related to the development of underlying cryptographic primitives for MACs are presented in the problem statement in section two. The third section discusses some key points that motivated us to conduct this research. The fourth section presents the contributions and objectives this research aims to achieve. The scope of the research is illustrated in section five.

Chapter 2 of this thesis is divided into seven sections. The first section describes some related mathematical principles, terms and preliminaries that are relevant in the literature. The second section describes a general overview of MAC. The third section outlines some general properties of a secured MAC. The design components of MAC are discussed in the fourth section. Similarly, some prominent methods of constructing secured and efficient MACs are identified in section five. Accordingly, some relevant methods of analyzing the security strength of MAC and its underlying compression function are discussed in sixth section. Summary of Chapter 2 is highlighted in last section.

Chapter 3 of this thesis is divided into four sections. The first section discusses the introduction of our proposed MAC. The second section discusses in much detail the methodology of developing the proposed MAC and compression function. The third section describes a simulation of our MAC. The fourth section gives the concluding summary of the chapter.

Chapter 4 of this thesis describes the development and implementation of compression function based on permutations, quasigroups and PinTar MAC.

Chapter 5 of this thesis is divided into three sections. The first section discusses results and security analysis of proposed compression function in comparison with other existing compression functions. The second section discusses the results and security analysis of entire proposed MAC in comparison with other existing cryptographic hash functions. The third section summarises the issues
discussed in the earlier sections.

Chapter 6 discusses the contributions of the research in the area of information security and also addresses the future directions of the research.
CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

MAC algorithms are required to be constructed using strong mathematical primitives. This chapter discusses some these mathematical principles that are relevant in the contructions of PinTar MAC algorithm. Given that MAC algorithms can be contructed using from the scratch, existing block ciphers and hash functions, some related works in the contruction of MAC algorithms using the different approaches have also been discussed. Figure 2.1 illustrates the taxonomy of the literature review.

![Figure 2.1: Taxonomy of Literature Review](image)

The preliminaries presented in Section 2.2 discusses the mathematical principles which are relevant in the contructions of MAC algorithms, hash functions and block ciphers. These principles includes binary operations, groups, permutations and quasigroups. The section also discusses applications and advantages adopting
permutations and quasigroups in designing secure MAC algorithms, with examples of methods used in generating permutations and quasigroups. The overview of MAC algorithm in Section 2.3 presents the general working principles of MAC algorithm, which includes the process of generating the hash digest and verifying the digest. The properties of a secure MAC algorithm are discussed in Section 2.4. Section 2.5 presents the design components of a MAC algorithm and their method of application. These components include the initial vector (IV), message pre-processing and generating hash digest, method of generating keys and the compression function of the MAC algorithm. The different approaches of constructing MAC algorithms have been presented in Section 2.6. Similarly, the section discusses the security analysis of the different approaches. Given that MAC algorithms can also be constructed using hash function, Section 2.7 outlines some prominent hash functions that are used as compression function for MAC algorithm. Section 2.8 discusses compression functions for MAC algorithms which are based on $2n - 10 - n$ bit compression function. Lastly, the summary of the chapter is given in Section 2.9.

2.2 Preliminaries

This section describes related mathematical terms, definitions and examples which are relevant in the development of the proposed compression function and MAC.

2.2.1 Binary Operations

**Definition 2.1.** A binary operation "\(\ast\)" on a set \(G\) is a function from \(G \times G\) to \(G\). If \(a, b \in G \times G\) then it is written as \(a \ast b\) to indicate the image of the element \((a, b)\) under the function "\(\ast\)" (Clark, 2001).

A binary operation "\(\ast\)" on a set \(G\) is a rule for combining two elements of \(G\) to produce a third element of \(G\). This rule must satisfy the following conditions:

(i) Closure: For all \(a, b \in G\), this implies that the product of \(a \ast b\) is in \(G\). This can also be represented in notation form as: \(\forall a, b \in G \implies a \ast b \in G\). This implies that the set \(G\) is closed under the operation "\(\ast\)."
(ii) Substitution of elements: For all \(a, b, c, d\) in \(G\), such that \(a = c\) and \(b = d\). This implies as \(a * b = c * d\). This can also be represented in notation form as:
\[
\forall a, b, c, d \in G : a = c \text{ and } b = d \implies a * b = c * d.
\]
This shows that substitution of values is permissible.

Other symbols such as \(+, \times, \oplus, \circ, \bullet, \circ\) can be also be used as binary operation (Clark, 2001). In this thesis, the binary operation "\(*\)" will be used consistently. In the subsequent sections, some examples are given to describe methods of using binary operations in equations.

### 2.2.2 Groups

**Definition 2.2.** A group is an algebraic structure \((G, *)\) consisting of a non-empty set \(G\) and a binary operation "\(*\)" (Clark, 2001). A group has the following properties:

(i) **Closure**: For all \(a, b\) in \(G\), the product of \(a * b\) is in \(G\). This property is known as *closure*. The notation for closure is given as: \(\forall a, b \in G \implies a * b \in G\).

(ii) **Commutativity**: For all \(a, b\) in \(G\) and \(a \neq b\), such that \(a * b = b * a\), this implies that binary operation "\(*\)" is commutative. The notation for commutativity is given as: \(\forall a, b \in G \land a \neq b : a * b = b * a \implies "*" \text{ is commutative.}\) However if \(a * b \neq b * a\) and \(a \neq b\), this implies that the operation is non-commutative. This is explained in Subsection 2.2.2.1.

(iii) **Associativity**: For all \(a, b, c\) in \(G\) and \((a * b) * c = a * (b * c)\), this implies that binary operation "\(*\)" is associative. The notation for associativity is given as: \(\forall a, b, c \in G \land a \neq b \neq c : (a * b) * c = a * (b * c) \implies "*" \text{ is associative.}\) However if \((a * b) * c \neq a * (b * c)\) and \(a \neq b \neq c\), this implies that the operation is non-associative. This is explained in Subsection 2.2.2.1.

(iv) **Existence of a Neutral element**: There is a unique member \(e\) in set \(G\) and for all \(a\) in \(G\), such that \(e * a = a\). The notation for neutral element is given as: \(\exists e \in G \land \forall a \in G : a * e = a \implies e \text{ is a neutral element.}\)

(v) **Existence of an Inverse**: For all \(a, a^{-1}\) in \(G\), such that \(a * a^{-1} = e\). This implies that \(a^{-1}\) is inverse of \(a\).
Groups can be categorized into *abelian* and *non-abelian* groups. Abelian groups, sometimes called *commutative groups* have the commutative and associative property. Abelian and non-abelian groups are described in Subsection 2.2.2.1.

### 2.2.2.1 Abelian and Non-Abelian Groups

A group \((G, \star)\) is said to be *abelian* if the group binary operation satisfies all the group properties described in Definition 2.2. The order in which elements are added or multiplied the operation in an abelian operation is not significant (Clark, 2001). Examples of abelian groups are Boolean operators such as XOR \((\oplus)\), AND \((\land)\), OR \((\lor)\), additive group of integers modulo \(n\) \(((a + b) \mod n)\) and multiplicative group of integers modulo \(n\) \(((a \times b) \mod n)\).

Non-abelian groups on the other hand do not satisfy all the group properties described in Definition 2.2, particularly commutativity and associativity. Essentially, it can be said that the significant difference between the two groups is that abelian groups are always commutative and associative, while non-abelian groups are not always commutative nor associative. Prominent example of non-abelian groups are quasigroups, which are discussed in detail in Subsection 2.2.4.

**Example 2.1.** The following example illustrates the commutative and associative property of an abelian group.

Let the set \(Z^+_p = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}\) be a finite group of positive integers under a XOR \((\oplus)\) binary operation. The XOR table is given in Table 2.1.
Table 2.1: Example of Abelian Group

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>7</td>
<td>6</td>
<td>9</td>
<td>8</td>
<td>B</td>
<td>A</td>
<td>D</td>
<td>C</td>
<td>F</td>
<td>E</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>6</td>
<td>7</td>
<td>4</td>
<td>5</td>
<td>A</td>
<td>B</td>
<td>8</td>
<td>9</td>
<td>E</td>
<td>F</td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>B</td>
<td>A</td>
<td>9</td>
<td>8</td>
<td>F</td>
<td>E</td>
<td>D</td>
<td>C</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>8</td>
<td>9</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>4</td>
<td>7</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>D</td>
<td>C</td>
<td>F</td>
<td>E</td>
<td>9</td>
<td>8</td>
<td>B</td>
<td>A</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>7</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>E</td>
<td>F</td>
<td>C</td>
<td>D</td>
<td>A</td>
<td>B</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>F</td>
<td>E</td>
<td>D</td>
<td>C</td>
<td>B</td>
<td>A</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>9</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>8</td>
<td>B</td>
<td>A</td>
<td>D</td>
<td>C</td>
<td>F</td>
<td>E</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>A</td>
<td>A</td>
<td>B</td>
<td>8</td>
<td>9</td>
<td>3</td>
<td>F</td>
<td>C</td>
<td>D</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>6</td>
<td>7</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>B</td>
<td>B</td>
<td>C</td>
<td>9</td>
<td>8</td>
<td>F</td>
<td>E</td>
<td>D</td>
<td>C</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>8</td>
<td>9</td>
<td>A</td>
<td>B</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>D</td>
<td>D</td>
<td>C</td>
<td>F</td>
<td>E</td>
<td>9</td>
<td>8</td>
<td>B</td>
<td>A</td>
<td>5</td>
<td>4</td>
<td>7</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>E</td>
<td>E</td>
<td>F</td>
<td>C</td>
<td>D</td>
<td>A</td>
<td>B</td>
<td>8</td>
<td>9</td>
<td>6</td>
<td>7</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>E</td>
<td>D</td>
<td>C</td>
<td>B</td>
<td>A</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Consider the following two operations, where the elements are added using the XOR table in Table 2.1.

\[(1 \oplus 2) \oplus 3 = 3 \oplus 3 = 0\]

\[1 \oplus (2 \oplus 3) = 1 \oplus 1 = 0\]

It can be observed that the result of the two operations are equal irrespective of the order in which the elements are XORed.

Example 2.2. This example illustrates the non-commutative and non-associative property of a non-abelian group. A random quasigroup of order 15 (Table 2.2) is used in this example.
Table 2.2: Example of Non-abelian Group

<table>
<thead>
<tr>
<th>*</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5</td>
<td>B</td>
<td>3</td>
<td>E</td>
<td>A</td>
<td>1</td>
<td>0</td>
<td>C</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>9</td>
<td>F</td>
<td>D</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
<td>1</td>
<td>0</td>
<td>C</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>9</td>
<td>F</td>
<td>D</td>
<td>7</td>
<td>8</td>
<td>5</td>
<td>B</td>
<td>3</td>
<td>E</td>
</tr>
<tr>
<td>2</td>
<td>F</td>
<td>D</td>
<td>7</td>
<td>8</td>
<td>5</td>
<td>B</td>
<td>3</td>
<td>E</td>
<td>A</td>
<td>1</td>
<td>0</td>
<td>C</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>8</td>
<td>5</td>
<td>B</td>
<td>3</td>
<td>E</td>
<td>A</td>
<td>1</td>
<td>0</td>
<td>C</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>9</td>
<td>F</td>
<td>D</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>9</td>
<td>F</td>
<td>D</td>
<td>7</td>
<td>8</td>
<td>5</td>
<td>B</td>
<td>3</td>
<td>E</td>
<td>A</td>
<td>1</td>
<td>0</td>
<td>C</td>
</tr>
<tr>
<td>5</td>
<td>B</td>
<td>3</td>
<td>E</td>
<td>A</td>
<td>1</td>
<td>0</td>
<td>C</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>9</td>
<td>F</td>
<td>D</td>
<td>7</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>C</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>9</td>
<td>F</td>
<td>D</td>
<td>7</td>
<td>8</td>
<td>5</td>
<td>B</td>
<td>3</td>
<td>E</td>
<td>A</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>C</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>9</td>
<td>F</td>
<td>D</td>
<td>7</td>
<td>8</td>
<td>5</td>
<td>B</td>
<td>3</td>
<td>E</td>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>9</td>
<td>F</td>
<td>D</td>
<td>7</td>
<td>8</td>
<td>5</td>
<td>B</td>
<td>3</td>
<td>E</td>
<td>A</td>
<td>1</td>
<td>0</td>
<td>C</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>0</td>
<td>C</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>9</td>
<td>F</td>
<td>D</td>
<td>7</td>
<td>8</td>
<td>5</td>
<td>B</td>
<td>3</td>
<td>E</td>
<td>A</td>
</tr>
<tr>
<td>A</td>
<td>3</td>
<td>E</td>
<td>A</td>
<td>1</td>
<td>0</td>
<td>C</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>9</td>
<td>F</td>
<td>D</td>
<td>7</td>
<td>8</td>
<td>5</td>
<td>B</td>
</tr>
<tr>
<td>B</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>9</td>
<td>F</td>
<td>D</td>
<td>7</td>
<td>8</td>
<td>5</td>
<td>B</td>
<td>3</td>
<td>E</td>
<td>A</td>
<td>1</td>
<td>0</td>
<td>C</td>
</tr>
<tr>
<td>C</td>
<td>5</td>
<td>B</td>
<td>3</td>
<td>E</td>
<td>A</td>
<td>1</td>
<td>0</td>
<td>C</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>9</td>
<td>F</td>
<td>D</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>D</td>
<td>6</td>
<td>2</td>
<td>9</td>
<td>F</td>
<td>D</td>
<td>7</td>
<td>8</td>
<td>5</td>
<td>B</td>
<td>3</td>
<td>E</td>
<td>A</td>
<td>1</td>
<td>0</td>
<td>C</td>
<td>4</td>
</tr>
<tr>
<td>E</td>
<td>D</td>
<td>7</td>
<td>8</td>
<td>5</td>
<td>B</td>
<td>3</td>
<td>E</td>
<td>A</td>
<td>1</td>
<td>0</td>
<td>C</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>9</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>E</td>
<td>A</td>
<td>1</td>
<td>0</td>
<td>C</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>9</td>
<td>F</td>
<td>D</td>
<td>7</td>
<td>8</td>
<td>5</td>
<td>B</td>
<td>3</td>
</tr>
</tbody>
</table>

Consider the following two operations, where the elements are added using the quasigroup table in Table 2.2.

\[(1 \ast 2) \ast 3 = 0 \ast 3 = E\]

\[1 \ast (2 \ast 3) = 1 \ast 8 = F\]

From the above example, it can be observed that the results of the two operations are not equal. This is due to the non-commutative and non-associative property of quasigroups. Damm (2003) calculated the number of non-commutative quasigroups that can be generated. The result of Damm’s finding is given in Table 2.3. The result also shows that quasigroups can always be replaced when needed.
Table 2.3: Number of Non-commutative Quasigroups (Damm, 2003)

<table>
<thead>
<tr>
<th>Order</th>
<th>Number of Quasigroups</th>
<th>Totally Non-commutative</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>24</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>1,344</td>
<td>18</td>
</tr>
<tr>
<td>6</td>
<td>1,128,960</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>12,198,297,600</td>
<td>4,800</td>
</tr>
<tr>
<td>8</td>
<td>2,697,818,265,354,240</td>
<td>116,640</td>
</tr>
<tr>
<td>9</td>
<td>15,224,734,061,278,915,461,120</td>
<td>222,634,440</td>
</tr>
<tr>
<td>10</td>
<td>2,750,892,211,809,148,994,633,229,926,400</td>
<td>&gt;40,000,000</td>
</tr>
</tbody>
</table>

This research will focus on non-abelian groups particularly quasigroup, as quasigroups are more preferable over abelian structures in the development of cryptographic algorithms and error detection systems. Formal definition and methods of generating quasigroups have been outlined in Subsection 2.2.4.

2.2.3 Permutations

A permutation of \([n]\) is a one-to-one and onto function from \([n]\) to \([n]\), and \(S_n\) is the set of all permutations of \([n]\) (Clark, 2001). This subsection will describe some properties, definition and method of generating permutations for cryptographic applications.

Definition 2.3. (Ecker & Poch, 1986) A Permutation \(\pi\) is a bijection of a finite group \((Q, +)\) onto itself \(\delta : Q \rightarrow Q\). The permutation a group \((Q, +)\) is said to be non-commutative iff for all \(x, y \in Q\) such that:

\[ x + \pi(y) \neq \pi(x) + y \implies x \neq \pi(x) \quad (2.1) \]

Proposition 2.1. Permutation \(\pi\) is a non-commutative permutation on a finite group \(Q\) and satisfy for all \(u, v \in Q\) such that:

\[ \pi(u) + v = \pi(v) + u \implies u = v \quad (2.2) \]

Proof: The proposition is converse of definition. Thus, they are equivalent.
2.2.3.1 Generating Permutations

The following subsection describes the basic methodology for generating permutations. Other methods can be found in (Ecker & Poch, 1986).

Example 2.3. Given a set \( x = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} \) we choose a weight \( w_i \) such that \( \{0 \times w_i \times 1, w_i \times 2, w_i \times 3, w_i \times 4, w_i \times 5, w_i \times 6, w_i \times 7, w_i \times 8, w_i \times 9\} \mod 10 \) gives a permutation \( w_i \) of the decimal digits. It is required that \( w_i \) is relatively prime to 10.

\[ \text{GCD} (w_i, 10) = 1 \] (2.3)

By choosing \( w_i = 3 \), the following permutation is generated:

\[ \pi(x) = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 0 & 3 & 6 & 9 & 2 & 5 & 8 & 1 & 4 & 7 \end{pmatrix} \]

Some properties of permutations will be highlighted in the next subsection.

2.2.3.2 Properties of Permutations

Four important properties can be observed from the permutation generated using Equation 2.3. These properties can be applied to all sets of permutations.

(i) The first property is bijection, which means that the permutation has a one to one and onto mapping.

(ii) Second property is that the permutation fixes values of 0 and 5, that is \( \pi(0) = 0 \) and \( \pi(5) = 5 \). A more preferred permutation is one that is bijective with no structure and no fix element such as \( \pi(x) = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 3 & 8 & 9 & 5 & 2 & 6 & 1 & 0 & 4 & 7 \end{pmatrix} \).

(iii) The third property is the structure of permutation. The permutation begins with even numbers "0 3 6 9 2", followed by odd numbers "5 8 1 4 7". A structured permutation could pose a security threat to a cryptographic system. Furthermore, error detection systems based on structures permutations have been shown to be less efficient than unstructured permutations with no fixed points (Schulz, 2001).
(iv) Fourthly the permutation is non-commutative, that is \( \pi(1) = 3 \) but \( \pi(3) \neq 1 \). The property applies to all the elements in the permutation.

Large permutations are preferred over smaller permutations in the development of cryptographic algorithms. For example, the fixed permutation used in AES encryption algorithm is a permutation of 256 bytes (NIST, 2001). A cryptographic algorithm that uses large permutations of say order 256 has the capability to utilize approximately about \( 9 \times 10^{506} \) possible combinations of permutations, which is practically unlimited number of permutations (Scieln, 2002). Similarly, non-commutative permutations are used to increases the efficiency of the system in error detection (Schulz, 1991; Belyavskaya et al., 2003).

2.2.4 Quasigroups (Latin squares)

A quasigroup \((Q, \ast)\) is a set of elements \(Q\) together with a binary operation \(\ast\). The multiplication table for a Latin square forms a quasigroup, such that an element occurs only once in a row and column (Schulz, 1991). A quasigroup has the following properties (Ecker & Poch, 1986):

**Definition 2.4.** (Scieln, 2002). A quasigroup \((Q, \ast)\) is an algebraic structure containing a set of elements \(Q\) together with a binary operation \(\ast\). For all \(a, b\) in \(Q\), there exist unique solution \(x, y\) in \(Q\), such that:

(i) For all \(a, b\) in \(Q\), the product of \(a \ast b\) is in \(Q\). This property is known as closure.

(ii) For all \(a, b\) in \(Q\), there exist unique solution \(x, y\) in \(Q\), such that the product of \(x \ast a = b\) and also the product of \(a \ast y = b\).

(iii) For all \(a, b\) in \(Q\) and \(a \neq b\), such that \(a \ast b \neq b \ast a\), this implies that binary operation \(\ast\) is non-commutative.

(iv) For all \(a, b, c\) in \(Q\), if \((a \ast b) \ast c = a \ast (b \ast c)\), this implies that binary operation \(\ast\) is associative.

**Proposition 2.2.** \((Q, \ast)\) is a non-associative quasigroup on a finite integer set \(Z_n\) which satisfy for all \(x, y\) in \(Q\), such that;
\[(x \ast y) \ast a \neq (x \ast a) \ast y \implies a \neq y\] \hspace{1cm} (2.4)

**Proof:** Given the integers \(x, y, a\) in \(Q\) under a binary operation \(\ast\). If \((x \ast y) \ast a = (x \ast a) \ast y\) under a non-associative binary operation, this implies that \(a = y\). Thus the quasigroup \((Q, \ast)\) is non-associative.

Quasigroups (Latin squares) have many applications especially in the area of cryptography (Battey & Parakh, 2013) and coding theory (Schulz, 1991; Ecker & Poch, 1986; Belyavskaya et al., 2003). Recent work in the development of block ciphers using hybrid cubes, which are generated from Latin squares can be found in the work of Jamel et al. (2011b). Similarly, they have been in the development of block and streams ciphers (Smile, 1997; Battey & Parakh, 2013). Other applications of quasigroups are in the development of error detection systems, and they have been shown to be more efficient than modulo \(m\) check digit systems (Ecker & Poch, 1986; Schulz, 1991; Belyavskaya et al., 2003). The next subsection will discuss some methods of generating quasigroups.

### 2.2.4.1 Generating Quasigroups

There are many different methods of generating quasigroups. This research will discuss only two methods for generating quasigroups. The first method was adopted in Marnas et al. (2003) and Gligoroski (2004) and the second method was discussed in the work of Ecker & Poch (1986). Both methods are simple, fast and easy to implement and can be used to generate huge quasigroups of size 256 bytes.

**A. Quasigroups based on Permutation**

This method was adopted by Marnas et al. (2003) and Gligoroski (2004) for developing cryptographic primitive for cipher.

**Definition 2.5.** (Marnas et al., 2003; Gligoroski, 2004). Given that \(Z_p = (1, 2, \ldots, j, \ldots, n)\) is a set of positive finite integers such that the permutation
\( P = (a_{11}, a_{12}, \ldots, j, \ldots, n) \) is a permutation of \( \mathbb{Z}_p \), which defines the first row of the quasigroup (Latin square). The method of constructing the quasigroup is defined by Equation 2.5.

\[ i \ast j = i \times a_{ij} \text{mod} P \]  

(2.5)

where \( i \) is row, \( j \) is column, \( a_{ij} \) is first row in quasigroup and \( P \) is prime such that \( P = n + 1 \). A description of generating quasigroup using permutations is given in the following example.

**Example 2.4.** (Marnas et al., 2003). A randomly generated permutation \( P \) of order 6 is given as \( P(\mathbb{Z}_p) = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 3 & 5 & 2 & 1 & 6 & 4 \end{pmatrix} \), such that the first row \( a_{1j} \) is 3, 5, 2, 1, 6, 4.

**Table 2.4:** Generating Quasigroup of order 6 based on Permutations

<table>
<thead>
<tr>
<th>*</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>2 \times 3 \text{mod} 7</td>
<td>2 \times 5 \text{mod} 7</td>
<td>2 \times 2 \text{mod} 7</td>
<td>2 \times 1 \text{mod} 7</td>
<td>2 \times 6 \text{mod} 7</td>
<td>2 \times 4 \text{mod} 7</td>
</tr>
<tr>
<td>3</td>
<td>3 \times 3 \text{mod} 7</td>
<td>3 \times 5 \text{mod} 7</td>
<td>3 \times 2 \text{mod} 7</td>
<td>3 \times 1 \text{mod} 7</td>
<td>3 \times 6 \text{mod} 7</td>
<td>3 \times 4 \text{mod} 7</td>
</tr>
<tr>
<td>4</td>
<td>4 \times 3 \text{mod} 7</td>
<td>4 \times 5 \text{mod} 7</td>
<td>4 \times 2 \text{mod} 7</td>
<td>4 \times 1 \text{mod} 7</td>
<td>4 \times 6 \text{mod} 7</td>
<td>4 \times 4 \text{mod} 7</td>
</tr>
<tr>
<td>5</td>
<td>5 \times 3 \text{mod} 7</td>
<td>5 \times 5 \text{mod} 7</td>
<td>5 \times 2 \text{mod} 7</td>
<td>5 \times 1 \text{mod} 7</td>
<td>5 \times 6 \text{mod} 7</td>
<td>5 \times 4 \text{mod} 7</td>
</tr>
<tr>
<td>6</td>
<td>6 \times 3 \text{mod} 7</td>
<td>6 \times 5 \text{mod} 7</td>
<td>6 \times 2 \text{mod} 7</td>
<td>6 \times 1 \text{mod} 7</td>
<td>6 \times 6 \text{mod} 7</td>
<td>6 \times 4 \text{mod} 7</td>
</tr>
</tbody>
</table>

The quasigroup given in Table 2.4 is generated using Equation 2.5. The first row in Table 2.4 holds the permutation 3, 5, 2, 1, 6, 4. The first column of the second row is calculated as: \( 2 \times 3 \text{mod} 7 \), where 2, 3 and 7 represents 2\(^{nd}\) row 1\(^{st}\) column, permutation and order of permutation respectively. The second column of the second row is also calculated as: \( 2 \times 5 \text{mod} 7 \), where 2, 3 and 7 represents 2\(^{nd}\) row 2\(^{nd}\) column, permutation and order of permutation respectively. The quasigroup in Table 2.4 is further refined and presented in Table 2.5.

**Table 2.5:** Result from Table 2.4

<table>
<thead>
<tr>
<th>*</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>6</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>6</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>
The next step is to determine if the quasigroup in Table 2.5 meets the properties described in Definition 2.4. This is explained in the following example.

**Example 2.5.** Let \((Q, \cdot)\) be a quasigroup from Table 2.5 and also values of \(x, y, a, b, c\) in \(Q\) be 3, 4, 5, 6, 1 receptively.

(i) When elements are multiplied under a defined quasigroup, the product must be found in the quasigroup. For instance, the product 5 \(*\) 6 under the quasigroup in Table 2.5 is 6. Similarly, the product of 3 \(*\) 4 under same quasigroup is 3. This property is known as closure property as defined in Definition 2.2.

(ii) The multiplication of different pairs of elements \((a \cdot b)\) and \((x \cdot y)\) can produce the equal result. For instance, using the quasigroup in Table 2.5. the product of 1 \(*\) 5 = 6. So also the product of 6 \(*\) 4 = 6. This is another property of quasigroup.

(iii) Multiplying pair of elements in a different order produces different results. For instance, using Table 2.5 3 \(*\) 4 = 3 and 4 \(*\) 3 = 1. This property is known as non-commutativity. Non-commutative quasigroups are preferred over commutative quasigroups especially in the design of error detection systems and check digit systems (Ecker & Poch, 1986; Bakeva et al., 2011; Schulz, 1991, 2001).

**B. Quasigroups based on Linear Mapping**

This method is similar to the method of generating permutations as described in Subsection 2.2.3.1, where three positive integers \(h, k, l\) are selected, such that \(h\) and \(k\) are relatively prime to \(n\). This defines a quasigroup as given in Subsection 2.2.4 (Ecker & Poch, 1986).

**Definition 2.6.** Given that \(h, k, l\) are fixed integers where \(h\) and \(k\) are relatively prime to \(n\) defines a quasigroup on the set \((\mathbb{Z}_n) = \{0, 1, \ldots, n - 1\}\). A quasigroup is defined as:

\[
a \cdot b = ((h \cdot a) + (k \cdot b) + l) \mod n
\]  

where \(n\) is a prime number and \(h \neq k\), such that \(\text{GCD}(h, n) = \text{GCD}(k, n) = 1\).
Example 2.6. Let $h, k, l, n$ be 2, 3, 5 and 7 respectively such that the GCD $(2, 7) = \text{GCD} (3, 7) = \text{GCD} (5, 7) = 1$. A quasigroup is generated as Table 2.6.

Table 2.6: Generating Quasigroup of order 7 using Linear Mapping

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>2</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>6</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>0</td>
</tr>
</tbody>
</table>

2.2.5 Quasigroups String Transformation

The concept of string transformation is illustrated in Definition 2.7.

Definition 2.7. (Smile, 1997) Given a quasigroup $(Q, *)$ with elements $a_1, a_2, \ldots, a_n$ where $a_i$ in $Q$, $i = 1, 2, \ldots, n$. Let $l$ be a constant leader where $l$ in $Q$. A quasigroup transformation $E = E^*, l : Q^+ \rightarrow Q^+$ is given as:

\[
\begin{align*}
    b_1 &= l * a_1 \\
    b_2 &= b_1 * a_2 \\
    b_3 &= b_2 * a_3 \\
    \vdots \\
    b_n &= b_{n-1} * a_n
\end{align*}
\]

The generalization for this transformation is given in Equation 2.7.

\[
E(a_1, \ldots, a_n) = (b_1, \ldots, b_n) = \begin{cases} 
    l * a_1, \\
    b_{i-1} * b_i, & i = 2 \ldots n
\end{cases}
\] (2.7)

Without loss of generalization, let $a = a_1, a_2, \ldots, a_n$ and $b = b_1, b_2, \ldots, b_n$. The transformation $E$ is given as $E^{(n)}a = b$, where $n$ is size of $a$ and $b$.

Example 2.7. This example demonstrates the transformation mechanism of quasigroup transformation. Assuming an input $a = 12345$, and leader $l = 6$ are processed using the transformation equation given in Equation 2.7 and quasigroup
given in Table 2.7. The following is obtained:

<table>
<thead>
<tr>
<th>*</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>2</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>6</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>0</td>
</tr>
</tbody>
</table>

6 = 6*1
2 = 6*2
4 = 2*3
4 = 4*4
0 = 4*5

Without loss of generality, let the quasigroup string transformation be represented as function \( F(x) \). From the above example, it can be deduced that the function \( F(x) \) which uses a quasigroup multiplication in Table 2.7 to transform the input 12345 into the output 62440 is a quasigroup transformation function.

The transformation property of quasigroup string transformation makes it a suitable candidate for developing primitives for cryptographic algorithms (Mileva & Markovski, 2010; Smile, 1997; Dimitrova & Bakeva, 2012).

2.3 Overview of Message Authentication Code (MAC)

Confidentiality protects the privacy of an information, however it does not guarantee the integrity of the information. A practical application of a MAC is that it allows one party (the sender) to send a message to another party (the receiver) in such a way that if the message is modified during transmission, then the receiver will almost certainly detect this. The MAC is also called data-origin authentication or message authentication. Message authentication is said to protect the integrity of a message, ensuring that each message that it is received and deemed acceptable is arriving in the same condition that it was sent out, with no bits inserted, missing or modified (Bellare
REFERENCES


NIST (2012). SHA-3 Selection Announcement The National Institute of Standards and Technology (NIST) is pleased to announce the selection of K, p. 3.


