SOME HASH FUNCTIONS USING CAYLEY GRAPHS

THESIS

SUBMITTED TO

THE UNIVERSITY OF CALICUT

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY IN MATHEMATICS

IN THE FACULTY OF SCIENCE

BY

V.VIBITHA KOCHAMANI

UNDER THE GUIDANCE OF

DR.LILLY.P.L

CENTRE FOR RESEARCH IN MATHEMATICAL SCIENCES

DEPARTMENT OF MATHEMATICS

ST. JOSEPH'S COLLEGE (AUTONOMOUS)

IRINJALAKUDA 680121

THRISSUR DISTRICT

KERALA STATE

INDIA

OCTOBER 2019

CERTIFICATE

Dr.Lilly P.L

Associate Professor

Department of Mathematics

St.Joseph's College (Autonomous)

Irinjalakuda-680121

This is to certify that the thesis entitled ***Some Hash Functions Using Cayley Graphs***, submitted by full-time research scholar Ms.V.Vibitha Kochamani, Department of Mathematics,St.Joseph's College (Autonomous), Irinjalakuda to the University of Calicut, in partial fulfillment of requirement for the degree of Doctor of Philosophy in Mathematics, is a bonafide record of research work undertaken by her in the Centre for Research in Mathematical Science, St.Joseph's College (Autonomous), Irinjalakuda, under my supervision during the period 2016-2019 and that no part thereof has been presented before for any other degree.

Irinjalakuda

October 2019

Dr.Lilly P.L

Research Supervisor

DECLARATION

I hereby declare that this thesis entitled ***Some Hash Functions Using Cayley Graphs***, is the record of bonafide research I carried out in the Centre for Research in Mathematical Sciences, St.Joseph's College (Autonomous), Irinjalakuda, under the supervision of Dr.Lilly P.L, Associate Professor, Department of Mathematics, St.Joseph's College (Autonomous), Irinjalakuda.

I further declare that this thesis, or any part thereof, has not previously formed the basis for the award of any other degree, diploma, associateship, fellowship or any other similar title of recognition.

V.Vibitha Kochamani
Full time Research Scholar
Department of Mathematics
St.Joseph's College (Autonomous)
Irinjalakuda
October 2019

ACKNOWLEDGEMENT

First and foremost, I would like to thank God Almighty for giving me the strength,knowledge,ability and opportunity to undertake this research study and to persevere and to complete it satisfactorily. Without his blessings, this achievement would not have been possible.

In my journey towards this study, I have found a teacher, an inspiration, a role model and a pillar of support in my guide Dr.Lilly P.L,Associate Professor, Department of Mathematics, St.Joseph's College (Autonomous), Irinjalakuda, for her hearfelt support, invaluable guidance, inspiration and suggestions at all times in my quest for knowledge.She has given me all the freedom to pursue my research, while silently and discreet, ensuring that I stay on course and do not deviate from the core of my research.Without her able guidance, this thesis would not have been possible and I shall eternally be grateful to her for her assistance.

I take pride in acknowledging the insightful guidance of Dr.Joju K.T, Associate Professor,Department of Mathematics,Prajyothi Nikethan College,Pudukad for sparing his valuable time whenever I approached him and showing me the way ahead.

I wish to express my sincere thanks to Dr.Tony Thomas,Associate Professor,Department of Mathematics,Indian Institute of Technology and Management, Kerala, for fruitful discussions that I had with him.

My special thanks to Linto V.J, Software Engineer, for clearing all my doubts regarding the software like Python Programming.

I wish to place on record my sincere thanks to Dr.Sr.Christy former principal, St.Joseph's College (Autonomous) and Dr.Sr.Isabel, Principal, St.Joseph's College (Autonomous) for various help and support I received.

The Department of Mathematics, St.Joseph's College (Autonomous), provided an excellent atmosphere for my research. I am thankful to Dr.Mangalambal N.R, former head of the Department of Mathematics and Sherin Jose.T, Assistant Professor,head of the Department of Mathematics for their encouraging words and support. Each and every member of the Department of Mathematics, St.Joseph's College (Autonomous),who have been so helpful and cooperative in giving their support at all times to help me to achieve my goal.

I am also thankful to Administrative and Library staff for the support they have extended.

I have great pleasure to acknowledge my gratitude to my former colleague as well as my respectable friend, Dr.Sabna K.S, Assitant Professor,Department of Mathematics,K.K.T.M Government College,Pullut, Kodungallur and all my dear friends for their support,encouragement and motivation, which was of great help in achieving my goal.

My acknowledgement would be incomplete without thanking the biggest source of my strength,my family.I would dedicated this work to Yateendradas.N, my husband and Azad Maliakkal Y.N, my son for standing up with my craziness and always being so encouraging, supportive and made a tremendous contribution in helping me

to reach this stage.I thank them for standing by me in all those difficult times.

I would like to convey my special thanks to R.Miruthula kumari my mother, Padmini.K my mother-in-law and M.D.Narayanan my father-in-law, for their unwavering and unselfish love and support given to me at all times.

I would like to convey my heartful gratitude and sincere appreciation to all people who have helped and inspired me during my doctoral study.

V.Vibitha Kochamani

Department of Mathematics

St.Joseph's College (Autonomous)

Irinjalakuda

October 2019

Contents

Notations Used **v**

List of Tables **vi**

Introduction **1**

1 Preliminaries **8**

1.1 Hash Functions 8

1.2 Cryptographic Hash functions 9

1.3 Mathematical Hash Functions 10

1.4 Cayley Hash Functions 11

1.5 Security Properties of Cayley Hash Functions 13

1.6 Review of Literature 14

1.7 Mathematical Backgrounds 16

2 Hashing with Discrete Heisenberg Group and its Security Aspects 22

2.1 Constructing Hash Function 22

2.2 Properties of Hash Function 24

2.3 Efficiency of the Hash function H_1 27

2.3.1 Observed values for Efficiency 28

2.4 Pseudo-Randomness 29
2.4.1 Chi-square test 29
2.5 Protecting against Different Attacks 31
2.5.1 Finding elements of small order 31
2.5.2 Symmetric Matrices 32
2.5.3 Generic Attack 32
2.5.4 Subgroup Attack 32
2.6 Features Comparison with other Hashing Algorithms 33

3 Vectorial Version of the Cayley Hash Function 35
3.1 Designing the Vectorial version of H_1 35
3.2 Execution of H_1 , H_2 36
3.3 Security Properties 37
3.4 Efficiency of the Hash function H_2 39
3.5 Pseudo-Randomness 40
3.5.1 Chi-Square Test 40
3.6 Comparison of H_1 and H_2 41
3.7 Modified Form of the Hash Function H_1 42
3.7.1 Security Properties 42
3.8 Efficiency 43

4 Neighbourhoods in $\mathbb{P}^2$ 45
4.1 Metric in $\mathbb{P}^2$ 47
4.2 Neighbourhoods in $\mathbb{P}^2$ 51

5 Free Generators Theorem in Projective General Linear Groups 78

5.1 Free Generators theorem 78

5.2 Statement of the Free Generators Theorem 82

5.3 Proof of the Free Generators Theorem 85

6 Hash function Using Free Generators Theorem 111

6.1 Construction of the Hash Function 112

6.2 Properties of the Hash Function 113

6.2.1 Small Modifications Property 113

6.3 Resistance against different attacks 115

6.4 Consequences of the determinant 117

6.4.1 Attacks through determinant 117

6.4.2 Weakness in the Distribution of the determinant 120

6.4.3 Padding 121

6.5 Condition for opting generators 121

6.6 Security on known attacks 122

6.7 Illustrations for prime p=3 125

6.7.1 Examples for p=3 125

Conclusion 132

Appendices 135

Research Papers 173

Bibliography 175

Notations Used

$\{0,1\}^*$: the set of arbitrary binary strings

$\{0,1\}^l$: the set of binary strings of length l

$x++y$: concatenation of strings x and y

$max\{x,y\}$: maximum of x and y

mod : modular reduction

$|x|$: absolute value of x

$\parallel x \parallel$: norm of x

$H : X \rightarrow Y$: the function from set X to set Y

H_i : proposed Hash functions

$ord(A)$: order of the matrix A

$det(A)$: determinant of the matrix A

F_p : the field with prime elements

F_{p^n} : the field with p^n elements

$F_p((x))$: field of formal Laurent series over F_p

$dlog$: discrete logs in F_q.

List of Tables

2.1 Average running time of H_1 29

3.1 Execution of H_1 , H_2 37

3.2 Average running time to execute H_2 39

3.3 Comparison of the Hash Functions H_1 and H_2 41

3.4 Average running time to execute the Hash function H_3 44

6.1 $G_i(\tilde{A}), G_i(\tilde{B})$ for simple choices of eigenvectors with d=0 127

6.2 Possible choices of $f, f', \tilde{f}, \tilde{f}'$ 128

Introduction

The expansion and development of human race was all started with communications and transfer of information through generations which happened both in the direct ways and the discrete ways.Nations always trusted the discrete ways to communicate,so that the crucial knowledge,plans,actions were kept secret from enemy states and protecting their integrity,confidentiality and authenticity of data or information.

Cryptography has a prolonged and interesting history.Cryptography is an ancient technique of secured communication dates back to 1500 BC [56].Definition of the cryptography is an art of writing or solving codes.This way of expressing cryptography does not catch the intrinsic nature of modern cryptography.Until 20^{th} century,Cryptography was more in the form of an art and it concentrated on the secret communication problem.

Late 20^{th} century, the perspective of cryptography was changed completely. The theory permitted the careful study of cryptography as science.The cryptographic fields concentrated more on message authentication,digital cash,digital signatures and so on problems rather than secrete communication problems.Since the modern cryptographic fields would work on worldwide webs, it would come under some external

threats.Without attempting to provide a perfect definition of modern cryptography, we would say that it is the scientific study of techniques for securing digital informations, transactions and distributed computations [48].

There are many differences between Classical(says before $1980's$) and Modern cryptography.The important one is who uses it? Nowadays the cryptography is used for all purposes of security mechanisms in day to day life.Cryptography was initially formed as an form of art with the purpose of secret communication for military and intelligence organisations.It further developed to the science which helps the ordinary people to work in the secured systems across the world.The origin of modern cryptography plays a vital role in mathematics,some of which are taken for the purposes of cryptographic applications.The basic aim of cryptography is to give the security characteristics such as confidentiality,integrity and authentication to the data or information.

In 1953,Hahns Peter Luhn (IBM),introduced the concept of hashing[37].In [52] the word "hash"means "to chop and mix"which means that cut into pieces and combine them together as a one thing.Hash functions are extremely important for many cryptographic purposes,namely a system of message authentication over an insecure channel,detection of modified message,and so on.Hash functions are in simple and efficient way to evaluate,that compress an arbitrary length input to a fixed size of output.The idea behind the hash function is to generate a single imprint of a message and the protection of the single imprint is easier than the protection of the message itself.If the two distinct messages are unlikely mapped to a same value then we can say the function as a good hash function.

For the last few years,Cryptology was focused mainly on the privacy problem. It was extensively accepted at that time that the Authentication problem was a sub-problem of privacy problem.The security of authenticity would automatically leads from the privacy problem.W.Diffe and M.Hellman in [64] concluded, *The problems of privacy and authentication are closely related and techniques for solving one can frequently be applied to the other.*

In 1976,seminal paper of W.Diffe and M.Hellman gave a clear picture about the privacy and authentication problem and they concluded with the result that the problems are distinct.About the protection of Authenticity, they state that in [64] *Not only must a meddler be prevented from injecting totally new, authentic messages into a channel,but he must be prevented from creating apparently authentic messages by combining, or merely repeating, old messages which he has copied in the past.A cryptographic system intended to guarantee privacy will not, in general,prevent this latter form of mischief.*For the last fifteen years, the significance of research area in cryptographic field is strongly related to the topics of development in both theoretical and practical cryptographic systems which give assurance to the authenticity.

The concealment of information or protection of privacy is as old as writing itself [64].Ciphers,Codes and stenography are among a few methods that are used to conceal the information.Communication evolved from letters by post to couriers and to the emails through computers.Since new communication methods and technology are all interconnected with worldwide networks and its servers risk associated with them are increased.Cryptology was the only solution that was able to make the leap from the closed world of generals and diplomats to worldwide commercial applications

[64].

Apart from the privacy problem, Authentication of a message is highly relevant.Authentication is that modification of both content and the originator of the information has not changed.An attacker who tries to modify contents or origin of information is called an Active attacker [64].

Nowadays,Cryptographic hash functions are the basic structure for security schemes and procedures in most of the computer networks.This function is used to guarantee authentication,confidentiality and mainly the integrity of data.The cryptographic hash functions are the strongest tool in the design techniques to protect authenticity of information and threat of reputation. Most of the hash functions in use today have constructions that apply some sort of iterative design involving a compression function and a transformation as the Merkle-Damgard [54].But some hash functions are endangered in modern days which give suggestions to investigate more new cryptographically secured designs.One of them is the Provably Secure Hash Function, is the function which depends on the difficulty of breaking a known "hard"problems associated with security.

Provably secured hash functions have many illustrations, one of them is the Cayley Hash Functions which is related to the Cayley graph of some groups.The Cayley graphs of the underlying groups of the past proposals are Expander graphs, thus presenting interesting properties such that of the rapid mixing of Markov chains [17].Cayley Hash functions are created and their security properties are alleged to the hardness of the mathematical problem which is related to the Cayley graphs of some groups.

Gilles Zemor introduced the idea of building Cryptographic hash functions from Cayley hash functions of large girth [35, 34].In Zemor's construction, the security properties of the hash functions are easily stated whose mathematical problems which were believed to be hard. Zemor's cryptographic hash functions had the worth and notable property that any Local modifications of a message text would definitely change its hashed value.In 1991, Zemor introduced a family of hash functions whose values correspond to matrix products in groups of the form $SL_2(F_p)$ for a prime p [35, 12].

In 1994,this occurrence of the Cayley hash functions was broken by Zemor.To enhance the security properties of the hash function they introduced a new idea which change the prime field to the field with 2^n elements.In other words, the new idea of Tillich and Zemor hash function with the group $SL_2(F_{2^n})$ where F_{2^n} is a field with 2^n elements [46, 45],which was well-accepted for its security for so many reasons.The Tillich-Zemor cryptographic hash functions, was comparable with the existing cryptographic standards for its efficiency.The properties of the computational speed and security depends on some mathematical problems of the hash functions,both of them are not satisfied at a time.But in Tillich-Zemor hash functions have both these properties.And also it preserves the Local modifications property of the text message.

In [43], Quisquator and Joye showed the Zemor-Tillich hash function would be ideal for authenticating video sequences.There are some other schemes for Cayley hash functions were encouraged based on the Ramanujan graphs constructed by Pizer and Lubotzky-Phillips-Sarnak(LPS)[17] and the Ramanujan graphs constructed by Morgenstern [60].

The study in this thesis is started with an investigation in the Cryptographic hash functions H_1 with two generators of the discrete Heisenberg group.To overcome the flaw we introduced the Vectorial version of H_1.Further, the study leads to derive the Free Generators theorem in Projective General Linear group and its applications in hashing.

The thesis content is subdivided in this following way.

Chapter 1 deals with some preliminary definitions and results which are required in further chapters to follow the study in this thesis.

Chapter 2 contains the following study: A new construction of the Cryptographic hash functions with the discrete Heisenberg group that are in related with directed Cayley graph and the Local Modifications property for the messages are retained by proving that the girth of the Cayley graph is large.Further,the running time to execute the Cryptographic hash functions using a discrete Heisenberg group for the pair of generators over a prime p are computed and the output distribution of the cryptographic hash functions are uniform which is tested by using Pesudo-randomness characteristics to reduce the chance of occurring collisions.Also,Verified the security properties of the hash function using different attacks.

Chapter 3 gives the Vectorial form of the hash function with the discrete Heisenberg group which gives the hashed value as in the vector form.Also examine the Chi-Square Goodness of fit test,that the output distribution of the hashed values in the defined hash function are uniform.Construct a new hash function which is of different concept to form a Factorisation Problem harder.

In chapter 4, for the construction of the Free generators theorem, the requirement of the metric and neighbourhoods in $\mathbb{P}^2$ are defined and their related properties are proved.

Chapter 5 deals the following study: the construction of the Free Generators theorem in projective general linear group.In which a general method to establish free generators of free subgroups of projective general linear group, $PGL_3(F_p((x)))$ and general linear group $GL_3(F_p((x)))$ where $F_p((x))$ is the field of formal Laurent series over the prime field.

In Chapter 6 the following ideas are, defining a new construction of the Cayley Hash functions using the Free generators theorem.For a polynomial generators $\tilde{A}$ and $\tilde{B}$ of a free subgroup in $GL_3(F_p((x)))$,when we project their images of $\tilde{A}$ and $\tilde{B}$ into F_q under the quotient by $< r_n(x) >$ and clearly says that for which choices of these hash functions are the best.

The conclusion of the thesis gives insights towards the scope of study in the future.

Chapter 1

Preliminaries

This chapter deals with some preliminaries of an abstract ideas on Cryptographic Hash functions,Mathematical Hash Functions, Cayley Hash functions, Review of Literature and the Mathematical backgrounds required for the next chapters.

1.1. Hash Functions

In [38] given a set, often referred to in this setting as an **alphabet**,we define a concatenation of elements in that set as a **word** or a **string**.The number of elements concatenated is referred to as the length of the word.A string with $\{0, 1\}$ is commonly referred to as a **binary string** or a **bit string** [38].
Now we have the formal definition of the Hash function.

Definition 1.1.1. [10] A hash function $H : D \longmapsto R$ where the domain $D = \{0, 1\}^*$, and the range $R = \{0, 1\}^n$,for some $n \geqslant 1$.

Hash functions along with some additional requirements then the function becomes Cryptographic Hash functions which is explained in the next section.

1.2. Cryptographic Hash functions

The informal definition of the One-Way Hash function was evidently said by M.Rabin in [65]

Definition 1.2.1. A **One-Way Hash function** is a function H satisfying the following conditions:

i. The description of H must be publicly known and should not require any secret information for its operation.

ii. The argument X can be of arbitrary length and the result $H(X)$ has a fixed length of n bits (with $n \geqslant 64$).

iii. Given H and X, the computation of $H(X)$ must be easy.

iv. The hash function must be One-Way in the sense that given a Y in the image of H, it is hard to find a message X such that $H(X) = Y$ and given X and $H(X)$ it is hard to find a message $X^{'} \neq X$ such that $H(X) = H(X^{'})$.

The informal definition is given by M.Rabin in [65] for the Collision-Resistant Hash Function.

Definition 1.2.2. A **Collision-Resistant Hash function** is a function H satisfying the following conditions:

i.The description of H must be publicly known and should not require any secret information for its operation.

ii. The argument X can be of arbitrary length and the result $H(X)$ has a fixed length of n bits (with $n \geqslant 128$).

iii. Given H and X, the computation of $H(X)$ must be easy.

iv. The hash function must be One-Way in the sense that given a Y in the image of H, it is hard to find a message X such that $H(X) = Y$ and given X and $H(X)$ it is

hard to find a message $X^{'} \neq X$ such that $H(X) = H(X^{'})$.

v. The hash function must be collision resistant: this means that it is hard to find two distinct messages that hash to the same result.

Now we defined in the formal way.

Definition 1.2.3. [28] A **Hash Family** consists of three elements (X, Y, H) where the following conditions are satisfied

(i) X is a set of possible messages.

(ii) Y is a finite set of possible message digests or authentication tags.

(iii) there is a hash function H such that $H : X \longrightarrow Y$.

Definition 1.2.4. [28] A hash function $H : X \longrightarrow Y$ and an element $y \in Y$, then find $x \in X$ such that $H(x) = y$. A hash function for which pre-image cannot be efficiently solved is often said to be one-way or pre-image resistant.

Definition 1.2.5. [28] A hash function $H : X \longrightarrow Y$ and an element $x \in X$, then find $x^{'} \in X$ such that $x^{'} \neq x$ and $H(x^{'}) = H(x)$. A hash function for which second pre-image cannot be efficiently solved is often said to be second pre-image resistant.

Definition 1.2.6. [28] A hash function $H : X \longrightarrow Y$, then find $x, x^{'} \in X$ such that $x \neq x^{'}$ and $H(x^{'}) = H(x)$.

A hash function for which collision cannot be efficiently solved is often said to be Collision Resistant.

1.3. Mathematical Hash Functions

[12] Nowadays most of the hash functions are constructed using iterative designs.These traditionally constructed hash functions are based on performing several rounds of

complex bit operations in sequence, with the hope that it is difficult to reverse.The two most commonly used mathematical hash functions are SHA-1 and MD5 and serious attacks were developed against them.Research community took lots of efforts to replace such type of hash functions [48].

Some options are proposed to create hash functions using algebraic structures.One example of a mathematical hash function is the *modular arithmetic secure hash algorithm* (MASH-1), which follows the Merkle-Damgard transform, but it uses a compression function based on modular arithmetic.MASH-1 involves the use of an RSA-like modulus N that should be difficult to factor.Its security is based partially on the difficulty of extracting modular roots.

Other example of the mathematical hash function is the *Provably Secure Hash Function.*The security of this hash function is based on some hard mathematical problems and finding collisions of the hash function is as hard as breaking the underlying problem.Their security is more than just relying on complex mixing of bits as in the classical approach.

1.4. Cayley Hash Functions

In [60] describes Cayley hashes as an *efficient and provably secure hash functions constructed from the Cayley graphs of (projective) linear groups.*

Definition 1.4.1. [12] A graph is an ordered pair $G = (V, E)$ composed of a vertex set V together with an edge multi-set E.The vertex set V can be any set and the edge multi-set E is a multi-set whose elements are of the form $\{v, w\}$ or $\{v\}$ where v and w are distinct vertices.An edge of the form $\{v\}$ is called loop.

Definition 1.4.2. [12] The diameter of G is defined by $diam(G) = max_{v,w \in V} dist(v, w)$,

that is the maximal length of the shortest path between any two vertices v and w. If there is no path connecting two vertices then conventionally the distance is define infinite.

Definition 1.4.3. [38] The directed girth of a graph G is the largest integer τ such that for any distinct vertices v, w there exist at most one path from v to w of length less than τ.

We note that the directed girth of a graph is different than the girth of a graph, which is defined as the length of the shortest cycle.

Definition 1.4.4. [12] Let G be a multiplicative group and $S = \{s_1, s_2, ..., s_k\}$ be a subset of G.

A Cayley graph $C_{G,S} = (V, E)$ is a k-regular graph constructed from the group G with respect to $S \subset G$ as follows: For each element $g \in G$, V contains a vertex v_g associated to g. E contains the directed edge (v_{g_1}, v_{g_2}) if and only if there is $s_i \in S$ for some i such that $g_2 = g_1 s_i$. The elements of S are called the graph generators.

If S is stable under inversion ($S = S^{-1}$), then the graph $C_{G,S}$ is undirected.

$C_{G,S}$ is a connected graph if and only if S generates the whole group G.

Definition 1.4.5. [12] Let G be a finite (semi) group with a set of generators $\mathcal{S}$ that has the same size as the text alphabet $\mathcal{A}$.Choose a function $\pi : \mathcal{A} \longmapsto \mathcal{S}$ such that defines an one to one correspondence between $\mathcal{A}$ and $\mathcal{S}$.

Algorithm:[12]

The hash value of the text $x_1x_2.....x_k$ is the (semi) group element $\pi(x_1)\pi(x_2).....\pi(x_k)$. One of the advantages of this design is that the computation of the hash value can be easily parallelized due to the associativity property $\pi(xy) = \pi(x)\pi(y)$ for any x and y in the (semi) group and if the graph has large girth (length of the smallest cycle in

the graph), the hash function is protected against small modifications of the message input.

1.5. Security Properties of Cayley Hash Functions

Cayley Hash functions are strongly connected to the hardness of the Mathematical problems.

Definition 1.5.1. [12] Let G be a group and let $S = \{s_1, ..., s_k\} \subset G$ be a generating set of G.Let L be of poly logarithmic (small) in the size of G.

(i) **Balance Problem:** Find an efficient algorithm that returns two words $m_1....m_l$ and $m_1^{'}.....m_l^{'}$, with $l, l^{'} < L$, $m_i, m_i^{'} \in \{1, ..., k\}$ that yield equal products in G, that is, $\Pi_{i=1}^{l} s_{m_i} = \Pi_{i=1}^{l^{'}} s_{m_i^{'}}$.

(ii)**Representation Problem:** Find an efficient algorithm that returns a word $m_1....m_l$ with $l < L$, $m_i \in \{1, ..., k\}$ such that $\Pi_{i=1}^{l} s_{m_i} = 1$.

(iii) **Factorization Problem:** Find an efficient algorithm that given any element $g \in G$ returns a words $m_1....m_l$ with $l < L$, $m_i \in \{1, ..., k\}$ such that $\Pi_{i=1}^{l} s_{m_i} = g$.

The Cayley Hash function is collision resistant if and only if the balance problem is hard in the defined group.The associated Cayley graph is second pre-image resistant if the representation problem is hard and it is pre-image resistant if and only if the corresponding factorization problem is hard in the defined group [62, 22].

1.6. Review of Literature

In the period of time 1990s, Zemor [35] introduced the idea of constructing hash functions from cayley graphs.Zemor's technique came from a desire to satisfy the following small modifications property that introduced in [36].

Proposition 1.6.1. *[36] [Small Modifications Property] There exists a* $d \in \mathbb{N}_0$ *such that if* $m^{'}$ *is any modification of* m *affecting fewer than* d *consecutive bits then* $h(m) \neq h(m^{'})$.

[38]The relation between the small modifications property and hash functions from Cayley graphs is now apparent.Namely, let G be a group with generating set $S = \{A, B\}$ and H be the associated hash function.If the directed girth of $C(G, S)$ is δ, then two messages of length less than δ cannot form a collision in H.From this motivation, Zemor present the idea of using hash functions over the group $SL_2(F_p)$ with two generators $A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$ and $B = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$ for a large prime p which was broken by many attacks by their weakness in the factorization.[12] In 1994, Tillich and Zemor's [46] paper proposed a family of hash functions that uses the group of SL_2 over a finite field of 2^n elements as platform for their design.

[12] Let n be a positive integer and let $p_n(x)$ be an irreducible polynomial of degree n over F_2.Let A_0 and A_1 be defined as follows: $A_0 = \begin{pmatrix} x & 1 \\ 1 & 0 \end{pmatrix}$ and $A_1 = \begin{pmatrix} x & x+1 \\ 1 & 1 \end{pmatrix}$.

Both A_0 and A_1 have determinant 1 over F_2.These matrices are the generators of

the Tillich-Zemor hash function.

Let $m = m_1 m_2 .....m_k \in \{0,1\}^*$ be a binary string representation of a message and $K = F_2[x]/ < p_n(x) > \cong F_{2^n}$.

The construction of the Tillich-Zemor hash functions also preserves the small modifications property using degree argument in ([46], lemma 3.5).

Lemma 1.6.2. *Suppose that* $m, m^{'}$ *are bit strings in* $\{0,1\}^*$ *such that* $H(m) = H(m^{'})$. *Then atleast one of* m,$m^{'}$ *must have length greater than* n.

[22] Finding Collision in the Tillich-Zemor hash function is equivalent to the problem of finding two products of reasonable length in {A,B} that are equal in G, which in mathematics sometimes referred to as the "Balance problem".Similarly we say that the Pre-image and Second Pre-image resistance are respectively reduced to the "Representation Problem"and "Factorization Problem".These problems are known to be very hard mathematical problems for many classes of groups [46].

Originally, some modifications of the Tillich-Zemor hash function were suggested [63, 20, 23] to improve its feasibility.

Later, the construction of the Tillich-Zemor hash function and Zemor's original idea, Cayley hashes from expander graphs have also been of great interest.The family of Ramanujan graphs over $PSL_2(F_p)$ introduced in [17]. In [60] suggested another construction based on expander graphs, called the Morgenstern hash function which took values in $PSL_2(F_{2^n})$, but was also constructible over $PSL_2(F_q)$.

In [44] Tillich and Zemor presented an attack based on the LPS hash function which was later extended to a pre-image finding algorithm by Petit in [61] and also alter the attack to the Morgenstern hash function [61]. Most recently, in [40, 41] suggests using Chui's cubic of Ramanujan graphs over $PSL_2(F_p)$, but immediately finds a corresponding attack.

Note that these expander graph constructions were different from the Tillich-Zemor's construction as they used undirected Cayley graphs and were further based on quatern ion-like algebras.Also the another construction using Pizer graphs over elliptic curves was given in [44].

The above all discussions exhibits the interest and scope of such constructions in the following chapters.

1.7. Mathematical Backgrounds

Definition 1.7.1. [13] The **Heisenberg group** is the group of all 3×3 upper triangular matrices of the form $\begin{pmatrix} 1 & a & c \\ 0 & 1 & b \\ 0 & 0 & 1 \end{pmatrix}$ under the operation of matrix multiplication.Elements a,b and c can be taken from any commutative ring with identity, often taken to be the ring of real numbers (resulting in the "Continuous Heisenberg group") or the ring of integers (resulting in the "Discrete Heisenberg group").

It is denoted by Heis group. From this definition, it is easily seen that the discrete Heisenberg group is generated by A= $\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ and B= $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$

Definition 1.7.2. [38] Let R be a commutative ring.We define the **general linear group** of degree k over R as $GL_k(R) = \{g \in M_{k\times k} : det(g) \in R^*\}$ is the group of invertible $k \times k$ matrices over R.

Now, we defined the Projective General linear group over R

Definition 1.7.3. [38] Let R be a commutative ring.We define the **projective general linear group** of degree k over R is $PGL_k(R) = GL_k(R)/Z$, where Z is the centre of $GL_k(R)$ and consists of all scalar matrices. Elements of $PGL_k(R)$ are thus cosets gZ with $g \in GL_k(R)$.

Definition 1.7.4. [38] Let p be a prime and F_p be the field with p-elements and $F_p((x))$ be the field of **formal Laurent series** over F_p.The elements of $F_p((x))$ are series of the form $f(x) = \Sigma_{k=m}^{\infty} f_k x^k$, for $f_i \in F_p$ and $m \in \mathbb{Z}$.
Because we want to see the elements of $F_p((x))$ as elements of an abstract field,not as functions and we will write f^{-1} to mean the multiplicative inverse of $f \in F_p((x))$.

Definition 1.7.5. [38] Let $M \in M_{3\times 3}(R)$. We then define $deg(M)$ to be the maximum degree of the entries of M.

Definition 1.7.6. [26, 70] **Valuation** $v(f)$ of an element $f \in F_p((x))$ as in the standard way, namely, $v(f) = \begin{cases} min\{k : g_k \neq 0\} & \text{if } g \neq 0 \\ \infty & \text{if } g = 0 \end{cases}$

Definition 1.7.7. [26, 70] The **absolute value** of $f \in F_p((x))$ as $|f| = p^{-v(f)}$

Example 1.7.8. *If* $f = x^3 + x^6$ *then* $|f| = p^{-3}$.
If $g = x^{-4} + x^{-1}$ *then* $|f| = p^4$.

Remark. [26, 70]

- $|.|$ is multiplicative, that is, $|fg| = |f||g|$ for any $f, g \in F_p((x))$
- $|.|$ is non-Archimedean, it means that it satisfies the Ultra-metric or Non-Archimedean Triangle inequality,
  $|f + g| \leqslant max\{|f|, |g|\}$ for all $f, g \in F_p((x))$

- $|.|$ is non-Archimedean, for $x, y \in F_p((x))$ then $|x| \neq |y| \Longrightarrow |x+y| = max\{|x|, |y|\}$.

Definition 1.7.9. [26, 70] The ring of integers of $F_p((x))$ with respect to the valuation and Absolute values is denoted and defined by

$\mathbb{O} = F_p[[x]] = \{f \in F_p((x)) : v(f) \geqslant 0\} = \{f \in F_p((x)) : |f| \leqslant 1\}$

Its multiplicative group of $\mathbb{O}$ is,

$\mathbb{O}^* = \{f \in F_p((x)) : v(f) = 0\} = \{f \in F_p((x)) : |f| = 1\}$.

Definition 1.7.10. [42] The **two-dimensional projective space** is defined over $F_p((x))$ as

$\mathbb{P}^2 = \mathbb{P}^2(F_p((x))) = V - \{(0,0,0)\}/\sim$, where $\sim$ is the equivalence relation as $(u_1, u_2, u_3) \sim (v_1, v_2, v_3)$, if there exist $k \in F_p((x))^* \ni (u_1, u_2, u_3) = (kv_1, kv_2, kv_3)$.

If for $(u_1, u_2, u_3) \neq (0,0,0)$, then define $[u] = [u_1 : u_2 : u_3] \in \mathbb{P}^2$ to be the equivalence class $\{k(u_1, u_2, u_3) \in V : k \in F_p((x))^*\}$.

Remark. (i) $[e_1] = [1:0:0]$, $[e_2] = [0:1:0]$,and $[e_3] = [0:0:1]$

(ii) $[f:g:h] = [1 : gf^{-1} : hf^{-1}]$ for $f \neq 0$

(iii) $[f:g:h] = [fg^{-1} : 1 : hg^{-1}]$ for $g \neq 0$

(iv) $[f:g:h] = [fh^{-1} : gh^{-1} : 1]$ for $h \neq 0$

Definition 1.7.11. [38] Consider the group $GL_3(F_p((x))$ and its subgroups are acting on V by matrix multiplication on the left,

that is, for $g = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \in GL_3(F_p((x)))$ and $u = (u_1, u_2, u_3) \in V$ then

$g.u = \begin{pmatrix} a_{11}u_1 + a_{12}u_2 + a_{13}u_3 \\ a_{21}u_1 + a_{22}u_2 + a_{23}u_3 \\ a_{31}u_1 + a_{32}u_2 + a_{33}u_3 \end{pmatrix}$.This gives the action of $GL_3(F_p((x)))$ and its sub

groups on $\mathbb{P}^2$ by $g.[u] = [g.u]$

Definition 1.7.12. [38] The elements of $PGL_3(F_p((x)))$ are cosets of the form gZ where Z is the centre of $GL_3(F_p((x)))$ and $g \in GL_3(F_p((x)))$.For ease of notation,

for an element $gZ \in PGL_3(F_p((x)))$ where $g = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$ denote it as

$gZ = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$. We know that Z acts trivially on $\mathbb{P}^2$.So consider $PGL_3(F_p((x)))$

and its subgroups as acting on $\mathbb{P}^2$ in the same way as $GL_3(F_p((x)))$ by left matrix multiplication.

Definition 1.7.13. [66] A **primitive root** of a field F_{p^k} is an element whose powers constitute all of $F^*_{p^k}$. That is, the roots is a generator of the cyclic group $F^*_{p^k}$

Remark. In [66] (i) An element $g \in F_{p^k}$ is a primitive root if and only if $g^{(p^k-1)/q} \neq 1$, for every prime q dividing $p^k - 1$.

(ii) If g is a primitive root modulo p^k then g^r is a primitive root if and only if $gcd(r, p^k - 1) = 1$

Definition 1.7.14. [5] For a fixed point $a \in X$. Let $S = \{f \in F(X) \ni f(a) \neq 0\}$ be the set of all polynomial functions that do not vanish at a.Then the fraction $\frac{f}{g}$ for

$f \in F(X)$ and $g \in S$ can be thought of as rational functions that are well defined at a is called the **Localization** where $F(X)$ is the ring of polynomial functions on X.

Definition 1.7.15. [38] Let $(u_1, u_2, u_3) \in V$ then the **∞-norm** is defined as
$\parallel (u_1, u_2, u_3) \parallel = max\{|u_1|, |u_2|, |u_3|\}$

Remark. [38] For any $u = (u_1, u_2, u_3) \in V$ with $u \neq (0, 0, 0)$ then we get that $|u_1|, |u_2|$ and $|u_3|$ must all be the zero or power of p which says that $\parallel u \parallel = \parallel (u_1, u_2, u_3) \parallel$ must also be a power of p.

The metric is defined in [32]

Definition 1.7.16. Let $[u], [v] \in \mathbb{P}^2$ such that $[u] = [u_1 : u_2 : u_3]$ and $[v] = [v_1 : v_2 : v_3]$ then the standard metric on $\mathbb{P}^2$ is defined as,

$$\begin{aligned} d([u], [v]) &= \frac{\parallel u \wedge v \parallel}{\parallel u \parallel \parallel v \parallel} \\ &= \frac{max_{i,j}\{|u_i v_j - u_j v_i|\}}{max\{|u_1|, |u_2|, |u_3|\} max\{|v_1|, |v_2|, |v_3|\}} \end{aligned}$$

Remark. Here,Non-Archimedean local fields of characteristic p is the field of formal Laurent series $F_p((x))$ over the finite field F_p.So we provide, $V \wedge V$ with ∞- norm , so that $\parallel u \wedge v \parallel = max_{i,j}\{|u_i v_j - u_j v_i|\}$.

Definition 1.7.17. [53] Let X be an arbitrary set.A **word** in X is a finite sequence of elements w which we write as $w = y_1 y_2 ..... y_n (y_i \in X)$.The number n is called the **length** of the word.The unit is the empty word.

Remark. [53] If $x \in X$ then symbols x and x^{-1} are called **literals** in X

Definition 1.7.18. [53] An expression of the type $w = x_{i_1}^{\epsilon_1} x_{i_2}^{\epsilon_2} ..... x_{i_n}^{\epsilon_n} (x_{i_j} \in X,$ $\epsilon_j \in \{1, -1\})$, is called the **group word** in X.

Definition 1.7.19. [53] A group word $w = y_1....y_n$, $(y_i \in X^{\pm 1})$ is **reduced** if for any $i = 1, ..., n-1$, $y_i \neq y_{i+1}^{-1}$.(i.e) w does not contain a sub word of the type yy^{-1} for a literal $y \in X^{\pm 1}$.

Definition 1.7.20. [53, 25] The **free group** over X is the set F(X) endowed with the product defined by:w*w' is the unique reduced word equivalent to the word ww'. The unit is the empty word.

Definition 1.7.21. [53] The cardinality of X is called the **rank** of the free group F(X).

Remark. [53] (i) If $X \subseteq G$ then every group word $w = x_{i_1}^{\epsilon_1} x_{i_2}^{\epsilon_2}.....x_{i_n}^{\epsilon_n}$ in X determines a unique element from G which is equal to the product $x_{i_1}^{\epsilon_1} x_{i_2}^{\epsilon_2}.....x_{i_n}^{\epsilon_n}$ of the elements $x_{i,j}^{\epsilon_j} \in G$.
(ii) The set X is called the **free basis of G** and G is called **free on X** or **freely generated by X**

Definition 1.7.22. [71] If G_1, G_2 are subgroups of G, then G is said to be **free product** of G_1 and G_2.If for every group H and homomorphisms $\theta_i : G_i \to H$, there is a unique homomorphism $\theta : G \to H$ so that $\theta|_{G_i} = \theta_i$, for $i = 1, 2$.

Chapter 2

Hashing with Discrete Heisenberg Group and its Security Aspects

Introduced a new Cryptographic Hash Function which is in correspondence with directed Cayley graph.Here we are showing why a large girth is needed to satisfy the local modifications of a text.We execute the average running time of various binary strings of different length and evaluate the other useful characteristics of a hash function which is Pseudo-Randomness of a hash function.Pseudo-Randomness of a hash function can be assigned uniformly over the range [0,p) to reduce the chance of occurring collisions.The security properties of the hash function using different attacks are verified.

2.1. Constructing Hash Function

Now, we are designing a Hash Function as follows: to an arbitrary text of $\{0,1\}^*$ associate the string of {A,B} obtained by substituting 0 for A and 1 for B,

then assign to A and B values of adequately chosen matrices of Heisenberg group over integers $\mathbb{Z}$,those could be A= $\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ and B= $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$ then evaluate the

product associated with the string of A's and B's in the group $Heis(F_p)$, where F_p is the field on p elements, p being chosen large prime number. The hashed value is the computed product.A multiplication by A or B in $Heis(F_p)$ requires essentially 4 additions, so hashing an n bit text requires 4n additions of log p bits. The hashed value related with the length of the input message would grow to an unlimited extent and that hashed values would lie in the matrix ring with unique factorization.Hence the message is retained digit by digit from right to left.Now, by applying the modular reductions in the hashed value which is the computed product.It loses some information and so, there is no longer factorization is trivial [22].

Definition 2.1.1. Let A= $\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ and B= $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$ be a pair of generators of

$Heis(F_P)$ and let $m = b_1b_2.....b_n$ be a binary string. Then

$H_1(m) = \pi(b_1)\pi(b_2).....\pi(b_n)$ where,

$$\pi(b_i) = \begin{cases} A & \text{if } b_i\text{=0,} \\ B & \text{if } b_i\text{=1} \end{cases}$$

This hash function is strongly related to the Cayley graph associated with $Heis(F_p)$ of generators A, B and denoted by $\mathcal{G}$.

2.2. Properties of Hash Function

Recall that the hash function construction presented above is directly associated with the Cayley graph $\mathcal{G}(G, S)$, where G is a group generated by the elements of the set S.

Proposition 2.2.1. *If we replace 'u' consecutive elements of the product,* $x = x_1x_2.....x_ix_{i+1}.....x_{i+u}x_{i+u+1}....x_t$ *where* $x_j \in S$,$j = 1$ *to* t *with 'v' consecutive elements of* $y_{i+1}.....y_{i+v} \in S$ *such that* $x = x_1x_2.....x_iy_{i+1}.....y_{i+v}x_{i+u+1}....x_t$ *have the same hashed value,then* $max(u, v) \geqslant g$.

In other words, if we can obtain Cayley graph with a large g, we protect against local modifications of the text.

Lemma 2.2.2. *Let* $A=\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ *and* $B=\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$. *Let* $S_1, S_2, ...S_t \in \{A, B\}$

with $M = S_1S_2.....S_t = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$ *with matrix multiplication over* $Heis(F_p)$. *Then* $a, b, c, d, e, f, g, h, i < 2^t$

Proof. We prove the lemma by using induction on t.

Suppose t=1, then $M = S_1 = A$ or B.

In both cases all the entries of the matrix A and matrix B are less than 2, which gives that the induction is true for t=1.

Assume that the lemma holds for $l < t$, then we get $a, b, c, d, e, f, g, h, i < 2^l$

Next, we need to prove the induction is true for t.

Let $S_1, S_2, ...S_t \in \{A, B\}$

Define $M = S_1 S_2 .....S_{t-1} = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$ and $a, b, c, d, e, f, g, h, i < 2^{t-1}$

If $S_t = A$, then $M.S_t = \begin{pmatrix} a & a+b & c \\ d & d+e & f \\ g & g+h & i \end{pmatrix} < 2^t$

Since, all the entries in the matrix M are less than 2^{t-1} and all entries in the matrix A are less than 2, so their product $M.S_t < 2^t$

Similarly, If $S_t = B$, then $M.S_t = \begin{pmatrix} a & b & c+b \\ d & e & f+e \\ g & h & i+h \end{pmatrix} < 2^t$

Since, all the entries in the matrix M are less than 2^{t-1} and all entries in the matrix B are less than 2, so their product $M.S_t < 2^t$

Hence in both cases, by induction hypothesis the lemma holds for t. ■

Proposition 2.2.3. *Let the bit '0' be mapped to the matrix* $A = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ *and the bit '1' be mapped to the matrix* $B = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$. *If two distinct messages 'u' and 'v' hashes to the same value, then the length of either u or v is atleast* $log_2(p)$.

Proof. Suppose the length of the messages u is 't' and length of the message v is $\leqslant t$.

If $\begin{pmatrix} 1 & a & b \\ 0 & 1 & c \\ 0 & 0 & 1 \end{pmatrix}$ is the hashed value of u then by lemma 2.2.2, all entries are less than or equal to 2^t.

Therefore, if $2^t < p$ then the hashed values of u and v cannot be equal because if it is equal over $\mathbb{Z}_p$ will also equal over $\mathbb{Z}$, which is impossible.

Hence, if the two binary strings have length less than $log_2(p)$, then their hashes cannot be equal over $\mathbb{Z}_p$ ■

Remark. (i) We can say the proposition 2.2.3 in other words, that the girth of the Cayley graph of $Heis(F_p)$ with generators A and B is atleast $log_2(p)$.

(ii) If we choose p is of order 2^{256}, then the above proposition 2.2.3 says that the hash function does not exist collisions unless the length of atleast one colliding bit strings is atleast 256 which have a close similarity to a lower bound of the girth of

the Cayley graph generated by $A = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ and $B = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$ over F_p.

2.3. Efficiency of the Hash function H_1

Let $A = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ and $B = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$ be the two generators of $Heis(F_p)$

and let $m = b_0b_1.....b_n$ be a binary string.Then $H_1(m) = \pi(b_0)\pi(b_1).....\pi(b_n)$ where, for $0 \leqslant i \leqslant n$,

$$\pi(b_i) = \begin{cases} A & \text{if } b_i\text{=0,} \\ B & \text{if } b_i\text{=1} \end{cases}$$

This hash function is strongly related to the Cayley Graph associated with $Heis(Z_p)$ and generators $\{A, B\}$ denoted by $\mathcal{G}$.

Computing the hash values $H_1(m)$ by using the matrix multiplication modulo p indexed by m.Then $H_1(m) = \{(a, b, c) \ni \text{ for some } a, b, c \in Z_p\}$. That is, we use the generators of the discrete Heisenberg group to construct the whole hash values of $H_1(m)$. In this way we compute the matrix multiplication of a bit strings of length n, it is essential to attain 5n multiplications and 4n additions modulo p.

In [3] Z_p, each addition requires $\mathcal{O}(logp)$ and each multiplication requires $\mathcal{O}(log^2p)$ bit operations. Thus, if $p \approx 2^i$ for some i, each process modulo p is $\mathcal{O}(i^2)$.

Thus, the performance of the number of bit strings of length n to compute $H_1(m)$ is $\mathcal{O}(i^2.n)$.

Here all the executions are done in Z_p and since we are using the elementary generator matrix A or matrix B, we can reduce the multiplication and addition operations.Here we are dealing with the upper triangular matrices with main diagonal 1 so that, we need to attain 4n additions and n multiplications to hash a bit string of length n.

2.3.1 Observed values for Efficiency

Here we tabulated the average running time to execute the hash values $H_1(m)$ for 100 random binary strings of same length.In our observations we apply only the matrix multiplication for the pair of generators, $A = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ and $B = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$ for 100,200,400,800,1000,2000,5000 length of binary strings.The tests were executed in Intel(R) Core (TM) i3 2.00GHZ computer with 4GB of RAM using python 2.7.1 code in Table 2.1.

Hence, we determined that our defined hash function with large prime $(2^{256} - 1053)$ without any parallelism hashes to a 5000 length of bit strings in 0.083 seconds in average.

Table 2.1: Average running time of H_1

prime,p	Length of the binary strings						
	100	200	500	800	1000	2000	5000
$2^{127}-1$	0.002s	0.003s	0.009s	0.013s	0.018s	0.033s	0.084s
$2^{137}-555$	0.002s	0.003s	0.008s	0.013s	0.017s	0.033s	0.083s
$2^{147}-387$	0.002s	0.003s	0.009s	0.013s	0.017s	0.033s	0.084s
$2^{157}-213$	0.002s	0.003s	0.014s	0.013s	0.016s	0.033s	0.082s
$2^{167}-771$	0.002s	0.0035s	0.009s	0.013s	0.017s	0.033s	0.083s
$2^{177}-919$	0.002s	0.003s	0.019s	0.013s	0.018s	0.033s	0.084s
$2^{187}-477$	0.002s	0.003s	0.011s	0.013s	0.019s	0.033s	0.084s
$2^{197}-775$	0.002s	0.003s	0.009s	0.013s	0.021s	0.034s	0.083s
$2^{207}-429$	0.002s	0.003s	0.009s	0.013s	0.019s	0.033s	0.082s
$2^{217}-675$	0.002s	0.003s	0.009s	0.013s	0.021s	0.033s	0.082s
$2^{227}-721$	0.002s	0.003s	0.009s	0.013s	0.018s	0.033s	0.082s
$2^{237}-949$	0.002s	0.003s	0.009s	0.013s	0.016s	0.033s	0.087s
$2^{247}-309$	0.002s	0.003s	0.010s	0.014s	0.016s	0.033s	0.082s
$2^{256}-1053$	0.002s	0.003s	0.009s	0.013s	0.016s	0.033s	0.082s

2.4. Pseudo-Randomness

A good hash function must produce an output distribution as random as possible.Chi-square goodness-of-fit test is a non-parametric test that is used to get the observed elements of a given situation is importantly different from the expected value.Here we exercised the chi-square goodness-of-fit test for the output distribution of the hashed values.We know that the elements of Z_p are uniformly distributed [3],we tested using the chi-square test that the output distribution of hashed values are also uniform.

2.4.1 Chi-square test

The test is very important to check whether the observed frequencies of the given input data satisfies the expected frequencies of that data.

Usually, the output distribution of the hashed values for the corresponding input binary strings attained in our hash function must be distributed uniformly over its range.

To check the chi-square goodness of fit test, the following conditions should be true:

- The observed frequencies are attained from the randomly chosen binary strings. (i.e) our sample in this test should be random to obtain the observed frequencies.
- Expected frequencies in every distribution should be greater than or equal to 5.

The above conditions are satisfied then we can check that the chi-square distribution with k-1 degrees of freedom is, $\tilde{\chi}^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i}$,where O_i denotes the observed frequencies obtained from each input binary strings and E_i denotes the expected frequencies of the corresponding distributions of each category and k denotes the number of categories.
Null Hypothesis (H_0) is the Output distribution of the hashed values are uniformly distributed and level of significance $\alpha = 0.01$ is fixed for all the distributions.

In this test, the above conditions for the chi-square test,we choose input as random number of binary strings in the hash function H_1 to get the output as random distribution of a,b and c values from the corresponding matrices of the discrete Heisenberg group and if $p > 3$, then the expected frequencies for each category is greater than or equal to 5.
By the definition of $\tilde{\chi}^2_\alpha(k-1)$ the probability of rejecting H_0 when H_0 is true, is $P(\tilde{\chi}^2 > \tilde{\chi}^2_\alpha(k-1)) = \alpha$, which is a type-I error.

The test of various distribution of the hashed values for every prime p was performed 100 times and each time with new distribution of hashed values was checked.The average pass proportion of the 100 random distributions of a,b and c are 100%, which strongly says that hashed values of the hash function H_1 from random binary strings are uniformly distributed.Hence,the tests were executed in Intel(R) Core (TM) i3 2.00GHZ computer with 4GB of RAM using python (x,y) code.

2.5. Protecting against Different Attacks

2.5.1 Finding elements of small order

This attack is based on the structure of the Discrete Heisenberg group.It is performed by computing the value of hash function (message digest) of random numbers of binary strings.This attack is performed until obtaining a random output as hashed values of the hash function H_1, which is a diagonalizable matrix.Now we have to find the hashed values of small order for a particular binary string and it has to be applied number of times into the hash of another binary string. This is equivalent to insert the identity matrix in the hash value and consequently obtaining a collision with the corresponding binary strings of the messages.

In our case, the minimal polynomial has a repeated root.So that there is no diagonalizable matrix except the identity matrix in the group. So, this attack is inconsistent to find a matrix of such small order. The possibility of getting a matrix of small order is infinitesimal.

2.5.2 Symmetric Matrices

This attack is based on the generators of the group. The preference of generators in the group can affect the security aspects of the hashing algorithm. In our case, both the generators are not symmetric matrices. So, this attack is inconsistent.

2.5.3 Generic Attack

In [22, 68] this kind of contest, involves a Birthday Paradox (Brute Force) search for all binary strings of length contingent on the size of the chosen field.

In our case, the length of the colliding bits strings is at least 256 (if p is of order 2^{256}) and this way implies the brute force search over binary strings of length minimum 128, which is computationally inconsistent. So that the reduction of birthday paradox is just not sufficient to create the attack consistent with our recommended size of a prime p.

2.5.4 Subgroup Attack

This attack is obstructed by selecting the group cautiously.The minimal requirement is that the cardinality of group has a large factor [45]. In our case, the cardinality of our group is very large (if p is of order 2^{256}).

2.6. Features Comparison with other Hashing Algorithms

- Cryptographic hash functions like $SHA-256$ and $SHA-512$ have maximum input message size of $2^{64}-1$ and $2^{128}-1$ bits respectively [67]. But in our hash functions H_1 has arbitrary input message size to output the hashed value of fixed size which is the advantage to our hash function.

- Now compare the Tillich-Zemor hash function [46] to our hash function H_1 with the output size of the hashed value, if one uses p is order 2^{256}, then the size of the hashed values of Tillich-Zemor is 1024 bits while our H_1 is 768 bits which says that our hash function H_1 also has another advantage.

- The number of bit operations in SHA-3 is 2^{1600} bit operations while we used in our hash function H_1 is 4n additions and n multiplications, which will enhance the efficiency of the hash function H_1.

- According to [27], SHA-1 hashes approximately 153 MiB/seconds (MiB stands for mebibyte and 1MiB = 2^{20} bytes) and so this is approximately 8388600 bits per second.Our proposed hash function H_1 hashes 8388600 bits in 156 MiB/seconds for a large prime $p = 2^{256}-1053$ without any parallelization.Our proposed hash function is less efficient than other hash algorithms,because the efficiency can be further improved if we use the latest features in the computers.Here the tests were executed in Intel(R) Core (TM) i3 2.00GHZ computer with 4GB of RAM using python 2.7.1 code.

Hence our proposal is modest and efficient.It is performed in 4n additions and n multiplications over F_p to hash a bit strings of length n.This defined proposal of the hash function H_1 has the output size of length $3logp$ while other hash functions using matrices over F_p to hashes the output size of length $4logp$.Also the randomness of the output distribution in H_1 was tested by the Chi-Square: Goodness of fit test.Hence our proposal H_1 is resistant to the known attacks in Cayley hash functions which says that our defined H_1 has no visible threat to the security.

Chapter 3

Vectorial Version of the Cayley Hash Function

We defined the new version of the Cayley Hash Function using discrete Heisenberg group which gives solution to the weakness in the defined hash function H_1.The vectorial version of the hash function which we defined is very efficient and also as fast as the provably secure hash function H_1.We examine the Chi-Square Goodness of fit test,that the output distribution of the hashed values in the hash function H_2 are uniform, that is, the output distribution is uniformly distributed,which reduce the chance of occurring collisions.We compared the results with our hash function H_1 gives the in-depth study of both the functions.

3.1. Designing the Vectorial version of H_1

One essential weakness in the defined hash function H_1 is that for short messages the pre-images can be computed easily.

This weakness can be overcome by one solution is using padding method and the other solution is by using the vectorial form of H_1.

We defined a new vector form of the hash function $H_2 \colon \{0,1\}^* \to F_p^3$ as $H_2(m) = vec_{H_1}(m)H_1(vec_{H_1}(m) \oplus C)$ where $vec_{H_1}(m)$ is the first row of H_1 and C is the random binary strings.

Like $H_1(m)$, the function $H_2(m)$ also computed one bit at a time and every matrix-by-matrix multiplication can now be replaced by a vector-by-matrix multiplication which says that $H_2(m)$ is as fast as $H_1(m)$.Hence the function H_2 is very efficient. The second event of $H_1(vec_{H_1}(m) \oplus C)$ having only p additional vector-by-matrix multiplication for the bits of length n.It is insignificant for long messages.

Many bits of the hash values of $vec_{H_1}(m)$ and $vec_{H_1}(m^{'})$ are different.So $H_1(vec_{H_1}(m) \oplus C)$ and $H_1(vec_{H_1}(m^{'}) \oplus C)$ will also differ by many bits.Then $H_1(m)$ and $H_1(m^{'})$ are also completely different.

Without inverting vec_{H_1}, finding a pair $vec_{H_1}(m)$ and $vec_{H_1}(m^{'})$ are equal, for some particular messages m and $m^{'}$ which means they changed only in the last bit seems to be a hard problem and the messages m and $m^{'}$ with more than one bits are different then also $H_2(m)$ and $H_2(m^{'})$ are completely different.
Hence the hash function H_2 will not affect by the malleability properties like the hash values of two or more messages tends to a same output.

3.2. Execution of H_1 , H_2

Here we executed the different hashed values of H_1 and H_2 using different test vectors of different length for fixed prime p. The tests were executed in Intel(R) Core

(TM) i3 2.00GHZ computer with 4GB of RAM using python (x,y) code.

Table 3.1: Execution of H_1 , H_2

Message m	length of strings	H_1	H_2
“ ”	empty string	$\begin{pmatrix} 1 & 13 & 11 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{pmatrix}$	(1, 28, 75)
“abc”	44	$\begin{pmatrix} 1 & 28 & 29 \\ 0 & 1 & 16 \\ 0 & 0 & 1 \end{pmatrix}$	(1, 31, 737)
123456789	80	$\begin{pmatrix} 1 & 43 & 67 \\ 0 & 1 & 37 \\ 0 & 0 & 1 \end{pmatrix}$	(1, 89, 1128)
123456789“abc”	125	$\begin{pmatrix} 1 & 71 & 24 \\ 0 & 1 & 54 \\ 0 & 0 & 1 \end{pmatrix}$	(1, 118, 1728)
abcdefghijkl mnopqrstu-vwxyz	234	$\begin{pmatrix} 1 & 109 & 141 \\ 0 & 1 & 125 \\ 0 & 0 & 1 \end{pmatrix}$	(1, 155, 2976)
abcdefghijcdefg hidefghijk lmnopaaaaa abbbbb	386	$\begin{pmatrix} 1 & 211 & 104 \\ 0 & 1 & 175 \\ 0 & 0 & 1 \end{pmatrix}$	(1, 258, 5214)
bbbbbbbbbbbbb bbbbbbbbccccc ccccccccccccccc-ceeeeeeeeeeeee	521	$\begin{pmatrix} 1 & 279 & 444 \\ 0 & 1 & 242 \\ 0 & 0 & 1 \end{pmatrix}$	(1,325, 7294)

3.3. Security Properties

The following propositions says that the H_2 is collision resistant if the original function H_1 is collision resistant.

Proposition 3.3.1. *Let $m, m^{'}$ be the two distinct messages.If H_1 is collision resistant then H_2 is also collision resistant.*

Proof. Let $m \neq m^{'}$ be the two messages. Then,

$$\begin{aligned} H_1(m) = H_1(m^{'}) &\Rightarrow \prod_{i=1}^{n} \pi(m_i) = \prod_{i=1}^{n} \pi(m_i^{'}) \\ &\Rightarrow vec_{H_1}(m) = vec_{H_1}(m^{'}) \end{aligned} \tag{3.1}$$

If we choose, $m_1 = vec_{H_1}(m) \oplus C_1 \neq vec_{H_1}(m^{'}) \oplus C_2 = m_2$, then by our hypothesis,

$$H_1(vec_{H_1}(m) \oplus C_1) = H_1(vec_{H_1}(m^{'}) \oplus C_2) \tag{3.2}$$

Multiplying (3.1) and (3.2), we get,

$$\begin{aligned} vec_{H_1}(m)H_1(vec_{H_1}(m) \oplus C_1) &= vec_{H_1}(m^{'})H_1(vec_{H_1}(m^{'}) \oplus C_2) \\ H_2(m) &= H_2(m^{'}) \end{aligned}$$

■

Proposition 3.3.2. *Let $m, m^{'}$ be the two distinct messages.If H_2 is collision resistant with $vec_{H_1}(m) = vec_{H_1}(m^{'})$ then H_1 is also collision resistant.*

Proof. Let $m \neq m^{'}$ be the two messages.

Since by our hypothesis we have,

$$\begin{aligned} H_2(m) = H_2(m^{'}) &\Rightarrow vec_{H_1}(m)H_1(vec_{H_1}(m) \oplus C_1) = vec_{H_1}(m^{'})H_1(vec_{H_1}(m^{'}) \oplus C_2) \\ &\Rightarrow H_1(vec_{H_1}(m) \oplus C_1) = H_1(vec_{H_1}(m^{'}) \oplus C_2) \end{aligned}$$

For our convenience, we represent $H_1(m) = H_1(m^{'})$ where, $m = vec_{H_1}(m) \oplus C_1$ and $m^{'} = vec_{H_1}(m^{'}) \oplus C_2$ ■

3.4. Efficiency of the Hash function H_2

Here we tabulated the average running time to execute the hash values $H_2(m)$ for 100 random binary strings of same length in Table 3.2. In our observations, we apply the vector-by-matrix multiplication for different length of binary strings. We completed all these calculation using python (x,y) program code in Intel(R) Core (TM) i3 2.00GHZ computer with 4GB of RAM. We determined that our proposed

Table 3.2: Average running time to execute H_2

p	Number of Binary Strings						
	100	200	400	800	1000	2000	5000
$2^{127} - 1$	0.022s	0.023s	0.029s	0.032s	0.034s	0.051s	0.100s
$2^{137} - 555$	0.021s	0.024s	0.028s	0.034s	0.034s	0.051s	0.100s
$2^{147} - 387$	0.023s	0.026s	0.027s	0.031s	0.038s	0.051s	0.099s
$2^{157} - 213$	0.022s	0.025s	0.028s	0.031s	0.035s	0.051s	0.099s
$2^{167} - 771$	0.023s	0.024s	0.029s	0.031s	0.034s	0.050s	0.098s
$2^{177} - 919$	0.022s	0.026s	0.027s	0.031s	0.035s	0.050s	0.100s
$2^{187} - 477$	0.021s	0.028s	0.030s	0.032s	0.034s	0.052s	0.099s
$2^{197} - 775$	0.021s	0.027s	0.027s	0.031s	0.034s	0.050s	0.103s
$2^{207} - 429$	0.020s	0.027s	0.026s	0.032s	0.034s	0.051s	0.105s
$2^{217} - 675$	0.021s	0.026s	0.026s	0.031s	0.037s	0.050s	0.098s
$2^{227} - 721$	0.020s	0.024s	0.026s	0.031s	0.034s	0.050s	0.106s
$2^{237} - 949$	0.021s	0.025s	0.027s	0.031s	0.035s	0.050s	0.100s
$2^{247} - 309$	0.021s	0.023s	0.026s	0.031s	0.034s	0.050s	0.099s
$2^{256} - 1053$	0.02s	0.022s	0.028s	0.033s	0.034s	0.051s	0.100s

hash function H_2 with large prime $(2^{256} - 1053)$ without any parallelism, hashes to a 5000 length of bit strings in 0.100 seconds in average.

3.5. Pseudo-Randomness

3.5.1 Chi-Square Test

The output distribution of the hashed values of the hash functions are uniformly distributed then we say that the hash function is of high quality.The purpose of the Chi-Square Goodness-of-Fit test is to find out the randomness in the output distribution of the hashed values in the hash function.

We know that the Output distribution of the hashed value of the hash function H_1 is uniformly distributed. We need to check vectorial version H_2 of the provably secure hash function H_1, output distributions are also uniformly distributed.The efficiency of this test is to find out whether the observed frequencies of the given input binary strings satisfies the expected frequencies of that input.

The following conditions are satisfied to check the chi-square goodness of fit test

(i) Observed frequencies are the hashed values of the randomly chosen messages (input binary strings).

(ii) Expected frequencies in every distribution should be greater than or equal to 5.

Using the Python (x, y) code,the test is conducted for the random distributions of hashed values from random binary strings are uniform. The test of various distribution of the hashed values was performed 100 times and each time with new distribution of hashed values was checked.The average pass proportion of the 100 random distributions of a and b are 100%, which strongly says that hashed values of the hash function H_2 from random binary strings are uniformly distributed.

3.6. Comparison of H_1 and H_2

Here we are doing the comparison of H_1 and H_2 which helps us to understand more about the designs, efficiency and also the security aspects of both hash functions.

Table 3.3: Comparison of the Hash Functions H_1 and H_2

Hash function H_1	**Hash function** H_2
One bit at a time to compute matrix-by-matrix multiplication	One bit at a time and replace the matrix-by-matrix multiplication by vector-by-matrix multiplication
The function H_1 is associated to both Algebraic and Graph theoretical problems	This vectorial function H_2 also related to both Algebraic and Graph theoretical problems
$H_1(m)$ needs to attain $4n$ additions and n multiplications to hash a bit strings of length n	$H_2(m)$ has p-additional vector-by-matrix multiplication of bit strings of length n, which is negligible for long messages
In our provably secure hash function H_1,for large prime $(2^{256}-1053)$ without any parallelism hashes to a 10^6 length of bit strings in 17.63 seconds	In this H_2 without any parallelism hashes to a 10^6 length of bit strings in 18.24 seconds for large prime $(2^{256}-1053)$
Output distribution of the hashed values of the hash function are uniformly distributed	Hashed Values in the output distribution of H_2 from random binary strings are uniform
If the order of the prime is 2^{256} then there is no collision for the hash function H_1, except the length of atleast one of the colliding bit strings is atleast 256	Without inverting vec_{H_1}, finding a pair $vec_{H_1}(m)$ and $vec_{H_1}(m^{'})$ very close to each other for some particular messages m and $m^{'}$ seems to be a hard problem, which says that $H_2(m)$ and $H_2(m^{'})$ are completely different. This will give that our defined vectorial version of the hash function H_2 are collision resistant.

3.7. Modified Form of the Hash Function H_1

One essential weakness in the hash function H_1 for short messages has the solution by changing the matrix form to a vector form H_2 and this H_2 is safer than H_1 particularly for the short messages.This new idea will make the mathematical problem, especially factorization problems more harder and hold an efficient way to impose limits on the type of factorisations for attacking H_1.

Definition 3.7.1. Let an integer, $g > 1$.Let $C \in Heis(F_p) - \{I, A, B\}$, where I is the identity element of $Heis(F_p)$. Define $H_3 : \{0,1\}^* \rightarrow Heis(F_p)$ by $H_3(m) = \prod_{i=1}^{n} D_i$ where

$$D_i = \begin{cases} \pi(b_i) & \text{if } g \nmid i \\ \pi(b_i)C & \text{if } g \mid i \end{cases}$$

3.7.1 Security Properties

First, we say that H_3 is atleast as secure as H_1

Proposition 3.7.2. *If we know a message b, such that $H_1(b) = h$, then we can efficiently find a message $x^{'}$, for every message x such that $H_3(x) = H_1(x^{'})$.*

Proof. Let b be a binary string such that $H_1(b) = h$.Let $x = x_1x_2.....x_n$
and assume $x^{'} = x_1x_2....x_gbx_{g+1}.....x_{2g}bx_{2g+1}.....x_n$.
Then we can simply says that $H_3(x) = H_1(x^{'})$ ■

Theorem 3.7.3. *Breaking the pre-image and collision resistance of H_3 respectively leads to breaking the pre-image and collision resistance of H_1.*

Proof. Let $h \in Heis(F_p) - \{I, A, B\}$ then $H_3(m) = h$.By the Proposition 3.7.2, there exist $m^{'}$ such that $H_1(m^{'}) = H_3(m) = h$.Hence it says that pre-image resistant of H_3

gives the pre-image resistant of H_1.Thus, the first part of the theorem holds.

Second Part of the theorem proves that, let $m \neq m_1$ with $H_1(m) = H_3(m_1)$.By proposition 3.7.2, there exist $m^{'}$ and $m^{''}$ such that $H_3(m) = H_1(m^{'})$ and $H_3(m_1) = H_1(m^{''})$. So $H_1(m^{'}) = H_3(m) = H_3(m_1) = H_1(m^{''})$.Thus the collision for H_3 is also the collision for H_1. ■

Now we assert that the hash function H_3 is safer than hash function H_1. Let $g < log_2(p)$, proposition 2.2.3, gives that there is no collision of length $< g$ for the hash function H_1, which says that the collision for H_1 cannot be directly used to find a collision for H_3.There are many attacks in [58] that are based on the factorization of elements over F_p are not practical in our hash function H_3, because they need to find a new special form of factorization as
$\pi(m_{i_1})\pi(m_{i_2})\pi(m_{i_3})\ldots\ldots\ldots\pi(m_{i_g})C\pi(m_{i_{g+1}})\pi(m_{i_{g+2}})\ldots\ldots\ldots\pi(m_{i_{2g}})C\pi(m_{i_{2g+1}})\ldots\ldots\ldots$

3.8. Efficiency

Suppose we take an input binary strings of length n with a prime number p (is of order 2^{256}) and fix g (for example $g = \lfloor log_2(p) \rfloor - 1$) and we use $C \in Heis(F_p) - \{I, A, B\}$ as fixed matrix (or we can assume C as one of the smallest hashed value of binary strings of length $l < n$).
Using these above parameters, we can compute the hashed value of H_3 of length n with $\lfloor n/g \rfloor$ matrix multiplications which is less than H_1, since it has n matrix multiplication.
Here, in all the observations, we apply the H_3 for different length of binary strings. We did all these calculation using python (x,y) program code in Intel(R) Core(TM)

i3 2.00GHZ computer with 4GB of RAM.

Table 3.4: Average running time to execute the Hash function H_3

p	Number of Binary Strings					
	100	500	800	1000	2000	5000
$2^{127}-1$	0.002s	0.01s	0.023s	0.03s	0.044s	0.09s
$2^{137}-555$	0.002s	0.01s	0.021s	0.03s	0.046s	0.09s
$2^{147}-387$	0.002s	0.01s	0.023s	0.03s	0.04s	0.10s
$2^{157}-213$	0.002s	0.01s	0.023s	0.03s	0.04s	0.09s
$2^{167}-771$	0.002s	0.01s	0.023s	0.03s	0.046s	0.09s
$2^{177}-919$	0.002s	0.01s	0.023s	0.03s	0.046s	0.09s
$2^{187}-477$	0.002s	0.01s	0.023s	0.03s	0.04s	0.09s
$2^{197}-775$	0.002s	0.01s	0.023s	0.03s	0.04	0.09s
$2^{207}-429$	0.002s	0.01s	0.023s	0.03s	0.044s	0.09s
$2^{217}-675$	0.002s	0.01s	0.021s	0.03s	0.046s	0.09s
$2^{227}-721$	0.002s	0.01s	0.035s	0.03s	0.045s	0.09s
$2^{237}-949$	0.002s	0.01s	0.034s	0.03s	0.04s	0.09s
$2^{247}-309$	0.002s	0.01s	0.033s	0.03s	0.04s	0.09s
$2^{256}-1053$	0.002s	0.01s	0.032s	0.03s	0.05s	0.09s

Thus, our defined hash function H_3 with large prime $(2^{256}-1053)$ without any parallelism hashes to a 5000 length of bit strings in 0.09 seconds in average.

Hence, we overcome the weakness in the hash function H_1 by introducing the vector form of the hash function H_2, which is very efficient and requires only p-additional vector-by-matrix multiplication,which is negligible for long messages.The modified form of H_1 is defined as H_3, which gives the factorisation problem more harder and there is no known factorisation attacks of this type.Both H_2 and H_3 is atleast as secure as the hash function H_1.

Chapter 4

Neighbourhoods in $\mathbb{P}^2$

Using the Metric, we proved that the elements of $GL_3(\mathbb{O})$ act by isometrics on $\mathbb{P}^2$. We defined the ϵ-Neighbourhood of an element in $\mathbb{P}^2$ and we picture out how the neighbourhoods related to metric will look like in $\mathbb{P}^2$. We also proved some results in Neighbourhoods in $\mathbb{P}^2$.

The following shows that $GL_3(\mathbb{O})$ acts transitively on $\mathbb{P}^2$.

Lemma 4.0.1. *Suppose that $[v] \in \mathbb{P}^2$ then there exists an element $g \in GL_3(\mathbb{O})$ such that $g.[v] = [e_3]$.*
In other words, shows that $GL_3(\mathbb{O})$ acts transitively on $\mathbb{P}^2$

Proof. Let $v = (v_1, v_2, v_3) \in V$ and assume that $v \neq (0, 0, 0)$
Notice if $max\{|v_1|, |v_2|, |v_3|\} = |v_1| = p^l$
$\Longrightarrow |x^l v_1| = 1$
$\Longrightarrow x^l v_1 \in \mathbb{O}^*$ and $|x^l v_2| \leqslant 1, |x^l v_3| \leqslant 1$ which says that $x^l v_2, x^l v_3 \in \mathbb{O}$.
Similarly, for $max\{|v_1|, |v_2|, |v_3|\} = |v_2| = p^l$
$\Longrightarrow |x^l v_2| = 1$

$\Longrightarrow x^l v_2 \in \mathbb{O}^*$ and $|x^l v_1| \leqslant 1, |x^l v_3| \leqslant 1$ which says that $x^l v_1, x^l v_3 \in \mathbb{O}$.

and also, for $max\{|v_1|, |v_2|, |v_3|\} = |v_3| = p^l$

$\Longrightarrow |x^l v_3| = 1$

$\Longrightarrow x^l v_3 \in \mathbb{O}^*$ and $|x^l v_1| \leqslant 1, |x^l v_2| \leqslant 1$ which says that $x^l v_1, x^l v_2 \in \mathbb{O}$.

Thus we have $[v] = [x^l v] = [x^l v_1 : x^l v_2 : x^l v_3]$

Assume that there exist a representative v of $[v]$ such that $[v] = [\alpha : \beta : \gamma]$

for some $\alpha, \beta, \gamma \in \mathbb{O}$ and either $\alpha \in \mathbb{O}^*$ or $\beta \in \mathbb{O}^*$ or $\gamma \in \mathbb{O}^*$ or all the three in $\mathbb{O}^*$.

If $\alpha \in \mathbb{O}^*$, then $g = \begin{pmatrix} \beta & -\alpha & 0 \\ \gamma & 0 & -\alpha \\ \alpha^{-1} & 0 & 0 \end{pmatrix} \in GL_3(\mathbb{O})$, $\alpha \neq 0$, then $g.[v] = [e_3]$

If $\beta \in \mathbb{O}^*$, then $g = \begin{pmatrix} -\beta & \alpha & 0 \\ 0 & \gamma & -\beta \\ 0 & \beta^{-1} & 0 \end{pmatrix} \in GL_3(\mathbb{O})$, $\beta \neq 0$, then $g.[v] = [e_3]$

If $\gamma \in \mathbb{O}^*$, then $g = \begin{pmatrix} \gamma & 0 & \alpha \\ 0 & -\gamma & \beta \\ 0 & 0 & \gamma^{-1} \end{pmatrix} \in GL_3(\mathbb{O})$, $\gamma \neq 0$, then $g.[v] = [e_3]$

If $\alpha, \beta, \gamma \in \mathbb{O}^*$, then $g = \begin{pmatrix} \alpha^{-1} & \beta^{-1} & 0 \\ -\gamma & 0 & \alpha \\ \beta & -\alpha & \gamma^{-1} \end{pmatrix} \in GL_3(\mathbb{O})$, $\alpha, \beta, \gamma \neq 0$,

then $g.[v] = [e_3]$ ■

4.1. Metric in $\mathbb{P}^2$

The lemma shows that ∞-norm is invariant under the action of elements of $GL_3(\mathbb{O})$ on V.

Lemma 4.1.1. *Let $g \in GL_3(\mathbb{O})$. Then $\| g.u \| = \| u \|$, for all $u = (u_1, u_2, u_3) \in V$.*

Proof. Let $g = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \in GL_3(\mathbb{O})$ and let $u = (u_1, u_2, u_3)$

then, $g.u = \begin{pmatrix} a_{11}u_1 + a_{12}u_2 + a_{13}u_3 \\ a_{21}u_1 + a_{22}u_2 + a_{23}u_3 \\ a_{31}u_1 + a_{32}u_2 + a_{33}u_3 \end{pmatrix}$

Since,the absolute value is non-Archimedean triangular inequality then we have,

$$\begin{aligned} \| g.u \| &= \| (a_{11}u_1 + a_{12}u_2 + a_{13}u_3, a_{21}u_1 + a_{22}u_2 + a_{23}u_3, a_{31}u_1 + a_{32}u_2 + a_{33}u_3) \| \\ &= max\{|a_{11}u_1 + a_{12}u_2 + a_{13}u_3|, |a_{21}u_1 + a_{22}u_2 + a_{23}u_3|, \\ &\quad |a_{31}u_1 + a_{32}u_2 + a_{33}u_3|\} \\ &\leqslant max\{max\{|a_{11}u_1|, |a_{12}u_2|, |a_{13}u_3|\}, max\{|a_{21}u_1|, |a_{22}u_2|, |a_{23}u_3|\}, \\ &\quad max\{|a_{31}u_1|, |a_{32}u_2|, |a_{33}u_3|\}\} \end{aligned} \tag{4.1}$$

We know that for $a \in \mathbb{O}$ then $|a| \leqslant 1$ which gives $|af| \leqslant |f|$ for all $f \in F_p((x))$.
Hence equation (4.1) becomes,

$$\begin{aligned} \| g.u \| &\leqslant max\{max\{|u_1|, |u_2|, |u_3|\}, max\{|u_1|, |u_2|, |u_3|\}, max\{|u_1|, |u_2|, |u_3|\}\} \\ &\leqslant max\{|u_1|, |u_2|, |u_3|\} = \| u \| \end{aligned} \tag{4.2}$$

since $g \in GL_3(\mathbb{O})$ then g^{-1} exist that also satisfies equation (4.2) as,
$\| g^{-1}.v \| \leqslant \| v \|$ for all $v \in V$.

Let take $v = g.u$ which gives the inequality,

$$\| u \| \leqslant \| g.u \| \tag{4.3}$$

From (4.2) and (4.3) $\| g.u \| = \| u \|$ ■

Our distance is invariant under the action of elements of $GL_3(\mathbb{O})$ on $\mathbb{P}^2$

Lemma 4.1.2. *Let $g \in GL_3(\mathbb{O})$ then $d([u],[v]) = d(g.[u], g.[v])$ for any $[u],[v] \in \mathbb{P}^2$. In other words, the elements of $GL_3(\mathbb{O})$ act isometrics on $\mathbb{P}^2$.*

Proof. Let $g = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \in GL_3(\mathbb{O})$ and

let $[u] = [u_1 : u_2 : u_3]$ and $[v] = [v_1 : v_2 : v_3]$ then

$g.[u] = [a_{11}u_1 + a_{12}u_2 + a_{13}u_3 : a_{21}u_1 + a_{22}u_2 + a_{23}u_3 : a_{31}u_1 + a_{32}u_2 + a_{33}u_3]$

$g.[v] = [a_{11}v_1 + a_{12}v_2 + a_{13}v_3 : a_{21}v_1 + a_{22}v_2 + a_{23}v_3 : a_{31}v_1 + a_{32}v_2 + a_{33}v_3]$

Now, $\| g.[u] \wedge g.[v] \| = max\{|A|, |B|, |C|\}$ where,

$$\begin{aligned} |A| &= |(a_{21}a_{32} - a_{31}a_{22})(u_1v_2 - u_2v_1) + (a_{21}a_{33} - a_{31}a_{23})(u_1v_3 - u_3v_1) + \\ &\quad (a_{22}a_{33} - a_{32}a_{23})(u_2v_3 - u_3v_2)|, \end{aligned}$$

$$\begin{aligned} |B| &= |(a_{31}a_{12} - a_{11}a_{32})(u_1v_2 - u_2v_1) + (a_{13}a_{31} - a_{11}a_{33})(u_1v_3 - u_3v_1) + \\ &\quad (a_{32}a_{13} - a_{12}a_{33})(u_2v_3 - u_3v_2)|, \end{aligned}$$

$$|C| = |(a_{11}a_{22} - a_{12}a_{21})(u_1v_2 - u_2v_1) + (a_{11}a_{23} - a_{13}a_{22})(u_1v_3 - u_3v_1) + (a_{12}a_{23} - a_{13}a_{22})(u_2v_3 - u_3v_2)|$$

Hence,$max\{|A|, |B|, |C|\}$ is either greater than or equal to $max_{i,j}\{|u_iv_j - u_jv_i|\}$ or less than or equal to $max_{i,j}\{|u_iv_j - u_jv_i|\}$

Case(i): $max\{|A|, |B|, |C|\} \geqslant max_{i,j}\{|u_iv_j - u_jv_i|\}$

$$\begin{aligned} \| g.[u] \wedge g.[v] \| &= max\{|A|, |B|, |C|\} \\ &\geqslant max_{i,j}\{|u_iv_j - u_jv_i|\} = \| u \wedge v \| \end{aligned} \tag{4.4}$$

Let $g \in GL_3(\mathbb{O})$ then g^{-1} exist that also satisfies equation (4.4) as,
$\| g^{-1}.[x] \wedge g^{-1}.[y] \| \geqslant \| [x] \wedge [y] \|$, for $[x], [y] \in \mathbb{P}^2$
Now take $[x] = g.[u]$ and $[y] = g.[v]$ then,

$$\| [u] \wedge [v] \| \geqslant \| g.[u] \wedge g.[v] \| \tag{4.5}$$

From (4.4) and (4.5) we have,

$$\| g.[u] \wedge g.[v] \| = \| [u] \wedge [v] \| \tag{4.6}$$

Case(ii): $max\{|A|, |B|, |C|\} \leqslant max_{i,j}\{|u_iv_j - u_jv_i|\}$

$$\begin{aligned} \| g.[u] \wedge g.[v] \| &= max\{|A|, |B|, |C|\} \\ &\leqslant max_{i,j}\{|u_iv_j - u_jv_i|\} \\ &= \| u \wedge v \| \end{aligned} \tag{4.7}$$

Let $g \in GL_3(\mathbb{O})$ then g^{-1} exist that also satisfies equation (4.7) as,

$\| g^{-1}.[x] \wedge g^{-1}.[y] \| \leqslant \| [x] \wedge [y] \|$,for $[x], [y] \in \mathbb{P}^2$

Now take $[x] = g.[u]$ and $[y] = g.[v]$ then,

$$\| [u] \wedge [v] \| \leqslant \| g.[u] \wedge g.[v] \| \tag{4.8}$$

From (4.7) and (4.8) we have,

$$\| g.[u] \wedge g.[v] \| = \| [u] \wedge [v] \| \tag{4.9}$$

We know that, $\| g.[u] \| = \| u \|$ and $\| g.[v] \| = \| v \|$.

Hence in both (4.6) and (4.9) we get,

$$\begin{aligned} d(g.[u], g.[v]) &= \frac{\| g.[u] \wedge g.[v] \|}{\| g.[u] \| \| g.[v] \|} \\ &= \frac{\| [u] \wedge [v] \|}{\| u \| \| v \|} \\ &= d([u], [v]) \end{aligned}$$

■

Remark. (i) We used this result with lemma 4.0.1 in the remaining proofs.

(ii) We know that the field F is Non-Archimedean then the metric is Ultra-Metric.

That is, $d([u], [w]) \leqslant max\{d([u], [v]), d([v], [w])\}$, for any $[u], [v], [w] \in \mathbb{P}^2$

Lemma 4.1.3. *If* $[u], [v] \in \mathbb{P}^2$ *then* $0 \leqslant d([u], [v]) \leqslant 1$ *and also, if* $d([u], [v]) \neq 0$ *then* $d([u], [v]) = \dfrac{1}{p^d}$ *for some* $d \in \mathbb{N}_0$

Proof. Let $[u], [v] \in \mathbb{P}^2$ then by lemma 4.0.1, we assume that $[v] = [e_3]$.

Fix $[u] = [a : b : c]$ then,

$$d([u], [v]) = d([a : b : c], [0 : 0 : 1]) \quad = \quad \frac{|a|}{max\{|a|, |b|, |c|\}} \tag{4.10}$$

We know that $d([u], [v]) \geqslant 0$.

Now, if $max\{|a|, |b|, |c|\} = |a|$ then (4.10) becomes, $d([u], [v]) = 1$ or

if $max\{|a|, |b|, |c|\} = |b|$ then (4.10) becomes, $d([u], [v]) < 1$ or

if $max\{|a|, |b|, |c|\} = |c|$ then (4.10) becomes, $d([u], [v]) < 1$.

Hence in all cases we have $0 \leqslant d([u], [v]) \leqslant 1$.

Now if $d([u], [v]) \neq 0$ then by (4.10) and all the three above cases, we get the absolute value $|.|$ is always zero or a power of p.

So, the distance $d([u], [v])$ is always zero or non-positive power of p.

Hence, $d([u], [v]) = \dfrac{1}{p^d}$ for some $d \in \mathbb{N}_0$ ■

4.2. Neighbourhoods in $\mathbb{P}^2$

With respect to our metric, we defined the ϵ-Neighbourhoods in $\mathbb{P}^2$ and visualize the neighbourhoods elements in $\mathbb{P}^2$.

Definition 4.2.1. [38] The set $N([u],\epsilon) = \{[v] \in \mathbb{P}^2 \ni d([u],[v]) \leqslant \epsilon\}$ is the ϵ-Neighbourhood of $[u]$ in $\mathbb{P}^2$.

Remark. (i) From the definition we notice that $d \in \mathbb{N}_0$ such that $N([u],\epsilon) = N([u],\frac{1}{p^d})$

(ii) We know that the representative of any element in $\mathbb{P}^2 - \{[e_3]\}$ has the form $[1:g:h]$ and $[x:1:z]$. Thus, consider these elements along with $[e_3]$ for the Neighbourhoods in $\mathbb{P}^2$

Proposition 4.2.2. *Let* $[1:g:h],[g:1:h] \in \mathbb{P}^2$ *and let* $d \in \mathbb{N}_0$ *then*

$$N([1:g:h],\frac{1}{p^{d+1}}) = \begin{cases} \{[1:a:b] \ni d([1:g:h],[1:a:b]) \leqslant \frac{1}{p^{d+1}}\} & \text{if } |g|,|h| \leqslant 1 \leqslant p^d \\ \{[1:a:b] \ni d([1:g:h],[1:a:b]) \leqslant \frac{1}{p^{d+1}}\} & \text{if } 1 < |g|,|h| \leqslant p^d \\ \{[1:a:b] \ni |a| > p^d\} \cup \{[e_3]\} & \text{if } |g|,|h| > p^d \end{cases}$$

$$N([g:1:h],\frac{1}{p^{d+1}}) = \begin{cases} \{[a:1:b] \ni d([g:1:h],[a:1:b]) \leqslant \frac{1}{p^{d+1}}\} & \text{if } |g|,|h| \leqslant 1 \leqslant p^d \\ \{[a:1:b] \ni d([g:1:h],[a:1:b]) \leqslant \frac{1}{p^{d+1}}\} & \text{if } 1 < |g|,|h| \leqslant p^d \\ \{[1:a:b] \ni |a| > p^d\} \cup \{[e_3]\} & \text{if } |g|,|h| > p^d \end{cases}$$

Additionally, $N([e_3],\frac{1}{p^{d+1}}) = \{[1:a:b] \ni |a| > p^d\} \cup \{[e_3]\}$ *if* $|g|,|h| > p^d$.

We prove the proposition as the following three lemmas

Lemma 4.2.3. *Let* $[1:g:h], [g:1:h] \in \mathbb{P}^2$ *and let* $d \in \mathbb{N}_0$ *then*

(i) If $|g|, |h| \leqslant 1 \leqslant p^d$ *then*

$N([1:g:h], \frac{1}{p^{d+1}}) = \{[1:a:b] \ni d([1:g:h], [1:a:b]) \leqslant \frac{1}{p^{d+1}}\}$

for all a, b *with the property* $|a| \leqslant p^d, |b| \leqslant p^d$

(ii) If $|g|, |h| \leqslant 1 \leqslant p^d$ *then*

$N([g:1:h], \frac{1}{p^{d+1}}) = \{[a:1:b] \ni d([g:1:h], [a:1:b]) \leqslant \frac{1}{p^{d+1}}\}$

for all a, b *with the property* $|a| \leqslant p^d, |b| \leqslant p^d$

Proof. (i) We know that $d([1:g:h], [1:a:b]) = \dfrac{max\{|gb-ah|, |h-b|, |a-g|\}}{max\{1, |g|, |h|\} max\{1, |a|, |b|\}}$

Since $|g|, |h| \leqslant 1 \leqslant p^d$ then $max\{1, |g|, |h|\} = 1$

Now, assume that $|a| \leqslant 1$, $|b| \leqslant 1$

Case(i): Suppose $|a| > 1, |b| \leqslant 1$ then

$max\{1, |a|, |b|\} = |a|$ and

$max\{|gb-ah|, |h-b|, |a-g|\} = max\{|gb-ah|, |h-b|, |a|\} > 1$

(since $|a-g| = |a|$,$|gb-ah| = |ah| \leqslant |a|$ or $|gb-ah| = |gb| \leqslant 1$ and

$|h-b| \leqslant 1$). Hence,

$$\begin{aligned} d([1:g:h], [1:a:b]) &= \frac{max\{|gb-ah|, |h-b|, |a-g|\}}{max\{1, |g|, |h|\} max\{1, |a|, |b|\}} \\ &> \frac{1}{1.|a|} \\ &> \frac{1}{p^d} \geqslant \frac{1}{p^{d+1}} \\ &\quad (\because |a| \leqslant p^d) \end{aligned}$$

Case(ii): Suppose $|a| \leqslant 1, |b| > 1$ then

$max\{1, |a|, |b|\} = |b|$ and

$max\{|gb-ah|, |h-b|, |a-g|\} = max\{|gb-ah|, |b|, |a-g|\} > 1$

(since $|h-b| = |b|, |gb-ah| = |gb| \leqslant |b|$ or $|gb-ah| = |ah| \leqslant 1$ and

$|a-g| \leqslant 1$)

Hence,

$$\begin{aligned} d([1:g:h],[1:a:b]) &= \frac{max\{|gb-ah|, |h-b|, |a-g|\}}{max\{1,|g|,|h|\}max\{1,|a|,|b|\}} \\ &> \frac{1}{1.|b|} \\ &> \frac{1}{p^d} \geqslant \frac{1}{p^{d+1}} \\ &\quad (\because |b| \leqslant p^d) \end{aligned}$$

Case(iii): Suppose $|a| > 1, |b| > 1$ then

$max\{1, |a|, |b|\} = max\{|a|, |b|\}$ and

$max\{|gb-ah|, |h-b|, |a-g|\} = max\{|gb-ah|, |a|, |b|\} > 1,$

(since,$|a-g| = |a|$,$|h-b| = |b|$ and $|gb-ah| = |gb| \leqslant |b|$ or

$|gb-ah| = |ah| \leqslant |a|$).

Hence we have,

$$\begin{aligned} d([1:g:h],[1:a:b]) &= \frac{max\{|gb-ah|, |h-b|, |a-g|\}}{max\{1,|g|,|h|\}max\{1,|a|,|b|\}} \\ &> \frac{1}{max\{|a|,|b|\}} = \frac{1}{|a|} or \frac{1}{|b|} \\ &> \frac{1}{p^d} \geqslant \frac{1}{p^{d+1}} \\ &\quad (\because |a|, |b| \leqslant p^d) \end{aligned}$$

Thus in all the three cases, we get contradiction that $[1 : a : b]$ is not in the

$N([1:g:h],\frac{1}{p^{d+1}})$.

Hence, $|a| \leqslant 1$, $|b| \leqslant 1$, $|g|, |h| \leqslant 1 \leqslant p^d$, then

$d([1:g:h],[1:a:b]) \leqslant \frac{1}{p^{d+1}}$ which says that $[1:a:b] \in N([1:g:h],\frac{1}{p^{d+1}})$

(ii) We know that $d([g:1:h],[a:1:b]) = \frac{max\{|b-h|,|ah-gb|,|g-a|\}}{max\{1,|g|,|h|\}max\{1,|a|,|b|\}}$

since, $|g|, |h| \leqslant 1 \leqslant p^d$ then $max\{1,|g|,|h|\} = 1$

Now, assume that $|a| \leqslant 1$, $|b| \leqslant 1$

Case(i): Suppose $|a| > 1, |b| \leqslant 1$ then

$max\{1,|a|,|b|\} = |a|$ and

$max\{|b-h|,|ah-gb|,|g-a|\} = max\{|b-h|,|ah-gb|,|a|\} > 1$

(since,$|b-h| \leqslant 1$, $|g-a| = |a|$ and $|ah-gb| = |ah| \leqslant |a|$ or

$|ah-gb| = |gb| \leqslant 1$).

Hence,

$$\begin{aligned} d([g:1:h],[a:1:b]) &= \frac{max\{|b-h|,|ah-gb|,|g-a|\}}{max\{1,|g|,|h|\}max\{1,|a|,|b|\}} \\ &> \frac{1}{1.|a|} > \frac{1}{p^d} \geqslant \frac{1}{p^{d+1}} \\ &\quad (\because |a| \leqslant p^d) \end{aligned}$$

Case(ii): Suppose $|a| \leqslant 1, |b| > 1$ then

$max\{1,|a|,|b|\} = |b|$ and

$max\{|b-h|,|ah-gb|,|g-a|\} = max\{|b|,|ah-gb|,|g-a|\} > 1$

(since,$|b-h| = |b|$, $|g-a| \leqslant 1$ and $|ah-gb| = |ah| \leqslant 1$ or

$|ah-gb| = |gb| \leqslant |b|$)

Hence,

$$\begin{aligned} d([g:1:h],[a:1:b]) &= \frac{max\{|b-h|,|ah-gb|,|g-a|\}}{max\{1,|g|,|h|\}max\{1,|a|,|b|\}} \\ &> \frac{1}{1.|b|} > \frac{1}{p^d} \geqslant \frac{1}{p^{d+1}} \\ &\quad (\because |b| \leqslant p^d) \end{aligned}$$

Case(iii): Suppose $|a|>1, |b|>1$ then $max\{1,|a|,|b|\} = max\{|a|,|b|\}$ and $max\{|b-h|,|ah-gb|,|g-a|\} = max\{|b|,|ah-gb|,|a|\} > 1$.
(since,$|b-h| = |b| > 1, |g-a| = |a| > 1$ and $|ah-gb| = |ah| \leqslant |a|$ or $|ah-gb| = |gb| \leqslant |b|$). Hence we have,

$$\begin{aligned} d([g:1:h],[a:1:b]) &= \frac{max\{|b-h|,|ah-gb|,|g-a|\}}{max\{1,|g|,|h|\}max\{1,|a|,|b|\}} \\ &> \frac{1}{max\{|a|,|b|\}} \\ &= \frac{1}{|a|} or \frac{1}{|b|} > \frac{1}{p^d} \geqslant \frac{1}{p^{d+1}} \\ &\quad (since, |a|,|b| \leqslant p^d) \end{aligned}$$

Thus in all the three cases, we get contradiction that $[a:1:b]$ is not in the $N([g:1:h],\frac{1}{p^{d+1}})$

So $|a| \leqslant 1$, $|b| \leqslant 1$, $|g|,|h| \leqslant 1 \leqslant p^d$ then
$d([g:1:h],[a:1:b]) \leqslant \frac{1}{p^{d+1}}$ which says that $[a:1:b] \in N([g:1:h],\frac{1}{p^{d+1}})$

■

Lemma 4.2.4. *Let* $[1:g:h],[g:1:h] \in \mathbb{P}^2$ *and let* $d \in \mathbb{N}_0$ *then*

(i) If $1 < |g|, |h| \leqslant p^d$ *then*

$N([1:g:h], \frac{1}{p^{d+1}}) = \{[1:a:b] \ni d([1:g:h],[1:a:b]) \leqslant \frac{1}{p^{d+1}}\}$

for all a, b *with the property* $|a| \leqslant p^d, |b| \leqslant p^d$

Further,if $|g| = |a|$ *and* $|h| = |b|$, *then*

$d([1:g:h],[1:a:b]) \leqslant \frac{1}{p^{d+1}}$

$\Rightarrow |gb - ah| \leqslant |g|^2 p^{-(d+1)}$ *or* $|gb - ah| \leqslant |h|^2 p^{-(d+1)}$

(ii) If $1 < |g|, |h| \leqslant p^d$ *then*

$N([g:1:h], \frac{1}{p^{d+1}}) = \{[a:1:b] \ni d([g:1:h],[a:1:b]) \leqslant \frac{1}{p^{d+1}}\}$

for all a, b *with the property* $|a| \leqslant p^d, |b| \leqslant p^d$

Further,if $|g| = |a|$ *and* $|h| = |b|$, *then*

$d([g:1:h],[a:1:b]) \leqslant \frac{1}{p^{d+1}}$

$\Rightarrow |ah - gb| \leqslant |g|^2 p^{-(d+1)}$ *or* $|ah - gb| \leqslant |h|^2 p^{-(d+1)}$

Proof. (i) We know that $d([1:g:h],[1:a:b]) = \frac{max\{|gb-ah|, |h-b|, |a-g|\}}{max\{1,|g|,|h|\} max\{1,|a|,|b|\}}$

since, $1 < |g|, |h| \leqslant p^d$ then $max\{1,|g|,|h|\} = max\{|g|,|h|\} \leqslant p^d$

Now, assume that $|a| > 1$, $|b| > 1$

Case(i): Suppose $|a| > 1, |b| \leqslant 1$ then $max\{1,|a|,|b|\} = |a|$ and

$$\begin{aligned} max\{|gb-ah|, |h-b|, |a-g|\} &> max\{|b-ah|, |h|, |1-g|\} \\ &= max\{|ah|, |h|, |g|\} = max\{|ah|, |g|\} \end{aligned}$$

(since, $|h-b| = |h|, |a-g| > |1-g| = |g|$ and

$|gb-ah| > |b-ah| = |ah| > |h| > 1$) Hence,

$$\begin{aligned} d([1:g:h],[1:a:b]) &= \frac{max\{|gb-ah|,|h-b|,|a-g|\}}{max\{1,|g|,|h|\}max\{1,|a|,|b|\}} \\ &> \frac{max\{|ah|,|g|\}}{|a|max\{|g|,|h|\}} \end{aligned}$$

Subcase(i): If $|g| > |ah| > |h|$ then

$$\begin{aligned} d([1:g:h],[1:a:b]) &> \frac{max\{|ah|,|g|\}}{|a|max\{|g|,|h|\}} = \frac{|g|}{|ag|} = \frac{1}{|a|} \\ &> \frac{1}{p^d} \geqslant \frac{1}{p^{d+1}} \\ &\quad (\because |a| \leqslant p^d) \end{aligned}$$

Subcase(ii): If $|g| < |h| < |ah|$ then

$$\begin{aligned} d([1:g:h],[1:a:b]) &> \frac{max\{|ah|,|g|\}}{|a|max\{|g|,|h|\}} \\ &= \frac{|ah|}{max\{|ah|,|ag|\}} = 1 or \frac{|h|}{|g|} > 1 or \frac{1}{|g|} \\ &> \frac{1}{p^d} \geqslant \frac{1}{p^{d+1}} (\because |g| \leqslant p^d, |h| > 1) \end{aligned}$$

Case(ii): Suppose $|a| \leqslant 1, |b| > 1$ then

$max\{1,|a|,|b|\} = |b|$ and

$max\{|gb-ah|,|h-b|,|a-g|\} > max\{|gb|,|b|,|g|\} = |gb|$.

(since, $|h-b| > |1-b| = |b| > 1, |a-g| = |g|$ and

$|gb-ah| > |gb-h| = |gb| > 1$).

Hence,

$$\begin{aligned} d([1:g:h],[1:a:b]) &= \frac{max\{|gb-ah|,|h-b|,|a-g|\}}{max\{1,|g|,|h|\}max\{1,|a|,|b|\}} \\ &> \frac{|gb|}{|b|max\{|g|,|h|\}} \\ &> \frac{1}{|h|} or 1 \\ &> \frac{1}{p^d} \geqslant \frac{1}{p^{d+1}} \\ &\quad (\because |h| \leqslant p^d) \end{aligned}$$

Case(iii): Suppose $|a| \leqslant 1, |b| \leqslant 1$ then $max\{1,|a|,|b|\} \leqslant 1$ and

$max\{|gb-ah|,|h-b|,|a-g|\} > max\{|h|,|g|\} > 1$.

(since, $|h-b| = |h|, |a-g| = |g|$ and $|gb-ah| > |b-ah| > |b-h| = |h|$).

Hence,

$$\begin{aligned} d([1:g:h],[1:a:b]) &= \frac{max\{|gb-ah|,|h-b|,|a-g|\}}{max\{1,|g|,|h|\}max\{1,|a|,|b|\}} \\ &> \frac{max\{|g|,|h|\}}{max\{|g|,|h|\}.1} \\ &> 1 > \frac{1}{p^d} \geqslant \frac{1}{p^{d+1}} \end{aligned}$$

Thus in all the three cases, we get contradiction that $[1 : a : b]$ is not in the $N([1 : g : h], \frac{1}{p^{d+1}})$

Hence,we consider $|a| > 1$, $|b| > 1$, $1 < |g|, |h| \leqslant p^d$ then
$d([1 : g : h], [1 : a : b]) \leqslant \frac{1}{p^{d+1}}$ which says that $[1 : a : b] \in N([1 : g : h], \frac{1}{p^{d+1}})$

Now see that, if $|g| \neq |a|$ and $|h| \neq |b|$ or if $|g| \neq |a|$ and $|h| = |b|$ or if $|g| = |a|$ and $|h| \neq |b|$, then

$$\begin{aligned} d([1 : g : h], [1 : a : b]) &= \frac{max\{|gb - ah|, |h - b|, |a - g|\}}{max\{1, |g|, |h|\}max\{1, |a|, |b|\}} \\ &= \frac{|gb - ah|}{max\{|g|, |h|\}max\{|a|, |b|\}} \\ &> \frac{|g - ah|}{max\{|g|, |h|\}max\{|a|, |b|\}} \\ &= \frac{max\{|g|, |ah|\}}{max\{|g|, |h|\}max\{|a|, |b|\}} \end{aligned}$$

Case(i): If $|g| > |ah| > |h|$ then

$$\begin{aligned} d([1 : g : h], [1 : a : b]) &> \frac{max\{|g|, |ah|\}}{max\{|g|, |h|\}max\{|a|, |b|\}} \\ &= \frac{|g|}{|g|max\{|a|, |b|\}} > \frac{1}{p^d} \geqslant \frac{1}{p^{d+1}} \end{aligned}$$

Case(ii): If $|g| < |h| < |ah|$ then

$$d([1:g:h],[1:a:b]) > \frac{max\{|g|,|ah|\}}{max\{|g|,|h|\}max\{|a|,|b|\}}$$

$$= \frac{|ah|}{|h|max\{|a|,|b|\}}$$

$$> \frac{1}{max\{|a|,|b|\}}$$

$$> \frac{1}{p^d} \geqslant \frac{1}{p^{d+1}}$$

Hence in both cases,for $|g| \neq |a|$ and $|h| \neq |b|$,for $|g| \neq |a|$ and $|h| = |b|$ and for $|g| = |a|$ and $|h| \neq |b|$,we get $d([1:g:h],[1:a:b]) \geqslant \frac{1}{p^{d+1}}$,
which is a contradiction.

Thus, from all the three above cases we can conclude that
$[1:a:b] \in N([1:g:h],\frac{1}{p^{d+1}})$ implies that $|g| = |a|$ and $|h| = |b|$.
Thus,we assume that $|g| = |a|$ and $|h| = |b|$ then we get

$$d([1:g:h],[1:a:b]) \leqslant \frac{1}{p^{d+1}} \Rightarrow \frac{|gb-ah|}{max\{|g|^2,|h|^2\}} \leqslant \frac{1}{p^{d+1}}$$

$$\Rightarrow |gb-ah| \leqslant max\{|g|^2,|h|^2\}p^{-(d+1)}$$

$$\Rightarrow |gb-ah| \leqslant |g|^2p^{-(d+1)} or$$

$$\Rightarrow |gb-ah| \leqslant |h|^2p^{-(d+1)}$$

(ii)We know that $d([g:1:h],[a:1:b]) = \dfrac{max\{|b-a|,|ah-gb|,|g-a|\}}{max\{1,|g|,|h|\}max\{1,|a|,|b|\}}$

since, $1 < |g|, |h| \leqslant p^d$ then $max\{1,|g|,|h|\} = max\{|g|,|h|\} \leqslant p^d$

Now, assume that $|a| > 1$, $|b| > 1$

Case(i): Suppose, $|a| > 1, |b| \leqslant 1$ then $max\{1,|a|,|b|\} = |a| > 1$ and

$$\begin{aligned} max\{|b-h|,|ah-gb|,|g-a|\} &> max\{|h|,|ah-g|,|1-a|\} \\ &> max\{|h|,|ah|\} = |ah| > 1 \end{aligned}$$

since, $|b-h| = |h|, |g-a| > |1-a| = |a| > 1$ and

$|ah-gb| > |ah-g| = |ah| > |a|$.

Hence,

$$\begin{aligned} d([g:1:h],[a:1:b]) &= \frac{max\{|b-h|,|ah-gb|,|g-a|\}}{max\{1,|g|,|h|\}max\{1,|a|,|b|\}} \\ &> \frac{|ah|}{|a|max\{|g|,|h|\}} > \frac{|h|}{max\{|g|,|h|\}} \\ &> \frac{1}{max\{|g|,|h|\}} \\ &> \frac{1}{p^d} \geqslant \frac{1}{p^{d+1}} \end{aligned}$$

Case(ii): Suppose, $|a| \leqslant 1, |b| > 1$ then $max\{1,|a|,|b|\} = |b|$ and

$max\{|b-h|,|ah-gb|,|g-a|\} > max\{|h|,|a-gb|,|g|\} = max\{|h|,|gb|\} > 1.$

since, $|b-h| > |1-h| = |h| > 1$, $|g-a| = |g| > 1$ and

$|ah-gb| > |a-gb| = |gb| > 1$

Hence,

$$\begin{aligned} d([g:1:h],[a:1:b]) &= \frac{max\{|b-h|,|ah-gb|,|g-a|\}}{max\{1,|g|,|h|\}max\{1,|a|,|b|\}} \\ &> \frac{max\{|h|,|gb|\}}{|b|max\{|g|,|h|\}} \end{aligned}$$

Subcase(i): If $|h| > |gb| > |g|$ then,

$$\begin{aligned} d([g:1:h],[a:1:b]) &> \frac{max\{|h|,|gb|\}}{|b|max\{|g|,|h|\}} > \frac{|h|}{|bh|} = \frac{1}{|b|} \\ &> \frac{1}{p^d} \geqslant \frac{1}{p^{d+1}} \end{aligned}$$

Subcase(ii): If $|gb| > |g| > |h|$ then,

$$\begin{aligned} d([g:1:h],[a:1:b]) &> \frac{max\{|h|,|gb|\}}{|b|max\{|g|,|h|\}} \\ &= \frac{|gb|}{|b|max\{|g|,|h|\}} > \frac{|g|}{max\{|g|,|h|\}} \\ &= \frac{1}{max\{|g|,|h|\}} > \frac{1}{p^d} \geqslant \frac{1}{p^{d+1}} \end{aligned}$$

Case(iii): Suppose, $|a| \leqslant 1, |b| \leqslant 1$ then $max\{1,|a|,|b|\} \leqslant 1$ and

$max\{|b-h|,|ah-gb|,|g-a|\} > max\{|h|,|g|\} > 1$.

since, $|b-h| = |h|, |g-a| = |g|$, and $|ah-gb| > |a-g| = |g|$.

Hence,

$$\begin{aligned} d([g:1:h],[a:1:b]) &= \frac{max\{|b-h|,|ah-gb|,|g-a|\}}{max\{1,|g|,|h|\}max\{1,|a|,|b|\}} \\ &> \frac{max\{|g|,|h|\}}{max\{|g|,|h|\}.1} = 1 \\ &> \frac{1}{p^d} \geqslant \frac{1}{p^{d+1}} \end{aligned}$$

Thus in all the three cases, we get contradiction that $[a:1:b]$ is not in the $N([g:1:h],\frac{1}{p^{d+1}})$
Hence,$|a|>1$, $|b|>1$, $1<|g|,|h|\leqslant p^d$, then
$d([g:1:h],[a:1:b])\leqslant\frac{1}{p^{d+1}}$ which says that $[a:1:b]\in N([g:1:h],\frac{1}{p^{d+1}})$

Now see that, if $|g|\neq|a|$ and $|h|\neq|b|$ or if $|g|\neq|a|$ and $|h|=|b|$ or if $|g|=|a|$ and $|h|\neq|b|$, then

$$\begin{aligned} d([g:1:h],[a:1:b]) &= \frac{max\{|ah-gb|,|b-h|,|g-a|\}}{max\{1,|g|,|h|\}max\{1,|a|,|b|\}} \\ &= \frac{|ah-gb|}{max\{|g|,|h|\}max\{|a|,|b|\}} \\ &> \frac{|h-gb|}{max\{|g|,|h|\}max\{|a|,|b|\}} \\ &= \frac{max\{|h|,|gb|\}}{max\{|g|,|h|\}max\{|a|,|b|\}} > \frac{1}{p^d} \geqslant \frac{1}{p^{d+1}} \end{aligned}$$

Case(i): If $|h| > |gb| > |g|$ then

$$\begin{aligned} d([1:g:h],[1:a:b]) &> \frac{max\{|h|,|gb|\}}{max\{|g|,|h|\}max\{|a|,|b|\}} \\ &= \frac{|h|}{|h|max\{|a|,|b|\}} \\ &> \frac{1}{p^d} \geqslant \frac{1}{p^{d+1}} \end{aligned}$$

Case(ii): If $|h| < |g| < |gb|$ then

$$\begin{aligned} d([1:g:h],[1:a:b]) &> \frac{max\{|h|,|gb|\}}{max\{|g|,|h|\}max\{|a|,|b|\}} \\ &= \frac{|gb|}{|g|max\{|a|,|b|\}} \\ &> \frac{1}{max\{|a|,|b|\}} \\ &> \frac{1}{p^d} \geqslant \frac{1}{p^{d+1}} \end{aligned}$$

Hence in all cases,for $|g| \neq |a|$ and $|h| \neq |b|$,for $|g| \neq |a|$ and $|h| = |b|$ and for $|g| = |a|$ and $|h| \neq |b|$,we get
$d([g:1:h],[a:1:b]) \geqslant \frac{1}{p^{d+1}}$, which is a contradiction.

Thus, from all the three above cases, we can conclude that,
$[a:1:b] \in N([g:1:h], \frac{1}{p^{d+1}})$ implies that $|g| = |a|$ and $|h| = |b|$.

Thus,we assume that $|g| = |a|$ and $|h| = |b|$ then we get

$$d([g:1:h],[a:1:b]) \leqslant \frac{1}{p^{d+1}} \quad \Rightarrow \quad \frac{|ah-gb|}{max\{|g|^2,|h|^2\}} \leqslant \frac{1}{p^{d+1}}$$

$$\Rightarrow \quad |ah-gb| \leqslant max\{|g|^2,|h|^2\}p^{-(d+1)}$$

$$\Rightarrow \quad |ah-gb| \leqslant |g|^2p^{-(d+1)} or |h|^2p^{-(d+1)}$$

■

Remark. (i) If $d([1:g:h],[u_1:u_2:u_3]) \leqslant \frac{1}{p^{d+1}}$ then by ultra-metric triangle inequality and lemma 4.2.5,we have

$$d([e_3],[u]) \leqslant max\{d([e_3],[1:g:h]),d([1:g:h],[u])\} \quad \leqslant \quad \frac{1}{p^{d+1}}$$

and $[u] = [u_1:u_2:u_3]$

(ii)If $d([e_3],[u]) \leqslant \frac{1}{p^{d+1}}$ then by ultra-metric triangle inequality and lemma 4.2.5,we have

$$d([1:g:h],[u]) \leqslant max\{d([1:g:h],[e_3]),d([e_3],[u])\} \quad \leqslant \quad \frac{1}{p^{d+1}}$$

(iii) If $d([g:1:h],[u_1:u_2:u_3]) \leqslant \frac{1}{p^{d+1}}$ then by ultra-metric triangle inequality and lemma 4.2.5,we have

$$d([e_3],[u]) \leqslant max\{d([e_3],[g:1:h]),d([g:1:h],[u])\} \quad \leqslant \quad \frac{1}{p^{d+1}}$$

and $[u] = [u_1 : u_2 : u_3]$

(iv)If $d([e_3], [u]) \leqslant \frac{1}{p^{d+1}}$ then by ultra-metric triangle inequality and lemma 4.2.5,we have

$$d([g : 1 : h], [u]) \leqslant max\{d([g : 1 : h], [e_3]), d([e_3], [u])\} \quad \leqslant \quad \frac{1}{p^{d+1}}$$

Hence from these observations,we know that lemma 4.2.5 follows the results.

Lemma 4.2.5. *Let* $[1 : g : h], [g : 1 : h] \in \mathbb{P}^2$ *and let* $d \in \mathbb{N}_0$.
If $|g|, |h| > p^d$ *and* $|g| < |h|$ *then*
(i) $N([1 : g : h], \frac{1}{p^{d+1}}) = \{[1 : a : b] \ni d([1 : g : h], [1 : a : b]) \leqslant \frac{1}{p^{d+1}}\} \cup \{[e_3]\}$

(ii) $N([g : 1 : h], \frac{1}{p^{d+1}}) = \{[1 : a : b] \ni d([1 : g : h], [1 : a : b]) \leqslant \frac{1}{p^{d+1}}\} \cup \{[e_3]\}$

(iii) $N([e_3], \frac{1}{p^{d+1}}) = \{[1 : a : b] \ni d([1 : g : h], [1 : a : b]) \leqslant \frac{1}{p^{d+1}}\} \cup \{[e_3]\}$

Proof. (i) We know that $d([1 : g : h], [1 : a : b]) = \frac{max\{|gb - ah|, |h - b|, |a - g|\}}{max\{1, |g|, |h|\}max\{1, |a|, |b|\}}$

since $|g|, |h| > p^d$ and $|g| < |h|$ then $max\{1, |g|, |h|\} = |h|$
Now, assume that $|a| > p^d$, $|b| > p^d$

Case(i): Suppose $|a| > p^d, |b| \leqslant p^d$ then $max\{1, |a|, |b|\} = |a|$ and

$$\begin{aligned} max\{|gb - ah|, |h - b|, |a - g|\} &> max\{|b - ah|, |h|, |a - g|\} \\ &> max\{|ah|, |h|, |ah|\} = |ah| \end{aligned}$$

(since, $|gb-ah| > |b-ah| = |ah|$,$|h-b| = |h| < |ah|$ and

$|a-g| > |a-h| > |a+ah| = |ah|$).

Hence,

$$\begin{aligned} d([1:g:h],[1:a:b]) &= \frac{max\{|gb-ah|,|h-b|,|a-g|\}}{max\{1,|g|,|h|\}max\{1,|a|,|b|\}} \\ &> \frac{|ah|}{|ah|} = 1 > \frac{1}{p^d} \geqslant \frac{1}{p^{d+1}} \end{aligned}$$

Case(ii): Suppose $|a| \leqslant p^d, |b| > p^d$ then $max\{1,|a|,|b|\} = |b|$ and

$$\begin{aligned} max\{|gb-ah|,|h-b|,|a-g|\} &> max\{|a-bh|,|a-bh|,|g|\} \\ &> max\{|bh|,|bh|,|g|\} = |bh| \end{aligned}$$

(since, $|gb-ah| > |a-ah| > |a-bh| = |bh|, |a-g| = |g| > p^d$ and

$|h-b| > |a-b| > |a-bh| = |bh| > p^d$)

Hence,

$$\begin{aligned} d([1:g:h],[1:a:b]) &= \frac{max\{|gb-ah|,|h-b|,|a-g|\}}{max\{1,|g|,|h|\}max\{1,|a|,|b|\}} \\ &> \frac{|bh|}{|bh|} = 1 > \frac{1}{p^d} \geqslant \frac{1}{p^{d+1}} \end{aligned}$$

Case(iii): Suppose, $|a| \leqslant p^d, |b| \leqslant p^d$ then $max\{1,|a|,|b|\} \leqslant p^d$ and

$max\{|gb-ah|,|h-b|,|a-g|\} > max\{|h|p^d,|h|,|g|\} > p^d$.

(since,$|gb-ah| > |b-ah| > |b-hp^d| > |h|p^d$,$|h-b| = |h|$ and $|a-g| = |g|$).

Hence,

$$d([1:g:h],[1:a:b]) = \frac{max\{|gb-ah|,|h-b|,|a-g|\}}{max\{1,|g|,|h|\}max\{1,|a|,|b|\}}$$

$$> \frac{|h|p^d}{|h|p^d} > 1 > \frac{1}{p^d} \geqslant \frac{1}{p^{d+1}}$$

Thus in all the three cases, we get contradiction that $[1:a:b]$ is not in the $N([1:g:h],\frac{1}{p^{d+1}})$.
Hence, $|a|>p^d$, $|b|>p^d$, $|g|,|h|>p^d$ and $|g|<|h|$,then
$d([1:g:h],[1:a:b]) \leqslant \frac{1}{p^{d+1}}$ which says that $[1:a:b] \in N([1:g:h],\frac{1}{p^{d+1}})$

(ii) Similarly, $|g|,|h|>p^d$,
then $d([g:1:h],[1:a:b]) = \frac{max\{|b-ah|,|h-gb|,|ga-1|\}}{max\{1,|g|,|h|\}max\{1,|a|,|b|\}}$

Now, assume that $|a|>p^d$, $|b|>p^d$

Case(i): Suppose, $|a|>p^d, |b| \leqslant p^d$ then
$max\{1,|a|,|b|\} = |a|$ and

$$\begin{aligned} max\{|b-ah|,|h-gb|,|ag-1|\} &> max\{|ah|,|ah|,|ag|\} \\ &= max\{|ah|,|ag|\} \\ &= |a|max\{|g|,|h|\} = |ah| \end{aligned}$$

(since, $|h-gb|>|h-ag|>|h-ah|=|ah|$).

Hence,

$$d([1:g:h],[1:a:b]) = \frac{max\{|b-ah|,|h-gb|,|ga-1|\}}{max\{1,|g|,|h|\}max\{1,|a|,|b|\}}$$

$$= \frac{|ah|}{|ah|} = 1 > \frac{1}{p^d} \geqslant \frac{1}{p^{d+1}}$$

Case(ii): Suppose $|a| \leqslant p^d, |b| > p^d$ then $max\{1,|a|,|b|\} = |b|$ and

$max\{|b-ah|,|h-gb|,|ga-1|\} > max\{|bh|,|gb|,|a-1|\} > |bh| > p^d.$

(since,$|b-ah| > |a-ah| > |a-bh| = |bh|$,
$|ga-1| > |a-1| = |a| < |bh|$ or $1 < |bh|$ and
$|h-gb| > |a-gb| > |a-bh| = |bh|$)
Hence,

$$d([1:g:h],[1:a:b]) = \frac{max\{|b-ah|,|h-gb|,|ga-1|\}}{max\{1,|g|,|h|\}max\{1,|a|,|b|\}}$$

$$> \frac{|bh|}{|bh|} > 1 > \frac{1}{p^d} \geqslant \frac{1}{p^{d+1}}$$

Case(iii): Suppose, $|a| \leqslant p^d, |b| \leqslant p^d$ then $max\{1,|a|,|b|\} \leqslant p^d$ and
$max\{|b-ah|,|h-gb|,|ga-1|\} > max\{|h|p^d,|a|\} > |h|p^d.$

(since,$|b-ah| > |b-hp^d| > |h|p^d, |ga-1| = |ga| > |a|$ and
$|h-gb| > |h-gp^d| > |h|p^d$)

Hence,

$$d([1:g:h],[1:a:b]) = \frac{max\{|b-ah|,|h-gb|,|ga-1|\}}{max\{1,|g|,|h|\}max\{1,|a|,|b|\}}$$

$$> \frac{|h|p^d}{|h|p^d} > 1 > \frac{1}{p^d} \geqslant \frac{1}{p^{d+1}}$$

Thus in all the three cases, we get contradiction that $[1:a:b]$ is not in the $N([g:1:h],\frac{1}{p^{d+1}})$

Hence, $|a| > p^d$, $|b| > p^d$, $|g|, |h| > p^d$, and $|g| < |h|$ then

$d([g:1:h],[1:a:b]) \leqslant \frac{1}{p^{d+1}}$, which says that

$[1:a:b] \in N([g:1:h],\frac{1}{p^{d+1}})$

(iii) Next we need to prove that, if $|g|, |h| > p^d$ then an element

$[e_3] = [0:0:1] \in N([1:g:h],\frac{1}{p^{d+1}})$ for $|g| < |h|$

We know that

$$d([0:0:1],[1:g:h]) = \frac{max\{|g|,1\}}{max\{1,|g|,|h|\}} = \frac{|g|}{max\{|g|,|h|\}}$$

for $|g|, |h| > p^d$

By the hypothesis, $|g| < |h|$

(i.e) if $max\{1,|g|,|h|\} = |h|$ then $d([0:0:1],[1:g:h]) = \frac{1}{|h|} \leqslant \frac{1}{p^{d+1}}$, which says that $[0:0:1]$ is in the neighbourhood of $[1:g:h]$, for $|g|, |h| > p^d$.

Therefore, $d([e_3],[1:g:h]) \leqslant \frac{1}{p^{d+1}}$ for $|g|, |h| > p^d$ and $|g| < |h|$ ■

Remark. We note that $\frac{1}{p^{d+1}}$ neighbourhoods of $[e_3], [1:g:h], [g:1:h]$ are equal if $|g|, |h| > p^d$. We can write the elements of $\mathbb{P}^2$ as $[1:g:h] = [\frac{1}{h} : \frac{g}{h} : 1]$ and $[g:1:h] = [\frac{g}{h} : \frac{1}{h} : 1]$ and choose large $|h|$ and greater than $|g|$,the points $[\frac{1}{h} : \frac{g}{h} : 1], [\frac{g}{h} : \frac{1}{h} : 1]$ tends to $[e_3] = [0:0:1]$

Proposition 4.2.6. *For any point $[u] \in \mathbb{P}^2$ and for each $d \in \mathbb{N}_0$ then there exist $1 + 4p^{2d+2}$ disjoint $\frac{1}{p^{d+1}}$ neighbourhoods. More Precisely, $N([u], \frac{1}{p^{d+1}})$ is one of these neighbourhoods.*

Further, these neighbourhoods cover $\mathbb{P}^2$

Proof. Let us fixed $d \in \mathbb{N}_0$ and consider $[1:g:h] \in \mathbb{P}^2$ such that $|g| \leqslant 1 \leqslant p^d$, $|h| \leqslant 1 \leqslant p^d$ and $g = g_0 + g_1 x + ... + g_d x^d + ..., $

$h = h_0 + h_1 x + ... + h_d x^d + ...$

Then by proposition 4.2.2,

$$N([1:g:h], \frac{1}{p^{d+1}}) = \{[1:a:b] \ni d([1:g:h], [1:a:b]) \leqslant \frac{1}{p^{d+1}}\}$$

So we have,

$$\begin{aligned} d([1:g:h], [1:a:b]) &= max\{|gb - ah|, |h - b|, |a - g|\} \leqslant \frac{1}{p^{d+1}} \\ &\quad (since, |g|, |h|, |a|, |b| \leqslant 1) \end{aligned}$$

If $max\{|gb - ah|, |h - b|, |a - g|\} = |gb - ah|$ or $|h - b|$ or $|a - g|$

That is, $|a - g| \leqslant \frac{1}{p^{d+1}}$ says that, all the terms of degree less than x^{d+1} are zero or $|h - b| \leqslant \frac{1}{p^{d+1}}$ says that, all the terms of degree less than x^{d+1} are zero or $|gb - ah| \leqslant \frac{1}{p^{d+1}}$ says that all the terms of degree less than x^{d+1} are zero.

Since , $a = g_0 + g_1x + ... + g_dx^d + r$, $b = h_0 + h_1x + ... + h_dx^d + r$, for some $r \in x^{d+1}\mathbb{O}$

With this in mind, for each $a_0, a_1, a_2...., a_d \in \mathbb{F}_p$ and $b_0, b_1, ..., b_d \in \mathbb{F}_p$

Consider the set

$$\begin{aligned} S &= \{[1 : a : b] \ni a = a_0 + a_1x + a_2x^2 + ...a_dx^d + r, \\ &\quad b = b_0 + b_1x + b_2x^2 + ...b_dx^d + r\} \end{aligned} \tag{4.11}$$

for some $r \in x^{d+1}\mathbb{O}$.

Hence for any point $[1 : g : h] \in \mathbb{P}^2$ such that $|g| \leqslant 1 \leqslant p^d, |h| \leqslant 1 \leqslant p^d$ with $a_0 = g_0, a_1 = g_1, ...a_d = g_d$ and $b_0 = h_0, b_1 = h_1, ...b_d = h_d$

We observe that for any element $[1 : a : b] \in S$.

Thus,$N([1 : g : h], \frac{1}{p^{d+1}}) = S$

Note that,there are p^{d+1} distinct choices of $a_0, a_1, a_2...., a_d \in \mathbb{F}_p$ and $b_0, b_1, ..., b_d \in \mathbb{F}_p$

So, there are $(p^{d+1})^2 = p^{2d+2}$ different sets of the form (4.11).

Similarly,by proposition 4.2.2,

$$N([g : 1 : h], \frac{1}{p^{d+1}}) = \{[a : 1 : b] \ni d([g : 1 : h], [a : 1 : b]) \leqslant \frac{1}{p^{d+1}}\}$$

So we have,

$$\begin{aligned} d([g : 1 : h], [a : 1 : b]) &= max\{|ah - gb|, |b - h|, |g - a|\} \leqslant \frac{1}{p^{d+1}} \\ &\quad (since, |g|, |h|, |a|, |b| \leqslant 1) \end{aligned}$$

If $max\{|ah - gb|, |b - h|, |g - a|\} = |ah - gb|$ or $|b - h|$ or $|g - a|$

That is, $|g-a| \leqslant \frac{1}{p^{d+1}}$ says that, all the terms of degree less than x^{d+1} are zero or

$|b-h| \leqslant \frac{1}{p^{d+1}}$ says that, all the terms of degree less than x^{d+1} are zero or

$|ah-gb| \leqslant \frac{1}{p^{d+1}}$ says that all the terms of degree less than x^{d+1} are zero.

Since , $a = g_0 + g_1x + ... + g_dx^d + r$, $b = h_0 + h_1x + ... + h_dx^d + r$ for some $r \in x^{d+1}\mathbb{O}$

With this in mind, for each $a_0, a_1, a_2...., a_d \in \mathbb{F}_p$ and $b_0, b_1, ..., b_d \in \mathbb{F}_p$

Consider the set

$$\begin{aligned} S &= \{[a:1:b] \ni a = a_0 + a_1x + a_2x^2 + ...a_dx^d + r, \\ &\quad b = b_0 + b_1x + b_2x^2 + ...b_dx^d + r\} \end{aligned} \tag{4.12}$$

for some $r \in x^{d+1}\mathbb{O}$.

Hence for any point $[g:1:h] \in \mathbb{P}^2$ such that $|g| \leqslant 1 \leqslant p^d, |h| \leqslant 1 \leqslant p^d$ with

$a_0 = g_0, a_1 = g_1, ...a_d = g_d$ and $b_0 = h_0, b_1 = h_1, ...b_d = h_d$

We observe that for any element $[a:1:b] \in S$.

Thus $N([g:1:h], \frac{1}{p^{d+1}}) = S$

Note that,there are p^{d+1} distinct choices of $a_0, a_1, a_2...., a_d \in \mathbb{F}_p$ and

$b_0, b_1, ..., b_d \in \mathbb{F}_p$.

So, there are $(p^{d+1})^2 = p^{2d+2}$ different sets of the form (4.12).

Now,consider $[1:g:h] \in \mathbb{P}^2$ such that $1 < |g| \leqslant p^d, 1 < |h| \leqslant p^d$, and

$g = g_0 + g_1x + ... + g_dx^d + ..., h = h_0 + h_1x + ... + h_dx^d + ...$

Then the neighbourhoods becomes,

$$N([1:g:h], \frac{1}{p^{d+1}}) = \{[1:a:b] \ni d([1:g:h],[1:a:b]) \leqslant \frac{1}{p^{d+1}}\}$$

So, assume that $|g| = |a|, |h| = |b|$,

$$d([1:g:h],[1:a:b]) \leqslant \frac{1}{p^{d+1}} \quad \Rightarrow \quad \frac{max\{|gb-ah|,|h-b|,|a-g|\}}{max\{|g|^2,|h|^2\}} \leqslant \frac{1}{p^{d+1}}$$

$$\Rightarrow \quad \frac{|gb-ah|}{max\{|g|^2,|h|^2\}} \leqslant \frac{1}{p^{d+1}}$$

$$\Rightarrow \quad |gb-ah| \leqslant \frac{1}{p^{d+1}}$$
$$(since, |g|, |h| > 1).$$

If $|gb-ah| \leqslant \frac{1}{p^{d+1}}$ says that all the terms of degree less than x^{d+1} are zero.Since $a = g_0 + g_1x + ... + g_dx^d + r$, $b = h_0 + h_1x + ... + h_dx^d + r$ for some $r \in x^{d+1}\mathbb{O}$
With this in mind, we choose for each $a_0, a_1, a_2...., a_d \in \mathbb{F}_p$ and $b_0, b_1, ..., b_d \in \mathbb{F}_p$

Consider the set,

$$\begin{aligned} S &= \{[1:a:b] \ni a = a_0 + a_1x + a_2x^2 + ...a_dx^d + r, \\ &\quad b = b_0 + b_1x + b_2x^2 + ...b_dx^d + r\} \end{aligned} \tag{4.13}$$

for some $r \in x^{d+1}\mathbb{O}$. Hence for any point $[1:g:h] \in \mathbb{P}^2$ such that $1 < |g| \leqslant p^d$, $1 < |h| \leqslant p^d$ with $a_0 = g_0, a_1 = g_1, ...a_d = g_d$ and $b_0 = h_0, b_1 = h_1, ...b_d = h_d$
We observe that for any element $[1:a:b] \in S$.
Thus, $N([1:g:h], \frac{1}{p^{d+1}}) = S$

Therefore, there are p^{d+1} distinct choices of $a_0, a_1, a_2...., a_d \in \mathbb{F}_p, b_0, b_1, ..., b_d \in \mathbb{F}_p$.
So, there are $p^{2(d+1)}$ different sets of the form (4.13).

Similarly,by proposition 4.2.2,

Consider $[g:1:h] \in \mathbb{P}^2$ such that $1 < |g| \leqslant p^d, 1 < |h| \leqslant p^d$, and

$g = g_0 + g_1x + ... + g_dx^d + ..., h = h_0 + h_1x + ... + h_dx^d + ...$

Then the neighbourhoods becomes,

$$N([g:1:h], \frac{1}{p^{d+1}}) \quad = \quad \{[a:1:b] \ni d([g:1:h],[a:1:b]) \leqslant \frac{1}{p^{d+1}}\}$$

So, assume that $|g| = |a|, |h| = |b|$,

$$\begin{aligned} d([g:1:h],[a:1:b]) \leqslant \frac{1}{p^{d+1}} \quad &\Rightarrow \quad \frac{max\{|ah-gb|, |b-h|, |g-a|\}}{max\{|g|^2, |h|^2\}} \leqslant \frac{1}{p^{d+1}} \\ &\Rightarrow \quad \frac{|ah-gb|}{max\{|g|^2, |h|^2\}} \leqslant \frac{1}{p^{d+1}} \\ &\Rightarrow \quad |ah-gb| \leqslant \frac{1}{p^{d+1}} \\ &\quad (since, |g|, |h| > 1). \end{aligned}$$

If $|ah - gb| \leqslant \frac{1}{p^{d+1}}$ says that all the terms of degree less than x^{d+1} are zero.Since $a = g_0 + g_1x + ... + g_dx^d + r$, $b = h_0 + h_1x + ... + h_dx^d + r$, for some $r \in x^{d+1}\mathbb{O}$

With this in mind, we choose for each $a_0, a_1, a_2...., a_d \in \mathbb{F}_p$ and $b_0, b_1, ..., b_d \in \mathbb{F}_p$

Consider the set,

$$\begin{aligned} S \quad = \quad &\{[a:1:b] \ni a = a_0 + a_1x + a_2x^2 + ...a_dx^d + r, \\ &b = b_0 + b_1x + b_2x^2 + ...b_dx^d + r\} \end{aligned} \tag{4.14}$$

for some $r \in x^{d+1}\mathbb{O}$. Hence for any point $[g : 1 : h] \in \mathbb{P}^2$ such that $1 < |g| \leqslant p^d$, $1 < |h| \leqslant p^d$ with $a_0 = g_0, a_1 = g_1, ..., a_d = g_d$ and $b_0 = h_0, b_1 = h_1, ..., b_d = h_d$
We observe that for any element $[a : 1 : b] \in S$. Thus, $N([g : 1 : h], \frac{1}{p^{d+1}}) = S$

Therefore, there are p^{d+1} distinct choices of $a_0, a_1, a_2...., a_d \in \mathbb{F}_p, b_0, b_1, ..., b_d \in \mathbb{F}_p$.
So, there are $p^{2(d+1)}$ different sets of the form (4.14).

Finally, by the remark,if $[u] \in \mathbb{P}^2$ such that $[u] = [e_3]$ or $[u] = [1 : g : h]$ or $[u] = [g : 1 : h]$, for some $|h| > p^d$ and $|g| > p^d$, then $N([u], \frac{1}{p^{d+1}})$ is exactly the set

$$S = \{[1 : a : b] \ni |a| > p^d, |b| > p^d\} \cup \{[e_3]\} \tag{4.15}$$

Hence, for all $[u] \in \mathbb{P}^2$, there exist a set S of the form (4.11), (4.12), (4.13), (4.14) and (4.15) such that $[u] \in S$ and $N([u], \frac{1}{p^{d+1}}) = S$
Additionally, there are $2p^{2d+2}$ neighbourhoods of the form ((4.11), (4.12)), $2p^{2d+2}$ neighbourhoods of the form ((4.13), (4.14)) and one neighbourhood of the form (4.15). Thus, we have totally, $2p^{2d+2} + 2p^{2d+2} + 1 = 1 + 4p^{2d+2}$ neighbourhoods and these taken sets are all disjoint which is understood from the construction. ■

We discussed the ∞-norm with some intermediary results and using the ∞-norm or max-norm,we defined the metric in $\mathbb{P}^2$ and their related properties are proved and we show that the points of neighbourhoods in $\mathbb{P}^2$ with respect to the metric.The notation of the distance between two points in $\mathbb{P}^2$ (i.e), $d(x, y)$ is used in the next chapter.

Chapter 5

Free Generators Theorem in Projective General Linear Groups

We desire to obtain conditions for which elements $A, B \in PGL_3(F_p((x)))$ will freely generate a free subgroup of $PGL_3(F_p((x)))$. To get into the above result, we need to consider the action of A and B on $\mathbb{P}^2$ by seeing the action of the pre-images of A and B in $PGL_3(F_p((x)))$ on $\mathbb{P}^2$.

5.1. Free Generators theorem

Before stating the Free Generators Theorem, we need to find out the general form of $A, B \in PGL_3(F_p((x)))$ in terms of eigenvalues and eigenvectors of $\tilde{A}, \tilde{B} \in GL_3(F_p((x)))$.

Lemma 5.1.1. *Let $\tilde{A}, \tilde{B}$ are the elements in $GL_3(F_p((x)))$.Suppose that $\tilde{A}$ has distinct eigenvectors $[a : b : c], [1 : g : h], [d : 1 : e]$ with corresponding eigenvalues $x, y, z \in F_p((x))$ and that $\tilde{B}$ has distinct eigenvectors $[1 : \tilde{b} : \tilde{c}], [1 : \tilde{g} : \tilde{h}], [\tilde{d} : 1 : \tilde{e}]$ with corresponding eigenvalues $x^{'}, y^{'}, z^{'} \in F_p((x))$*

then the respective images in $PGL_3(F_p((x)))$ *are*

$$A=\left[\begin{pmatrix} a(eg-h)+b(dhf'-ef)+c(f-dgf') & ae(f-1)+adh(1-f')+cd(f'-f) & adg(f'-1)+a(1-f)+bd(f-f') \\ beg(1-f)+bh(f'-1)+cg(f-f') & a(egf-hf')+b(dh-e)+c(f'-dgf) & ag(f'-f)+bdg(f-1)+b(1-f') \\ beh(f'-f)+ceg(1-f')+ch(f-1) & aeh(f-f')+cdh(1-f)+ce(f'-1) & a(egf'-hf)+b(dhf-ef')+c(1-dg) \end{pmatrix}\right]$$

and

$$B=\left[\begin{pmatrix} (\tilde{e}\tilde{g}-\tilde{h})+\tilde{b}(\tilde{d}\tilde{h}\tilde{f}'-\tilde{e}\tilde{f})+\tilde{c}(\tilde{f}-\tilde{d}\tilde{g}\tilde{f}') & \tilde{e}(\tilde{f}-1)+\tilde{d}\tilde{h}(1-\tilde{f}')+\tilde{c}\tilde{d}(\tilde{f}'-\tilde{f}) & \tilde{d}\tilde{g}(\tilde{f}'-1)+(1-\tilde{f})+\tilde{b}\tilde{d}(\tilde{f}-\tilde{f}') \\ \tilde{b}\tilde{e}\tilde{g}(1-\tilde{f})+\tilde{b}\tilde{h}(\tilde{f}'-1)+\tilde{c}\tilde{g}(\tilde{f}-\tilde{f}') & (\tilde{e}\tilde{g}\tilde{f}-\tilde{h}\tilde{f}')+\tilde{b}(\tilde{d}\tilde{h}-\tilde{e})+\tilde{c}(\tilde{f}'-\tilde{d}\tilde{g}\tilde{f}) & \tilde{g}(\tilde{f}'-\tilde{f})+\tilde{b}\tilde{d}\tilde{g}(\tilde{f}-1)+\tilde{b}(1-\tilde{f}') \\ \tilde{b}\tilde{e}\tilde{h}(\tilde{f}'-\tilde{f})+\tilde{c}\tilde{e}\tilde{g}(1-\tilde{f}')+\tilde{c}\tilde{h}(\tilde{f}-1) & \tilde{e}\tilde{h}(\tilde{f}-\tilde{f}')+\tilde{c}\tilde{d}\tilde{h}(1-\tilde{f})+\tilde{c}\tilde{e}(\tilde{f}'-1) & (\tilde{e}\tilde{g}\tilde{f}'-\tilde{h}\tilde{f})+\tilde{b}(\tilde{d}\tilde{h}\tilde{f}-\tilde{e}\tilde{f}')+\tilde{c}(1-\tilde{d}\tilde{g}) \end{pmatrix}\right]$$

where $f=\dfrac{y}{x}, f'=\dfrac{z}{x}, \tilde{f}=\dfrac{y'}{x'}$ *and* $\tilde{f}'=\dfrac{z'}{x'}$

Proof. We know that using eigenvalues and eigenvectors of $\tilde{A}$, we can write it as

$$\begin{pmatrix} a & 1 & d \\ b & g & 1 \\ c & h & e \end{pmatrix} \begin{pmatrix} x & 0 & 0 \\ 0 & y & 0 \\ 0 & 0 & z \end{pmatrix} \begin{pmatrix} (eg-h) & (dh-e) & (1-dg) \\ (c-be) & (ae-cd) & (bd-a) \\ (bh-cg) & (c-ah) & (ag-b) \end{pmatrix} = Q \wedge Q^{-1}$$

where $Q = \begin{pmatrix} a & 1 & d \\ b & g & 1 \\ c & h & e \end{pmatrix}$ be the 3×3 matrix whose j^{th}-column is the eigenvectors

of the matrix $\tilde{A}$,

$\wedge = \begin{pmatrix} x & 0 & 0 \\ 0 & y & 0 \\ 0 & 0 & z \end{pmatrix}$ be the diagonal matrix whose diagonal elements is the corresponding

eigenvalues $\wedge_{ii} = \lambda_i$ and

$Q^{-1} = \begin{pmatrix} (eg-h) & (dh-e) & (1-dg) \\ (c-be) & (ae-cd) & (bd-a) \\ (bh-cg) & (c-ah) & (ag-b) \end{pmatrix}$ be the inverse of the matrix Q whose

elements are necessarily distinct eigenvectors and non-zero.

So,the matrix Q^{-1} is scaled by $det(Q) = a(eg-h) + b(dh-e) + c(1-dg)$, which is also non-zero.So,the scaled value is not needed here to get into the form of $\tilde{A}$.

Since,the eigenvalues of $\tilde{A} \in GL_3(F_p((x)))$ are non-zero.

In particular $x \neq 0, \frac{1}{x}$ exist.

Thus,the matrix $\wedge$ will become as,$\wedge = \begin{pmatrix} x & 0 & 0 \\ 0 & y & 0 \\ 0 & 0 & z \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & f^{'} \end{pmatrix}$

where $f = \frac{y}{x}, f^{'} = \frac{z}{x}$.

Hence, we solve these matrices which gives the matrix A in $PGL_3(F_p((x)))$ as,

$$\begin{pmatrix} a & 1 & d \\ b & g & 1 \\ c & h & e \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & f^{'} \end{pmatrix} \begin{pmatrix} (eg-h) & (dh-e) & (1-dg) \\ (c-be) & (ae-cd) & (bd-a) \\ (bh-cg) & (c-ah) & (ag-b) \end{pmatrix} = A$$

Similarly,we know that using eigenvalues and eigenvectors of $\tilde{B}$, we can write it as,

$$\begin{pmatrix} 1 & 1 & \tilde{d} \\ \tilde{b} & \tilde{g} & 1 \\ \tilde{c} & \tilde{h} & \tilde{e} \end{pmatrix} \begin{pmatrix} x^{'} & 0 & 0 \\ 0 & y^{'} & 0 \\ 0 & 0 & z^{'} \end{pmatrix} \begin{pmatrix} (\tilde{e}\tilde{g}-\tilde{h}) & (\tilde{d}\tilde{h}-\tilde{e}) & (1-\tilde{d}\tilde{g}) \\ (\tilde{c}-\tilde{b}\tilde{e}) & (\tilde{e}-\tilde{c}\tilde{d}) & (\tilde{b}\tilde{d}-1) \\ (\tilde{b}\tilde{h}-\tilde{c}\tilde{g}) & (\tilde{c}-\tilde{h}) & (\tilde{g}-\tilde{b}) \end{pmatrix} = Q \wedge Q^{-1}$$

where $Q = \begin{pmatrix} 1 & 1 & \tilde{d} \\ \tilde{b} & \tilde{g} & 1 \\ \tilde{c} & \tilde{h} & \tilde{e} \end{pmatrix}$ be the 3×3 matrix whose j^{th}-column is the eigenvectors

of the matrix $\tilde{B}$,

$\wedge = \begin{pmatrix} x^{'} & 0 & 0 \\ 0 & y^{'} & 0 \\ 0 & 0 & z^{'} \end{pmatrix}$ be the diagonal matrix whose diagonal elements is the

corresponding eigenvalues $\wedge_{ii} = \lambda_i$ and

$Q^{-1} = \begin{pmatrix} (\tilde{e}\tilde{g} - \tilde{h}) & (\tilde{d}\tilde{h} - \tilde{e}) & (1 - \tilde{d}\tilde{g}) \\ (\tilde{c} - \tilde{b}\tilde{e}) & (\tilde{e} - \tilde{c}\tilde{d}) & (\tilde{b}\tilde{d} - 1) \\ (\tilde{b}\tilde{h} - \tilde{c}\tilde{g}) & (\tilde{c} - \tilde{h}) & (\tilde{g} - \tilde{b}) \end{pmatrix}$ be the inverse of the matrix Q whose

elements are necessarily distinct eigenvectors and non-zero.

So,the matrix Q^{-1} is scaled by $det(Q) = (\tilde{e}\tilde{g} - \tilde{h}) + \tilde{b}(\tilde{d}\tilde{h} - \tilde{e}) + \tilde{c}(1 - \tilde{d}\tilde{g})$, which is also non-zero.So,the scaled value is not needed here to get into the form of $\tilde{B}$.

Since,the eigenvalues of $\tilde{B} \in GL_3(F_p((x)))$ are non-zero.

In particular $x^{'} \neq 0, \frac{1}{x^{'}}$ exist.

Thus,the matrix $\wedge$ will become as, $\wedge = \begin{pmatrix} x^{'} & 0 & 0 \\ 0 & y^{'} & 0 \\ 0 & 0 & z^{'} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \tilde{f} & 0 \\ 0 & 0 & \tilde{f}' \end{pmatrix}$,

where $\tilde{f} = \frac{y^{'}}{x^{'}}, \tilde{f}' = \frac{z^{'}}{x^{'}}$.

Hence, we solve these matrices which gives the matrix B in $PGL_3(F_p((x)))$ as,

$$\begin{pmatrix} 1 & 1 & \tilde{d} \\ \tilde{b} & \tilde{g} & 1 \\ \tilde{c} & \tilde{h} & \tilde{e} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \tilde{f} & 0 \\ 0 & 0 & \tilde{f}' \end{pmatrix} \begin{pmatrix} (\tilde{e}\tilde{g} - \tilde{h}) & (\tilde{d}\tilde{h} - \tilde{e}) & (1 - \tilde{d}\tilde{g}) \\ (\tilde{c} - \tilde{b}\tilde{e}) & (\tilde{e} - \tilde{c}\tilde{d}) & (\tilde{b}\tilde{d} - 1) \\ (\tilde{b}\tilde{h} - \tilde{c}\tilde{g}) & (\tilde{c} - \tilde{h}) & (\tilde{g} - \tilde{b}) \end{pmatrix} = B$$ ■

Remark. If we assume that the six distinct elements of $\mathbb{P}^2$ are of the form $[a : b : c], [1 : g : h], [d : 1 : e], [1 : \tilde{b} : \tilde{c}], [1 : \tilde{g} : \tilde{h}], [\tilde{d} : 1 : \tilde{e}]$ for some $a, b, c, g, h, d, e, \tilde{b}, \tilde{c}, \tilde{d}, \tilde{e}, \tilde{g}, \tilde{h} \in F_p((x))$, then the lemma 5.1.1 gives the general form of A and B in $PGL_3(F_p((x)))$

5.2. Statement of the Free Generators Theorem

The proof of the theorem will use the Proposition 1.1 (Ping-Pong Lemma) in [73]

Theorem 5.2.1. *If there exist* $a, b, c, g, h, d, e, \tilde{b}, \tilde{c}, \tilde{d}, \tilde{e}, \tilde{g}, \tilde{h} \in F_p((x))$ *and* $f, f', \tilde{f}, \tilde{f'} \in F_p((x))^*$ *such that,*
(i) $d([u],[v]) > \frac{1}{p^{d+1}}$*,for each pair of* $[u],[v]$ *in* $\{[a:b:c],[1:g:h],[d:1:e],[1:\tilde{b}:\tilde{c}],[1:\tilde{g}:\tilde{h}],[\tilde{d}:1:\tilde{e}]\}$

(ii) $min\{|f|,|f^{-1}|\} \leqslant \frac{1}{p^{2d+1}}, min\{|f'|,|(f')^{-1}|\} \leqslant \frac{1}{p^{2d+1}}$ *and* $min\{|\tilde{f}|,|\tilde{f}^{-1}|\} \leqslant \frac{1}{p^{2d+1}}, min\{|\tilde{f'}|,|(\tilde{f'})^{-1}|\} \leqslant \frac{1}{p^{2d+1}}$

(iii) there exist $[k] \in \mathbb{P}^2$ *such that* $d([k],[u]) > \frac{1}{p^{d+1}}$ *for each* $[u]$ *in* $\{[a:b:c],[1:g:h],[d:1:e],[1:\tilde{b}:\tilde{c}],$
$[1:\tilde{g}:\tilde{h}],[\tilde{d}:1:\tilde{e}]\}$*, then the matrices,*

$$A=\begin{pmatrix} a(eg-h)+b(dhf'-ef)+c(f-dgf') & ae(f-1)+adh(1-f')+cd(f'-f) & adg(f'-1)+a(1-f)+bd(f-f') \\ beg(1-f)+bh(f'-1)+cg(f-f') & a(egf-hf')+b(dh-e)+c(f'-dgf) & ag(f'-f)+bdg(f-1)+b(1-f') \\ beh(f'-f)+ceg(1-f')+ch(f-1) & aeh(f-f')+cdh(1-f)+ce(f'-1) & a(egf'-hf)+b(dhf-ef')+c(1-dg) \end{pmatrix}$$

and

$$B=\begin{pmatrix} (\tilde{e}\tilde{g}-\tilde{h})+\tilde{b}(\tilde{d}\tilde{h}\tilde{f'}-\tilde{e}\tilde{f})+\tilde{c}(\tilde{f}-\tilde{d}\tilde{g}\tilde{f'}) & \tilde{e}(\tilde{f}-1)+\tilde{d}\tilde{h}(1-\tilde{f'})+\tilde{c}\tilde{d}(\tilde{f'}-\tilde{f}) & \tilde{d}\tilde{g}(\tilde{f'}-1)+(1-\tilde{f})+\tilde{b}\tilde{d}(\tilde{f}-\tilde{f'}) \\ \tilde{b}\tilde{e}\tilde{g}(1-\tilde{f})+\tilde{b}\tilde{h}(\tilde{f'}-1)+\tilde{c}\tilde{g}(\tilde{f}-\tilde{f'}) & (\tilde{e}\tilde{g}\tilde{f}-\tilde{h}\tilde{f'})+\tilde{b}(\tilde{d}\tilde{h}-\tilde{e})+\tilde{c}(\tilde{f'}-\tilde{d}\tilde{g}\tilde{f}) & \tilde{g}(\tilde{f'}-\tilde{f})+\tilde{b}\tilde{d}\tilde{g}(\tilde{f}-1)+\tilde{b}(1-\tilde{f'}) \\ \tilde{b}\tilde{e}\tilde{h}(\tilde{f'}-\tilde{f})+\tilde{c}\tilde{e}\tilde{g}(1-\tilde{f'})+\tilde{c}\tilde{h}(\tilde{f}-1) & \tilde{e}\tilde{h}(\tilde{f}-\tilde{f'})+\tilde{c}\tilde{d}\tilde{h}(1-\tilde{f})+\tilde{c}\tilde{e}(\tilde{f'}-1) & (\tilde{e}\tilde{g}\tilde{f'}-\tilde{h}\tilde{f})+\tilde{b}(\tilde{d}\tilde{h}\tilde{f}-\tilde{e}\tilde{f'})+\tilde{c}(1-\tilde{d}\tilde{g}) \end{pmatrix}$$

generate a free subgroup in $PGL_3(F_p((x)))$*, where* p *is a prime and* $d \in \mathbb{N}_0$

Exactly, for any pre-images of A and B in $GL_3(F_p((x)))$ also generate a free subgroup.

The following proposition tells about for which prime p and an integer d, the Free Generators Theorem will work.

Proposition 5.2.2. *Let p be a prime and $d \in \mathbb{N}_0$, if $p \geqslant 2$ and $d \geqslant 0$ then there exist $a, b, c, g, h, d, e, \tilde{b}, \tilde{c}, \tilde{d}, \tilde{e}, \tilde{g}, \tilde{h} \in F_p((x))$ and $f, f', \tilde{f}, \tilde{f}' \in F_p((x))^*$ such that the following conditions (i) and (ii) of the Free Generators theorem are satisfied.*
Further, if $a, b, c, g, h, d, e, \tilde{b}, \tilde{c}, \tilde{d}, \tilde{e}, \tilde{g}, \tilde{h} \in F_p((x))$,satisfy condition (i) then there exist $[k] \in \mathbb{P}^2$ such that satisfies the condition (iii) of the Free Generators Theorem.

Proof. We need to choose $f, f', \tilde{f}, \tilde{f}'$ are sufficiently small elements of $F_p((x))$ to satisfy the condition (ii).
So,we assume that $|f| = |f'| = |\tilde{f}| = |\tilde{f}'| = \frac{1}{p^{d+1}}$ of $F_p((x))$
Suppose if $p > 2$ and $d \geqslant 0$ and we take these vectors $[a : b : c], [1 : g : h]$, $[d : 1 : e], [1 : \tilde{b} : \tilde{c}], [1 : \tilde{g} : \tilde{h}], [\tilde{d} : 1 : \tilde{e}]$ where $a, b, c, g, h, d, e, \tilde{b}, \tilde{c}, \tilde{d}, \tilde{e}, \tilde{g}, \tilde{h} \in \{0, 1, 2\}$ then each pair of their distance must be greater than or equal to 1 and note that $1 > \frac{1}{p^{d+1}}$ for $p > 2$ and $d \geqslant 0$
Suppose, if $p = 2$ and $d \geqslant 0$ and we take these vectors $[1 : 1 : x], [x : 1 : 1]$, $[1 : 0 : 0], [0 : 0 : 1], [0 : 1 : 0], [1 : 1 : 1], [1 : 1 : 0], [1 : x : 0], [0 : 1 : x]$, for each pair of vectors is of distance greater than or equal to 1. Hence $1 > \frac{1}{p^{d+1}}$
By the proposition 4.2.6, we know that each of these $[a : b : c], [1 : g : h]$, $[d : 1 : e], [1 : \tilde{b} : \tilde{c}], [1 : \tilde{g} : \tilde{h}], [\tilde{d} : 1 : \tilde{e}]$ must each have $\frac{1}{p^{d+1}}$-neighbourhoods.

If $d = 0$ and $p \geqslant 2$, we see that $1 + 4p^{2d+2} = 1 + 4p^2 > 6$
and if $d > 0$ and $p \geqslant 2$, we see that $1 + 4p^{2d+2} > 6$.

Hence, in both cases, we see that there exist atleast one non-empty $\frac{1}{p^{d+1}}$-neighbourhood, namely N which is disjoint from the other $\frac{1}{p^{d+1}}$-neighbourhoods such as $[a:b:c]$, $[1:g:h], [d:1:e], [1:\tilde{b}:\tilde{c}], [1:\tilde{g}:\tilde{h}], [\tilde{d}:1:\tilde{e}]$ and we can say that $[k] \in \mathbb{P}^2$ be any element in N. Hence the condition (iii) is satisfied, ■

5.3. Proof of the Free Generators Theorem

The following Ping-Pong Lemma in [73] will give the proof of the Free Generators Theorem

Ping-Pong Lemma states that, "Let P be a set, L an index set, G a group acting on P, $(G_i)_{i\in L}$ a family of subgroups generating G. $(P_i)_{i\in L}$ a family of subsets of P and $[k]$ a point of $P - \{\cup_{i\in L} P_i\}$. Assume that for all $i, j \in L$ with $i \neq j$ and all $g \in G_i - \{I\}$, one has $g(P_j \cup \{[k]\}) \subset P_i$. Then G is the free product of the subgroups $G_i (i \in L)$"

Here,let us take that $P = \mathbb{P}^2$, $L = \{A, B\}$, $G_A =< A >$ and $G_B =< B >$,
G to be a subgroup of $PGL_3(F_p((x)))$ generated by G_A and G_B.
Let $(P_i)_{i\in L}$ be a family of subsets of $P = \mathbb{P}^2$,(i.e) P_A and P_B subsets of $P = \mathbb{P}^2$
For our discussion,let us assume $|f|, |f'|, |\tilde{f}|, |\tilde{f}'| \leqslant \frac{1}{p^{d+1}}$ by condition (ii) in theorem 5.2.1.

Let $\tilde{A}$ and $\tilde{B}$ be the pre-images of A and B in $GL_3(F_p((x)))$ and from lemma 5.1.1, we get that, $\tilde{A}$ has distinct eigenvectors $[a:b:c], [1:g:h], [d:1:e]$ with (upto scalar multiple) corresponding eigenvalues $x, y, z \in F_p((x))$ such that $\frac{y}{x} = f, \frac{z}{x} = f'$. Similarly, $\tilde{B}$ has distinct eigenvectors $[1:\tilde{b}:\tilde{c}], [1:\tilde{g}:\tilde{h}], [\tilde{d}:1:\tilde{e}]$ with correspond-

ing eigenvalues $\tilde{x}, \tilde{y}, \tilde{z} \in F_p((x))$ such that $\frac{\tilde{y}}{\tilde{x}} = \tilde{f}, \frac{\tilde{z}}{\tilde{x}} = \tilde{f}'$
Since we assume that the case $|f| \leqslant \frac{1}{p^{d+1}}$ and $|f^{'}| \leqslant \frac{1}{p^{d+1}}$, we get that the absolute value of the eigenvalue corresponding to $[a : b : c]$ is large, compared to the corresponding eigenvalue of $[1 : b : c]$.Thus, A will map elements of $\mathbb{P}^2$ towards $[a : b : c]$. A similar observation can be made for B, as we are assuming $|\tilde{f}| \leqslant \frac{1}{p^{d+1}}, |\tilde{f}^{'}| \leqslant \frac{1}{p^{d+1}}$. That is,B will map elements of $\mathbb{P}^2$ towards $[1 : \tilde{b} : \tilde{c}]$.

From the above eigenvectors, we will define,
$N_{[a:b:c]} = N([a : b : c], \frac{1}{p^{d+1}}), N_{[1:g:h]} = N([1 : g : h], \frac{1}{p^{d+1}})$

and $N_{[d:1:e]} = N([d : 1 : e], \frac{1}{p^{d+1}})$

Similarly,$N_{[1:\tilde{b}:\tilde{c}]} = N([1 : \tilde{b} : \tilde{c}], \frac{1}{p^{d+1}}), N_{[1:\tilde{g}:\tilde{h}]} = N([1 : \tilde{g} : \tilde{h}], \frac{1}{p^{d+1}})$

and $N_{[\tilde{d}:1:\tilde{e}]} = N([\tilde{d} : 1 : \tilde{e}], \frac{1}{p^{d+1}})$

Then we can choose, $P_A = N_{[a:b:c]} \cup N_{[1:g:h]} \cup N_{[d:1:e]}$ and $P_B = N_{[1:\tilde{b}:\tilde{c}]} \cup N_{[1:\tilde{g}:\tilde{h}]} \cup N_{[\tilde{d}:1:\tilde{e}]}$
Now, we want a point, $[k]$ in $P - \{\cup_{i \in L} P_i\} = P - \{P_A \cup P_B\}$ which is clear that the point exists from the condition (iii) in theorem 5.2.1.

To show that the conditions of Ping-Pong lemma are satisfied and to prove theorem 5.2.1, we need to prove for all $i, j \in L$ with $i \neq j$ and all $g \in G_i - \{I\}$,
one has $g(P_j \cup \{[k]\}) \subset P_i$. That is, $g \in G_A - I$, one has $g(P_B \cup \{[k]\}) \subset P_A$ and $g \in G_B - \{I\}$, one has $g(P_A \cup \{[k]\}) \subset P_B$
Note that if we replace $[a : b : c]$ with $[1 : \tilde{b} : \tilde{c}]$,$[1 : g : h]$ with $[1 : \tilde{g} : \tilde{h}]$, $[d : 1 : e]$

with $[\tilde{d} : 1 : \tilde{e}]$ and f with $\tilde{f}$ and $f^{'}$ with $\tilde{f}^{'}$, we obtain B from A.Thus, B has same form of A. So, we need to show that $g \in G_A - \{I\}$,one has $g(P_B \cup \{[k]\}) \subset P_A$.

We need to prove for any $g \in G_A - \{I\}$, one has $g(P_B \cup \{[k]\}) \subset P_A$

which leads to the strong property to prove as,

$A.(\mathbb{P}^2 - \{N([1 : g : h]) \cup N([d : 1 : e])\}) \subseteq N([a : b : c])$ and to prove that the inverse is like $A^{-1}.(\mathbb{P}^2 - \{N([a : b : c])\}) \subseteq \{N([1 : g : h]) \cup N([d : 1 : e])\}$

We prove the above results by the following sequence of lemmas.

Lemma 5.3.1. *If $A \in PGL_3(F_p((x)))$ is taken from Theorem 5.2.1 such that $|g|, |h|, |d|, |e| \leqslant p^d$ and $[e_3] \in \mathbb{P}^2 - \{N([1 : g : h]) \cup N([d : 1 : e])\}$.If any one of the conditions*

(i) $|b|, |c| \leqslant |a|$ with the property $p^d < |a| < |dg| \leqslant p^{2d}, |b|, |c| \leqslant 1$

(ii) $|a|, |c| \leqslant |b|$ with the property $p^d < |b| < |dg| \leqslant p^{2d}, |a|, |c| \leqslant 1$, and

(iii) $|a|, |b| \leqslant |c|$ with the property $p^d < |c| < |dg| \leqslant p^{2d}, |a|, |b| \leqslant 1$ are satisfied,

then $A.[e_3] \in N_{[a:b:c]}$

Proof. We know that,

$$\text{A}= \begin{bmatrix} a(eg-h)+b(dhf^{'}-ef)+c(f-dgf^{'}) & ae(f-1)+adh(1-f^{'})+cd(f^{'}-f) & adg(f^{'}-1)+a(1-f)+bd(f-f^{'}) \\ beg(1-f)+bh(f^{'}-1)+cg(f-f^{'}) & a(egf-hf^{'})+b(dh-e)+c(f^{'}-dgf) & ag(f^{'}-f)+bdg(f-1)+b(1-f^{'}) \\ beh(f^{'}-f)+ceg(1-f^{'})+ch(f-1) & aeh(f-f^{'})+cdh(1-f)+ce(f^{'}-1) & a(egf^{'}-hf)+b(dhf-ef^{'})+c(1-dg) \end{bmatrix}$$

We note that, $|f| \leqslant \dfrac{1}{p^{d+1}} < 1$, $|f^{'}| \leqslant \dfrac{1}{p^{d+1}} < 1$

$$\text{Now, } A.[e_3] = \begin{pmatrix} bd(f-f^{'}) - adg(1+f^{'}) + a(1-f) \\ bdg(f-1) + b(1-f^{'}) + ag(f^{'}-f) \\ b(dhf-ef^{'}) + c(1-dg) + a(egf^{'}-hf) \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}$$

where, $v_1 = bd(f-f^{'}) - adg(1+f^{'}) + a(1-f)$,

$v_2 = bdg(f-1) + b(1-f^{'}) + ag(f^{'}-f)$ and

$v_3 = b(dhf - ef^{'}) + c(1 - dg) + a(egf^{'} - hf)$

We know that the distance between two vectors are

$$\begin{aligned} d([a:b:c], A.[e_3]) &= d([a:b:c], [v_1:v_2:v_3]) \\ &= \frac{max\{|bv_3 - cv_2|, |cv_1 - av_3|, |av_2 - bv_1|\}}{max\{|a|, |b|, |c|\} max\{|v_1|, |v_2|, |v_3|\}} \end{aligned}$$

Since, by the non-Archimedean property, $|x| \neq |y|$ then $|x+y| = max\{|x|, |y|\}$ we get the values of v_1, v_2 and v_3

Now,$|v_1| = max\{|bd(f - f^{'})|, |adg(1 + f^{'})|, |a(1 - f)|\} = max\{|bd(f - f^{'})|, |adg|, |a|\}$
(since, $|adgf^{'}| < |adg|, |af| < |a|$)
Thus, $|v_1| = max\{|bd(f - f^{'})|, |adg|, |a|\}$
Similarly for,

$$\begin{aligned} |v_2| &= max\{|bdg(f-1)|, |b(1-f^{'})|, |ag(f^{'} - f)|\} \\ &= max\{|bdg|, |b|, |ag(f^{'} - f)|\} \end{aligned}$$

$$\begin{aligned} |v_3| &= max\{|b(dhf - ef^{'})|, |c(1-dg)|, |a(egf^{'} - hf)|\} \\ &= max\{|bdhf|, |bef^{'}|, |c|, |cdg|, |aegf^{'}|, |ahf|\} \end{aligned}$$

We know that $|dg| > 1$ then $|adg| > |a|, |bdg| > |b|$ and $|cdg| > |c|$. Now

$$\begin{aligned} max\{|v_1|, |v_2|, |v_3|\} &= max\{|adg|, |a|, |bdg|, |b|, |cdg|, |c|\} \\ &= max\{|adg|, |bdg|, |cdg|\} \end{aligned} \tag{5.1}$$

Since,$|bd(f-f^{'})| < |b|, |bdhf| < |b|, |bef^{'}| < |b|, |ag(f-f^{'})| < |a|,$
$|aegf^{'}| < |a|, |ahf| < |a|$

Now to find the $max\{|bv_3 - cv_2|, |cv_1 - av_3|, |av_2 - bv_1|\}$

$$\begin{aligned}
|bv_3 - cv_2| &= |b^2(dhf - ef^{'}) + bc(1-dg) + ab(egf^{'} - hf) - cbdg(f-1) \\
&\quad -cb(1-f^{'}) - acg(f^{'} - f)| \\
&= |acg(f-f^{'}) - bcdgf + ab(egf^{'} - hf) + b^2(dhf - ef^{'}) + bcf^{'}| \\
&\leqslant max\{|acg(f-f^{'})|, |bcdgf|, |ab(egf^{'} - hf)|, |b^2(dhf - ef^{'})|, |bcf^{'}|\} \\
&< max\{|ac|, |bcdg|, |ab|, |b^2|, |bc|\} < max\{|ac|, |bcdg|, |ab|, |b^2|\} \\
&\quad (since, |dg| > 1 \Rightarrow |bcdg| > |bc|)
\end{aligned}$$

$$\begin{aligned}
|cv_1 - av_3| &= |bcd(f-f^{'}) - acdg(1+f^{'}) + ac(1-f) - ab(dhf - ef^{'}) \\
&\quad +ac(1-dg) + a^2(egf^{'} - hf)| \\
&= |bc(df - df^{'}) + a^2(hf - egf^{'}) + ab(ef^{'} - dhf) - acdgf^{'} - acf| \\
&\leqslant max\{|bc(df - df^{'})|, |a^2(hf - egf^{'})|, |ab(ef^{'} - dhf)|, |acdgf^{'}|, |acf|\} \\
&< max\{|bc|, |a^2|, |ab|, |acdg|, |ac|\} < max\{|acdg|, |bc|, |ab|, |a^2|\} \\
&\quad (since, |dg| > 1 \Rightarrow |acdg| > |ac|)
\end{aligned}$$

$$
\begin{aligned}
|av_2 - bv_1| &= |abdg(f-1) + ab(1-f^{'}) + a^2g(f^{'}-f) - b^2d(f-f^{'}) \\
&\quad -abdg(1+f^{'}) + ab(1-f)| \\
&= |ab(f-f^{'}) + abdg(f+f^{'}) + b^2(df^{'}-df) + a^2g(f^{'}-f)| \\
&\leqslant max\{|ab(f-f^{'})|, |abdg(f+f^{'})|, |b^2(df^{'}-df)|, |a^2g(f^{'}-f)|\} \\
&< max\{|abdg|, |a^2|, |b^2|\} \\
&\quad (since, |dg| > 1 \Rightarrow |abdg| > |abg| > |ab|)
\end{aligned}
$$

Thus, we have,

$max\{|bv_3 - cv_2|, |cv_1 - av_3|, |av_2 - bv_1|\} \leqslant max\{|bcdg|, |acdg|, |abdg|, |a^2|, |b^2|\}$

(since,$|bcdg| > |bc|, |acdg| > |ac|, |abdg| > |ab|$).

Hence, we are using (5.1)

$$
\begin{aligned}
d([a:b:c], A.[e_3]) &= d([a:b:c], [v_1:v_2:v_3]) \\
&= \frac{max\{|bv_3 - cv_2|, |cv_1 - av_3|, |av_2 - bv_1|\}}{max\{|a|, |b|, |c|\}max\{|v_1|, |v_2|, |v_3|\}} \\
&\leqslant \frac{max\{|bcdg|, |acdg|, |abdg|, |a^2|, |b^2|\}}{max\{|a|, |b|, |c|\}max\{|adg|, |bdg|, |cdg|\}} \\
&= \frac{max\{|bcdg|, |acdg|, |abdg|, |a^2|, |b^2|\}}{max\{|a^2dg|, |b^2dg|, |c^2dg|\}}
\end{aligned}
$$

Case(i): If $|b|, |c| \leqslant |a|$ with the property $p^d < |a| < |dg| \leqslant p^{2d}$, $|b|, |c| \leqslant 1$ then

$$\begin{aligned} d([a:b:c], A.[e_3]) &\leqslant \frac{max\{|a^2|, |acdg|, |abdg|\}}{|a^2dg|} \\ &= \frac{max\{|a|, |cdg|, |bdg|\}}{|adg|} \leqslant \frac{max\{|a|, |dg|\}}{|adg|} < \frac{|dg|}{|adg|} \\ &< \frac{1}{|a|} < \frac{1}{p^d} \leqslant \frac{1}{p^{d+1}} \end{aligned}$$

Case(ii): If $|a|, |c| \leqslant |b|$ with the property $p^d < |b| < |dg| \leqslant p^{2d}$, $|a|, |c| \leqslant 1$ then

$$\begin{aligned} d([a:b:c], A.[e_3]) &\leqslant \frac{max\{|b^2|, |bcdg|, |abdg|\}}{|b^2dg|} \\ &= \frac{max\{|b|, |cdg|, |adg|\}}{|bdg|} \leqslant \frac{max\{|b|, |dg|\}}{|bdg|} < \frac{|dg|}{|bdg|} \\ &< \frac{1}{|b|} < \frac{1}{p^d} \leqslant \frac{1}{p^{d+1}} \end{aligned}$$

Case(iii): If $|a|, |b| \leqslant |c|$ with the property $p^d < |c| < |dg| \leqslant p^{2d}$,$|a|, |b| \leqslant 1$ then

$$\begin{aligned} d([a:b:c], A.[e_3]) &\leqslant \frac{max\{|bcdg|, |acdg|\}}{|c^2dg|} \\ &= \frac{max\{|bdg|, |adg|\}}{|cdg|} \leqslant \frac{max\{|dg|\}}{|cdg|} \\ &\leqslant \frac{1}{|c|} < \frac{1}{p^d} \leqslant \frac{1}{p^{d+1}} \end{aligned}$$

Hence if any one of the conditions are satisfied then,we get
$d([a:b:c], A.[e_3]) \leqslant \frac{1}{p^{d+1}}$. Therefore,we get $A.[e_3] \in N_{[a:b:c]}$. ■

Lemma 5.3.2. *Let $A \in PGL_3(F_p((x)))$ is taken from Theorem 5.2.1 such that $|g|, |h|, |d|, |e| \leqslant p^d$ and $[1 : x : y] \in \mathbb{P}^2 - \{N([1 : g : h]) \cup N([d : 1 : e])\}$.If any one of the conditions*

(i) $|b|, |c| \leqslant |a|$ with the property $p^d < |a| < |dh| \leqslant p^{2d}, |b|, |c| \leqslant 1$,

(ii) $|a|, |c| \leqslant |b|$ with the property $p^d < |b| < |dh| \leqslant p^{2d}, |a|, |c| \leqslant 1$ and

(iii) $|a|, |b| \leqslant |c|$ with the property $p^d < |c| < |dh| \leqslant p^{2d}, |a|, |b| \leqslant 1$ are satisfied, then $A.[1 : x : y] \in N_{[a:b:c]}$

Proof.We know that,

$$\text{A}= \left[\begin{pmatrix} a(eg-h)+b(dhf^{'}-ef)+c(f-dgf^{'}) & ae(f-1)+adh(1-f^{'})+cd(f^{'}-f) & adg(f^{'}-1)+a(1-f)+bd(f-f^{'}) \\ beg(1-f)+bh(f^{'}-1)+cg(f-f^{'}) & a(egf-hf^{'})+b(dh-e)+c(f^{'}-dgf) & ag(f^{'}-f)+bdg(f-1)+b(1-f^{'}) \\ beh(f^{'}-f)+ceg(1-f^{'})+ch(f-1) & aeh(f-f^{'})+cdh(1-f)+ce(f^{'}-1) & a(egf^{'}-hf)+b(dhf-ef^{'})+c(1-dg) \end{pmatrix} \right]$$

Now,

$$A.[1 : x : y] = \begin{bmatrix} c(f-dgf^{'}+dxf^{'}-dxf)+a(eg-h+exf+dhx-dhxf^{'}+dgyf^{'}-dgy+y-yf)+b(dhf^{'}-ef+dyf-dyf^{'}) \\ c(gf-gf^{'}+ef^{'}-dgxf)+a(gexf-hxf^{'}+gyf)+b(eg-egf+hf^{'}-h+dhx-xe+dgyf-dgy+y-yf^{'}) \\ c(eg-egf^{'}+hf-h+dhx-dhxf+exf^{'}-ex+y-dgy)+a(hxef-hxef^{'}+egyf^{'}-hyf)+b(hef^{'}-hef+dhyf-yef^{'}) \end{bmatrix} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}$$

where $v_1 = c(f-dgf^{'}+dxf^{'}-dxf)+a(eg-h+exf+dhx-dhxf^{'}+dgyf^{'}-dgy+y-yf)+b(dhf^{'}-ef+dyf-dyf^{'})$,
$v_2 = c(gf-gf^{'}+ef^{'}-dgxf)+a(gexf-hxf^{'}+gyf)+b(eg-egf+hf^{'}-h+dhx-xe+dgyf-dgy+y-yf^{'})$ and

$v_3 = c(eg - egf^{'} + hf - h + dhx - dhxf + exf^{'} - ex + y - dgy) + a(hxef - hxef^{'}$
$+ egyf^{'} - hyf) + b(hef^{'} - hef + dhyf - yef^{'})$

We first noted that the point $[1 : x : y]$ is in $\mathbb{P}^2 - \{N([1 : g : h]) \cup N([d : 1 : e])\}$,
if $|g|, |h|, |d|, |e| \leqslant p^d$ and $|x| > 1, |y| \leqslant 1$ and
assume that $|f| \leqslant \frac{1}{p^{d+1}} < 1, |f^{'}| \leqslant \frac{1}{p^{d+1}} < 1$
Since, by the non-Archimedean property $|x| \neq |y|$ then $|x + y| = max\{|x|, |y|\}$,
we get the values of v_1, v_2 and v_3. Now,

$$\begin{aligned}
|v_1| &= |c(f - dgf^{'} + dxf^{'} - dxf) + a(eg - h + exf + dhx - dhxf^{'} + \\
&\quad dgyf^{'} - dgy + y - yf) + b(dhf^{'} - ef + dyf - dyf^{'})| \\
&= |c(f - dgf^{'}) + cxd(f^{'} - f) + ae(g - xf) + adhx(1 - f^{'}) + adgy(f^{'} - 1) \\
&\quad + ay(1 - f) + b(dhf^{'} - ef) + bdy(f - f^{'})| \\
&= max\{|adhx|, |adgy|, |aeg|, |cdx(f^{'} - f)|, |bdy(f - f^{'})|, |c(f - dgf^{'})|\} \\
&= max\{|adhx|, |bdyf|, |cdgf^{'}|\}
\end{aligned}$$

Similarly, for $|v_2|$ and $|v_3|$

$$\begin{aligned}
|v_2| &= |c(gf - gf^{'} + ef^{'} - dgxf) + a(gexf - hxf^{'} + gyf) + b(eg - egf + hf^{'} - h \\
&\quad + dhx - xe + dgyf - dgy + y - yf^{'})| \\
&= |cg(f - f^{'}) + c(ef^{'} - dgx) + ax(gef - hf^{'}) + agyf + beg(1 - f) + \\
&\quad bh(f^{'} - 1) + bx(dh - e) + bdgy(f - 1) + by(1 - f^{'})| \\
&= max\{|beg|, |bh|, |bdgy|, |by|, |bdhx|, |ax(gef - hf^{'})|, |cdgx|, |cg(f - f^{'})|\} \\
&= max\{|beg|, |bdgy|, |bdhx|, |agexf|, |cef^{'}|, |cgf|, |cgf^{'}|\}
\end{aligned}$$

$$
\begin{aligned}
|v_3| &= |c(eg - egf^{'} + hf - h + dhx - dhxf + exf^{'} - ex + y - dgy) + a \\
&\quad (hxef - hxef^{'} + egyf^{'} - hyf) + b(hef^{'} - hef + dhyf - yef^{'})| \\
&= |ceg(1 - f^{'}) + ch(f - 1) + cdhx(1 - f) + cex(f^{'} - 1) + cy(1 - dg) \\
&\quad + axeh(f - f^{'}) + ay(egf^{'} - hf) + bhe(f^{'} - f) + by(dhf - ef^{'})| \\
&= max\{|ceg|, |ch|, |cdhx|, |cxe|, |cy|, |ahex(f - f^{'})|, |ay(egf^{'} - hf)|, \\
&\quad |bhe(f^{'} - f)|, |by(dhf - ef^{'})|\} \\
&= max\{|ceg|, |cdhx|, |axehf|, |axehf^{'}|, |bhef|, |bhef^{'}|\}
\end{aligned}
$$

Hence

$$
\begin{aligned}
max\{|v_1|, |v_2|, |v_3|\} &= max\{max\{|adhx|, |bdyf|, |cdgf^{'}|\}, \\
&\quad max\{|beg|, |bdgy|, |bdhx|, |agexf|, |cef^{'}|, |cgf|, |cgf^{'}|\}, \\
&\quad max\{|ceg|, |cdhx|, |axehf|, |axehf^{'}|, |bhef|, |bhef^{'}|\}\} \\
&= max\{|adhx|, |bdhx|, |cdhx|\}
\end{aligned}
$$

We know that the distance between two vectors are,

$$
\begin{aligned}
d([a : b : c], A.[1 : x : y]) &= d([a : b : c], [v_1 : v_2 : v_3]) \\
&= \frac{max\{|bv_3 - cv_2|, |cv_1 - av_3|, |av_2 - bv_1|\}}{max\{|a|, |b|, |c|\}max\{|v_1|, |v_2|, |v_3|\}} \\
&= \frac{max\{|bv_3 - cv_2|, |cv_1 - av_3|, |av_2 - bv_1|\}}{max\{|a|, |b|, |c|\}max\{|adhx|, |bdhx|, |cdhx|\}}
\end{aligned}
$$

Now we have to find the $max\{|bv_3 - cv_2|, |cv_1 - av_3|, |av_2 - bv_1|\}$

$$
\begin{aligned}
|bv_3 - cv_2| &= |bc(eg - egf^{'} + hf - h + dhx - dhxf + exf^{'} - ex + y - dgy) + ab(hxef - hxef^{'} + egyf^{'} - \\
&\quad hyf) + b^2(hef^{'} - hef + dhyf - yef^{'}) - c^2(gf - gf^{'} + ef^{'} - dgxf) + ac(gexf - hxf^{'} + \\
&\quad gyf) + bc(eg - egf + hf^{'} - h + dhx - xe + dgyf - dgy + y - yf^{'})| \\
&= |bc(hf - egf^{'} + egf - hf^{'} - dgyf) + bcx(dhf - ef^{'}) + abx(ehf - ehf^{'}) + aby(egf^{'} - \\
&\quad hf) + b^2he(f^{'} - f) + b^2y(dhf - ef^{'}) + c^2g(f^{'} - f) - c^2ef^{'} + c^2xdgf + acx(gef - \\
&\quad hf^{'}) - acgyf| \\
&\leqslant max\{|bc(hf - egf^{'} + egf - hf^{'} - dgyf)|, |bcx(dhf - ef^{'})|, |abx(ehf - ehf^{'})|, \\
&\quad |aby(egf^{'} - hf)|, |b^2he(f^{'} - f)|, |b^2y(dhf - ef^{'})|, |c^2g(f^{'} - f)|, |c^2ef^{'}|, |c^2xdgf|, \\
&\quad |acx(gef - hf^{'})|, |acgyf|\} \\
&\leqslant max\{|bcdhx|, |abx|, |b^2x|, |acx|, |c^2x|\}
\end{aligned}
$$

$$
\begin{aligned}
|cv_1 - av_3| &= |c^2(f - dgf^{'} + dxf^{'} - dxf) + ac(eg - h + exf + dhx - dhxf^{'} + dgyf^{'} - dgy + y - \\
&\quad yf) + bc(dhf^{'} - ef + dyf - dyf^{'}) - ac(eg - egf^{'} + hf - h + dhx - dhxf + exf^{'} - \\
&\quad ex + y - dgy) + a^2(hxef - hxef^{'} + egyf^{'} - hyf) + ba(hef^{'} - hef + dhyf - yef^{'})| \\
&= |c^2(f - dgf^{'} + dxf^{'} - dxf) + acx(ef - dhf^{'}) + acy(dgf^{'} - f) + bc(dhf^{'} - ef) + \\
&\quad bcy(df - df^{'}) + ac(egf^{'} - hf) + acx(dhf - ef^{'} + e) + a^2x(hef^{'} - hef) + a^2y(hf - \\
&\quad egf^{'}) + ab(hef - hef^{'}) + aby(ef^{'} - dh)| \\
&\leqslant max\{|c^2(f - dgf^{'} + dxf^{'} - dxf)|, |acx(ef - dhf^{'})|, |acy(dgf^{'} - f)|, |bc(dhf^{'} - ef)|, \\
&\quad |bcy(df - df^{'})|, |ac(egf^{'} - hf)|, |acx(dhf - ef^{'} + e)|, |a^2x(hef^{'} - hef)|, |a^2y(hf - \\
&\quad egf^{'})|, |ab(hef - hef^{'})|, |aby(ef^{'} - dh)|\} \\
&\leqslant max\{|c^2x|, |bcx|, |acdhx|, |a^2x|, |abx|\}
\end{aligned}
$$

$$
\begin{aligned}
|av_2 - bv_1| &= |ac(gf - gf^{'} + ef^{'} - dgxf) + a^2(gexf - hxf^{'} + gyf) + ab(eg - egf + hf^{'} - h + dhx - \\
&\quad xe + dgyf - dgy + y - yf^{'}) - bc(f - dgf^{'} + dxf^{'} - dxf) + ab(eg - h + exf + dhx - \\
&\quad dhxf^{'} + dgyf^{'} - dgy + y - yf) + b^2(dhf^{'} - ef + dyf - dyf^{'})| \\
&= |acg(f - f^{'}) + ac(ef^{'} - dgxf) + a^2x(gef - hf^{'}) + a^2gyf + ab(hf^{'} - egf + gyf - \\
&\quad yf^{'}) + bc(dgf^{'} - f - dxf^{'} + dxf) - abex(f - 1) + abd(hxf^{'} - gyf^{'}) + b^2(ef - dhf^{'} - \\
&\quad dyf + dyf^{'})| \\
&\leqslant max\{|acg(f - f^{'})|, |ac(ef^{'} - dgxf)|, |a^2x(gef - hf^{'})|, |a^2gyf|, |ab(hf^{'} - egf + gyf - \\
&\quad yf^{'})|, |bc(dgf^{'} - f - dxf^{'} + dxf)|, |abex(f - 1)|, |abd(hxf^{'} - gyf^{'})|, |b^2(ef - \\
&\quad dhf^{'} - dyf + dyf^{'})|\} \\
&\leqslant max\{|acx|, |a^2x|, |abdhx|, |bcx|, |b^2x|\}
\end{aligned}
$$

Hence,we have

$$\begin{aligned} d([a:b:c],A.[1:x:y]) &= d([a:b:c],[v_1:v_2:v_3]) \\ &= \frac{max\{|bv_3-cv_2|,|cv_1-av_3|,|av_2-bv_1|\}}{max\{|a|,|b|,|c|\}max\{|v_1|,|v_2|,|v_3|\}} \\ &\leqslant \frac{max\{|a^2|,|b^2|,|c^2|,|abdhx|,|acdhx|,|bcdhx|,|abx|,|acx|,|bcx|\}}{max\{|a|,|b|,|c|\}max\{|adhx|,|bdhx|,|cdhx|\}} \end{aligned}$$

Case(i): If $|b|,|c| \leqslant |a|$ with the property $p^d < |a| < |dh| \leqslant p^{2d}, |b|,|c| \leqslant 1$ then

$$\begin{aligned} d([a:b:c],A.[1:x:y]) &= d([a:b:c],[v_1:v_2:v_3]) \\ &\leqslant \frac{max\{|a^2|,|b^2|,|c^2|,|abdhx|,|acdhx|,|bcdhx|,|acx|,|abx|,|bcx|\}}{max\{|a|,|b|,|c|\}max\{|adhx|,|bdhx|,|cdhx|\}} \\ &\leqslant \frac{max\{|a^2|,|abdhx|,|acdhx|,|acx|,|abx|\}}{|a^2dhx|} \\ &< \frac{max\{|adh|,|abdhx|,|acdhx|\}}{|a^2dhx|} < \frac{max\{|dh|,|bdhx|,|cdhx|\}}{|adhx|} \\ &\quad (since, |dh|>1 \Rightarrow |abdhx|>|abx|, |acdhx|>|acx|) \\ &< \frac{max\{1,|bx|,|cx|\}}{|ax|} \leqslant \frac{max\{1,|x|\}}{|ax|} < \frac{1}{|ax|} or \frac{|x|}{|ax|} \\ &\quad (since, |ax|>|a|) \\ &< \frac{1}{|a|} or \frac{1}{|a|} < \frac{1}{p^d} \leqslant \frac{1}{p^{d+1}} \end{aligned}$$

Case(ii): If $|a|, |c| \leqslant |b|$ with the property $p^d < |b| < |dh| \leqslant p^{2d}, |a|, |c| \leqslant 1$ then

$$
\begin{aligned}
d([a:b:c], A.[1:x:y]) &= d([a:b:c], [v_1:v_2:v_3]) \\
&\leqslant \frac{max\{|a^2|, |b^2|, |c^2|, |abdhx|, |acdhx|, |bcdhx|, |acx|, |abx|, |bcx|\}}{max\{|a|, |b|, |c|\} max\{|adhx|, |bdhx|, |cdhx|\}} \\
&\leqslant \frac{max\{|b^2|, |abdhx|, |bcdhx|, |bcx|, |abx|\}}{|b^2dhx|} \\
&< \frac{max\{|bdh|, |abdhx|, |bcdhx|\}}{|b^2dhx|} \\
&\quad (since, |dh| > 1 \Rightarrow |abdhx| > |abx|, |bcdhx| > |bcx|) \\
&< \frac{max\{|dh|, |adhx|, |cdhx|\}}{|bdhx|} \\
&< \frac{max\{1, |ax|, |cx|\}}{|bx|} \leqslant \frac{max\{1, |x|\}}{|bx|} \\
&< \frac{1}{|bx|} or \frac{|x|}{|bx|} \\
&\quad (since, |bx| > |b|) \\
&< \frac{1}{|b|} or \frac{1}{|b|} \\
&< \frac{1}{p^d} \leqslant \frac{1}{p^{d+1}}
\end{aligned}
$$

Case(iii): If $|a|, |b| \leqslant |c|$ with the property $p^d < |c| < |dh| \leqslant p^{2d}$, $|a|, |b| \leqslant 1$ then

$$\begin{aligned}
d([a:b:c], A.[1:x:y]) &= d([a:b:c], [v_1:v_2:v_3]) \\
&\leqslant \frac{max\{|a^2|, |b^2|, |c^2|, |abdhx|, |acdhx|, |bcdhx|, |acx|, |abx|, |bcx|\}}{max\{|a|, |b|, |c|\} max\{|adhx|, |bdhx|, |cdhx|\}} \\
&\leqslant \frac{max\{|c^2|, |acdhx|, |bcdhx|, |bcx|, |acx|\}}{|c^2dhx|} \\
&< \frac{max\{|cdh|, |acdhx|, |bcdhx|\}}{|c^2dhx|} \\
&\quad (since, |dh| > 1 \Rightarrow |acdhx| > |acx|, |bcdhx| > |bcx|) \\
&< \frac{max\{|dh|, |adhx|, |bdhx|\}}{|cdhx|} \\
&< \frac{max\{1, |ax|, |bx|\}}{|cx|} \leqslant \frac{max\{1, |x|\}}{|cx|} \\
&< \frac{1}{|cx|} or \frac{|x|}{|cx|} \\
&\quad (since, |cx| > |c|) \\
&< \frac{1}{|c|} or \frac{1}{|c|} < \frac{1}{p^d} \leqslant \frac{1}{p^{d+1}}
\end{aligned}$$

Hence in all cases with any one of the conditions in the hypothesis are satisfied, then $d([a:b:c], A.[1:x:y]) \leqslant \frac{1}{p^{d+1}}$. Therefore,we get $A.[1:x:y] \in N_{[a:b:c]}$.

■

Similarly we can prove the result for $A.[x:1:y] \in N([a:b:c])$

Lemma 5.3.3. *Let $A \in PGL_3(F_p((x)))$ is taken from Theorem 5.2.1 such that $|g|, |h|, |d|, |e| \leqslant p^d$ and $[x:1:y] \in \mathbb{P}^2 - \{N([1:g:h]) \cup N([d:1:e])\}$. If any one of the following conditions ,*

(i) $|b|, |c| \leqslant |a|$ with the property $p^d < |a| < |eg| \leqslant p^{2d}, |b|, |c| \leqslant 1$,

(ii) $|a|, |c| \leqslant |b|$ with the property $p^d < |b| < |eg| \leqslant p^{2d}, |a|, |c| \leqslant 1$ and

(iii) $|a|, |b| \leqslant |c|$ with the property $p^d < |c| < |eg| \leqslant p^{2d}, |a|, |b| \leqslant 1$ are satisfied then, $A.[x:1:y] \in N_{[a:b:c]}$.

Proof.We know that,

$$\text{A}= \left[\begin{pmatrix} a(eg-h)+b(dhf'-ef)+c(f-dgf') & ae(f-1)+adh(1-f')+cd(f'-f) & adg(f'-1)+a(1-f)+bd(f-f') \\ beg(1-f)+bh(f'-1)+cg(f-f') & a(egf-hf')+b(dh-e)+c(f'-dgf) & ag(f'-f)+bdg(f-1)+b(1-f') \\ beh(f'-f)+ceg(1-f')+ch(f-1) & aeh(f-f')+cdh(1-f)+ce(f'-1) & a(egf'-hf)+b(dhf-ef')+c(1-dg) \end{pmatrix} \right]$$

and Now,

$$A.[x:1:y] = \begin{bmatrix} a(xeg-xh+ef-e+dh-dhf'+dgyf'-dgy+y-yf)+b(dhxf'-exf+dyf-dyf')+c(xf-dgxf'+df'-df) \\ a(gef-hf'+gyf'-gyf)+b(egx-egxf+hxf'-hx+dh-e+dgyf-dgy+y-yf')+c(gxf-gxf'+f'-dgf) \\ a(hef-hef'+geyf'-hyf)+b(hxef'-hxef+dhyf-eyf')+c(gex-gexf'+hxf-hx+dh-dhf+ef'-e) \end{bmatrix} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}$$

where, $v_1 = a(xeg-xh+ef-e+dh-dhf'+dgyf'-dgy+y-yf)+b(dhxf'-exf+dyf-dyf')+c(xf-dgxf'+df'-df)$,

$v_2 = a(gef-hf'+gyf'-gyf)+b(egx-egxf+hxf'-hx+dh-e+dgyf-dgy+y-yf')+c(gxf-gxf'+f'-dgf)$,

$v_3 = a(hef - hef^{'} + geyf^{'} - hyf) + b(hxef^{'} - hxef + dhyf - eyf^{'}) + c(gex - gexf^{'} + hxf - hx + dh - dhf + ef^{'} - e)$

We first noted that the point $[x : 1 : y]$ is in $\mathbb{P}^2 - \{N([1 : g : h]) \cup N([d : 1 : e])\}$, if $|g|, |h|, |d|, |e| \leqslant p^d$ and $|x| > 1, |y| \leqslant 1$ and $|f| \leqslant \frac{1}{p^{d+1}} < 1, |f^{'}| \leqslant \frac{1}{p^{d+1}} < 1$

Since, by the non-Archimedean property $|x| \neq |y|$ then $|x + y| = max\{|x|, |y|\}$, we get the values of v_1, v_2 and v_3.

$$\begin{aligned}
|v_1| &= |a(xeg - xh + ef - e + dh - dhf^{'} + dgyf^{'} - dgy + y - yf) + b(dhxf^{'} \\
&\quad -exf + dyf - dyf^{'}) + c(xf - dgxf^{'} + df^{'} - df)| \\
&= |ax(eg - h) + ae(f - 1) + adh(1 - f^{'}) + adgy(f^{'} - 1) + ay(1 - f) \\
&\quad +bx(dhf^{'} - ef) + bgy(f - f^{'}) + cx(f - dgf^{'}) + cd(f^{'} - f)| \\
&= max\{|ae|, |ax(eg - h)|, |adh|, |adgy|, |ay|, |bx(dhf^{'} - ef)|, |bdgy(f - f^{'})|, \\
&\quad |cx(f - dgf^{'})|, |cd(f^{'} - f)|\} \\
&= max\{|aegx|, |bxdhf^{'}|, |cxdgf^{'}|\}
\end{aligned}$$

Similarly for $|v_2|, |v_3|$,

$$\begin{aligned}
|v_2| &= |a(gef - hf^{'} + gyf^{'} - gyf) + b(egx - egxf + hxf^{'} - hx + dh - e \\
&\quad +dgyf - dgy + y - yf^{'}) + c(gxf - gxf^{'} + f^{'} - dgf)| \\
&= |a(egf - hf^{'}) + agy(f^{'} - f) + begx(1 - f) + bhx(f^{'} - 1) + bdh - be + \\
&\quad bdgy(f - 1) + by(1 - f^{'}) + cgx(f - f^{'}) + c(f^{'} - dgf)| \\
&= max\{|begx|, |bhx|, |bdh|, |be|, |bdgy|, |by|, |a(gef - hf^{'})|, |agy(f^{'} - f)|, \\
&\quad |cdgf|, |cgx(f - f^{'})|, |cf^{'}|\} \\
&= max\{|begx|, |agef|, |agyf^{'}|, |agyf|, |cdgf|\}
\end{aligned}$$

and

$$\begin{aligned}
|v_3| &= |a(hef - hef^{'} + geyf^{'} - hyf) + b(hxef^{'} - hxef + dhyf \\
&\quad -eyf^{'}) + c(gex - gexf^{'} + hxf - hx + dh - dhf + ef^{'} - e)| \\
&= |ahe(f - f^{'}) + ay(f^{'}ge - hf) + bhxe(f^{'} - f) + by(dhf - ef^{'}) \\
&\quad +cgex(1 - f^{'}) + chx(f - 1) + cdh(1 - f) + ce(f^{'} - 1)| \\
&= max\{|cegx|, |chx|, |cdh|, |ce|, |ahe(f - f^{'})|, |ay(egf^{'} - hf)|, \\
&\quad |bhe(f^{'} - f)|, |by(dhf - ef^{'})|\} \\
&= max\{|cegx|, |cdh|, |aehf|, |ahef^{'}|, |ayegf^{'}|, |bxhef|, |bxhef^{'}|\}
\end{aligned}$$

Hence,

$$\begin{aligned}
max\{|v_1|, |v_2|, |v_3|\} &= max\{max\{|aegx|, |bxdhf^{'}|, |cxdgf^{'}|\}, \\
&\quad max\{|begx|, |agef|, |agyf^{'}|, |agyf|, |cdgf|\}, max\{|cegx|, \\
&\quad |cdh|, |aehf|, |ahef^{'}|, |ayegf^{'}|, |bxhef|, |bxhef^{'}|\}\} \\
&= max\{|aegx|, |begx|, |cegx|\}
\end{aligned}$$

We know that the distance between two vectors are,

$$\begin{aligned}
d([a : b : c], A.[x : 1 : y]) &= d([a : b : c], [v_1 : v_2 : v_3]) \\
&= \frac{max\{|bv_3 - cv_2|, |cv_1 - av_3|, |av_2 - bv_1|\}}{max\{|a|, |b|, |c|\}max\{|v_1|, |v_2|, |v_3|\}} \\
&= \frac{max\{|bv_3 - cv_2|, |cv_1 - av_3|, |av_2 - bv_1|\}}{max\{|a|, |b|, |c|\}max\{|aegx|, |begx|, |cegx|\}}
\end{aligned}$$

Now we have to find the $max\{|bv_3 - cv_2|, |cv_1 - av_3|, |av_2 - bv_1|\}$

$$
\begin{aligned}
|bv_3 - cv_2| &= |b(a(hef - hef^{'} + geyf^{'} - hyf) + b(hxef^{'} - hxef + dhyf - eyf^{'}) + c(gex - gexf^{'} + \\
&\quad hxf - hx + dh - dhf + ef^{'} - e)) - c(a(gef - hf^{'} + gyf^{'} - gyf) + b(egx - egxf + \\
&\quad hxf^{'} - hx + dh - e + dgyf - dgy + y - yf^{'}) + c(gxf - gxf^{'} + f^{'} - dgf))| \\
&= |abhe(f - f^{'}) + aby(gef^{'} - hf) + b^2hxe(f^{'} - f) + b^2y(dhf - ef^{'}) + bcx(gef^{'} - hf + \\
&\quad egf - hf^{'}) + bc(ef^{'} - dhf + dgy - dgyf - y + yf^{'}) + ac(gef - hf^{'} - gyf^{'} - gyf) + \\
&\quad c^2(gxf^{'} - gxf - f^{'}dgf)| \\
&\leqslant max\{|abhe(f - f^{'})|, |aby(gef^{'} - hf)|, |b^2hxe(f^{'} - f)|, |b^2y(dhf - ef^{'})|, |bcx(gef^{'} - \\
&\quad hf + egf - hf^{'})|, |bc(ef^{'} - dhf + dgy - dgyf - y + yf^{'})|, |ac(gef - hf^{'} - gyf^{'} - \\
&\quad gyf)|, |c^2(gxf^{'} - gxf - f^{'}dgf)|\} \\
&\leqslant max\{|bcegx|, |ab|, |b^2x|, |ac|, |c^2x|\}
\end{aligned}
$$

$$
\begin{aligned}
|cv_1 - av_3| &= |c(a(xeg - xh + ef - e + dh - dhf^{'} + dgyf^{'} - dgy + y - yf) + b(dhxf^{'} - exf + \\
&\quad dyf - dyf^{'}) + c(xf - dgxf^{'} + df^{'} - df)) - a(a(hef - hef^{'} + geyf^{'} - hyf) + b(hxef^{'} - \\
&\quad hxef + dhyf - eyf^{'}) + c(gex - gexf^{'} + hxf - hx + dh - dhf + ef^{'} - e))| \\
&= |a^2he(f^{'} - f) + a^2y(gef^{'} - hf) + abhxe(f^{'} - f) + ab(eyf^{'} - dhyf) + acx(gef^{'} - hf) + \\
&\quad ac(dhf - ef^{'} + ef - dhf^{'}) + acdgy(f^{'} - 1) + acy(1 - f) + bcx(dhf^{'} - ef) + bcdy(f - \\
&\quad f^{'}) + c^2x(f - df^{'}) + c^2d(f^{'} - f)| \\
&\leqslant max\{|a^2he(f^{'} - f)|, |a^2y(gef^{'} - hf)|, |abhxe(f^{'} - f)|, |ab(eyf^{'} - dhyf)|, |acx(gef^{'} - hf)|, \\
&\quad |ac(dhf - ef^{'} + ef - dhf^{'})|, |acdgy(f^{'} - 1)|, |acy(1 - f)|, |bcx(dhf^{'} - ef)|, |bcdy(f - f^{'})|, \\
&\quad |c^2x(f - df^{'})|, |c^2d(f^{'} - f)|\} \\
&\leqslant max\{|c^2x|, |bcx|, |acegx|, |a^2|, |abx|\}
\end{aligned}
$$

$$
\begin{aligned}
|av_2 - bv_1| &= |a(a(gef - hf^{'} + gyf^{'} - gyf) + b(egx - egxf + hxf^{'} - hx + dh - e + dgyf - \\
&\quad dgy + y - yf^{'}) + c(gxf - gxf^{'} + f^{'} - dgf)) - b(a(xeg - xh + ef - e + dh - \\
&\quad dhf^{'} + dgyf^{'} - dgy + y - yf) + b(dhxf^{'} - exf + dyf - dyf^{'}) + c(xf - \\
&\quad dgxf^{'} + df^{'} - df))| \\
&= |a^2(gef - hf^{'}) + a^2gy(f^{'} - f) + abx(egf - hf^{'}) + abdgy(f - 1) + abyf^{'} + \\
&\quad acgx(f - f^{'}) + ac(f^{'} - dgf) + ab(ef - dhf^{'}) + adgy(1 - f^{'}) + abyf + b^2x(ef - \\
&\quad dhf^{'}) + b^2dy(f^{'} - f) + bcx(dgf^{'} - f) + bcd(f - f^{'})| \\
&\leqslant max\{|a^2(gef - hf^{'})|, |a^2gy(f^{'} - f)|, |abx(egf - hf^{'})|, |abdgy(f - 1)|, |abyf^{'}|, \\
&\quad |acgx(f - f^{'})|, |ac(f^{'} - dgf)|, |ab(ef - dhf^{'})|, |adgy(1 - f^{'})|, |abyf|, |b^2x(ef - dhf^{'})|, \\
&\quad |b^2dy(f^{'} - f)|, |bcx(dgf^{'} - f)|, |bcd(f - f^{'})|\} \\
&\leqslant max\{|acx|, |a^2|, |abegx|, |bcx|, |b^2x|\}
\end{aligned}
$$

Hence,we have

$$\begin{aligned} d([a:b:c],A.[x:1:y]) &= d([a:b:c],[v_1:v_2:v_3]) \\ &= \frac{max\{|bv_3-cv_2|,|cv_1-av_3|,|av_2-bv_1|\}}{max\{|a|,|b|,|c|\}max\{|v_1|,|v_2|,|v_3|\}} \\ &\leqslant \frac{max\{|a^2|,|b^2x|,|c^2x|,|abegx|,|acegx|,|bcegx|,|abx|,|acx|,|bcx|\}}{max\{|a|,|b|,|c|\}max\{|aegx|,|begx|,|cegx|\}} \\ & (since,|abx|>|ab|,|acx|>|ac|) \end{aligned}$$

Case(i): If $|b|,|c| \leqslant |a|$ with the property $p^d < |a| < |eg| \leqslant p^{2d}, |b|,|c| \leqslant 1$ then

$$\begin{aligned} d([a:b:c],A.[x:1:y]) &= d([a:b:c],[v_1:v_2:v_3]) \\ &\leqslant \frac{max\{|a^2|,|b^2|,|c^2|,|abegx|,|acegx|,|bcegx|,|acx|,|abx|,|bcx|\}}{max\{|a|,|b|,|c|\}max\{|aegx|,|begx|,|cegx|\}} \\ &\leqslant \frac{max\{|a^2|,|abegx|,|acegx|,|acx|,|abx|\}}{|a^2egx|} \\ &< \frac{max\{|aeg|,|abegx|,|acegx|\}}{|a^2egx|} \\ & (\because |eg|>1 \Rightarrow |abegx|>|abx|,|acegx|>|acx|) \\ &< \frac{max\{|eg|,|begx|,|cegx|\}}{|aegx|} < \frac{max\{1,|bx|,|cx|\}}{|ax|} \\ &\leqslant \frac{max\{1,|x|\}}{|ax|} < \frac{1}{|ax|} or \frac{|x|}{|ax|} \\ & (since,|ax|>|a| then \frac{1}{|ax|} < \frac{1}{|a|}) \\ &< \frac{1}{|a|} or \frac{1}{|a|} \\ &< \frac{1}{p^d} \leqslant \frac{1}{p^{d+1}} \end{aligned}$$

Case(ii): If $|a|, |c| \leqslant |b|$ with the property $p^d < |b| < |eg| \leqslant p^{2d}, |a|, |c| \leqslant 1$ then

$$
\begin{aligned}
d([a:b:c], A.[x:1:y]) &= d([a:b:c], [v_1:v_2:v_3]) \\
&\leqslant \frac{max\{|a^2|, |b^2|, |c^2|, |abegx|, |acegx|, |bcegx|, |acx|, |abx|, |bcx|\}}{max\{|a|, |b|, |c|\} max\{|aegx|, |begx|, |cegx|\}} \\
&\leqslant \frac{max\{|b^2|, |abegx|, |bcegx|, |bcx|, |abx|\}}{|b^2egx|} \\
&< \frac{max\{|beg|, |abegx|, |bcegx|\}}{|b^2egx|} \\
&\quad (since, |eg| > 1 \Rightarrow |abegx| > |abx|, |bcegx| > |bcx|) \\
&< \frac{max\{|eg|, |aegx|, |cegx|\}}{|begx|} \\
&< \frac{max\{1, |ax|, |cx|\}}{|bx|} \\
&\leqslant \frac{max\{1, |x|\}}{|bx|} < \frac{1}{|bx|} or \frac{|x|}{|bx|} \\
&\quad (since, |bx| > |b| then \frac{1}{|bx|} < \frac{1}{|b|}) \\
&< \frac{1}{|b|} or \frac{1}{|b|} \\
&< \frac{1}{p^d} \leqslant \frac{1}{p^{d+1}}
\end{aligned}
$$

Case(iii): If $|a|, |b| \leqslant |c|$ with the property $p^d < |c| < |eg| \leqslant p^{2d}, |a|, |b| \leqslant 1$ then

$$
\begin{aligned}
d([a:b:c], A.[x:1:y]) &= d([a:b:c], [v_1:v_2:v_3]) \\
&\leqslant \frac{max\{|a^2|, |b^2|, |c^2|, |abegx|, |acegx|, |bcegx|, |acx|, |abx|, |bcx|\}}{max\{|a|, |b|, |c|\} max\{|aegx|, |begx|, |cegx|\}} \\
&\leqslant \frac{max\{|c^2|, |acegx|, |bcegx|, |bcx|, |acx|\}}{|c^2egx|} \\
&< \frac{max\{|ceg|, |acegx|, |bcegx|\}}{|c^2egx|} \\
&\quad (since, |eg| > 1 \Rightarrow |acegx| > |acx|, |bcegx| > |bcx|) \\
&< \frac{max\{|eg|, |aegx|, |begx|\}}{|cegx|} \\
&< \frac{max\{1, |ax|, |bx|\}}{|cx|} \leqslant \frac{max\{1, |x|\}}{|cx|} < \frac{1}{|cx|} or \frac{|x|}{|cx|} \\
&\quad (since, |cx| > |c| then \frac{1}{|cx|} < \frac{1}{|c|}) \\
&< \frac{1}{|c|} or \frac{1}{|c|} \\
&< \frac{1}{p^d} \leqslant \frac{1}{p^{d+1}}
\end{aligned}
$$

Hence in all cases with any one of the conditions in the hypothesis are satisfied, then, $d([a:b:c], A.[x:1:y]) \leqslant \frac{1}{p^{d+1}}$. Therefore,we get $A.[x:1:y] \in N_{[a:b:c]}$.

■

Corollary 5.3.4. *If $A \in PGL_3(F_p((x)))$ is taken from Theorem 5.2.1 then* $A^{-1}(\mathbb{P}^2 - N_{[a:b:c]}) \subset \{N_{[1:g:h]} \cup N_{[d:1:e]}\}$

Proof. Suppose,$A^{-1}(\mathbb{P}^2 - N_{[a:b:c]}) \nsubseteq \{N_{[1:g:h]} \cup N_{[d:1:e]}\}$.

Consider an element, $[t] \in \mathbb{P}^2 - \{N_{[a:b:c]}\}$,which says that
$A^{-1}.[t] \notin \{N_{[1:g:h]} \cup N_{[d:1:e]}\}$

Thus,

$$A^{-1}.[t] \in \mathbb{P}^2 - \{N_{[1:g:h]} \cup N_{[d:1:e]}\} \tag{5.2}$$

By lemma 5.3.1 to 5.3.3, the equation (5.2) will be changed as,
$A(A^{-1}.[t]) = [t] \in N_{[a:b:c]}$ which is a contradiction to $[t] \in \mathbb{P}^2 - \{N_{[a:b:c]}\}$.
Therefore our assumption is wrong. ■

Thus the Free generators theorem is proved and it provides a great choices of generators.It says that the pre-images of any pair of matrices in $PGL_3(F_p((x)))$ generates a free subgroup in $PGL_3(F_p((x)))$ and also it generates a free subgroup in $GL_3(F_p((x)))$.Its application in Cayley hash function is discussed in the next chapter.

Chapter 6

Hash function Using Free Generators Theorem

We are recollecting the facts that the pre-images of any pair of matrices generating a free subgroup in $PGL_3(F_p((x)))$ and it generates a free subgroup in $GL_3(F_p((x)))$.We desire to obtain such a pair of matrices over $M_{3\times 3}(F_p[x])$ that were free generators of the subgroups of $GL_3(F_p((x)))$ and define the projection mapping from an algebraic structure to quotient structure which gives the images in $GL_3(F_q)$.

By using Free Generators Theorem,we obtain the collection of pair of matrices with entries in $F_p[x]$ which is defined in set $\mathbb{D}$.We work with the elements in $\mathbb{D}$ to construct the Cayley Hash function and the security properties like protection of the local modifications of a text, of the corresponding hash function.We can create an infinite number of hash functions using the Free Generators Theorem by varying values of p and n and here we clearly says that for what choices of parameters these hash functions are the best.

6.1. Construction of the Hash Function

We require the following definitions to construct the Hash Function

Definition 6.1.1. Let p be a prime and we define the set as
$\mathbb{D} = \{(\tilde{A}, \tilde{B}) \ni \tilde{A}, \tilde{B} \in M_{3\times 3}(F_p((x)))$ are pre-images of A and B in $PGL_3(F_p((x)))\}$
That is, we define the set of all pair of matrices $(\tilde{A}, \tilde{B})$ such that $\tilde{A}, \tilde{B} \in GL_3(F_p[x])$ are pre-images of A and B in $PGL_3(F_p((x)))\}$ given by the theorem 5.2.1

Definition 6.1.2. Given a prime p and an irreducible polynomial $r_n(x)$ in $F_p[x]$ of degree n and $F_q \cong F_p[x]/ < r_n(x) >$.Let us define the projection function
$\pi_{r_n} : S \longrightarrow GL_3(F_q)$ be the mapping from the set
$S = \{M \in M_{3\times 3}(F_p[x]) \ni r_n(x) \nmid det(M)\}$ to the group $GL_3(F_q)$.
That is,the map yields the matrix entries to their projection in F_q.For ease of notation, we will write π_{r_n} as π , when $r_n(x)$ is not mentioned.

Remark. If $det(\tilde{A})$ and $det(\tilde{B})$ are not divisible by $r_n(x)$ then the construction of the hash function using the projection map from a choice of generators $(\tilde{A}, \tilde{B})$ in $\mathbb{D}$ to elements of $GL_3(F_q)$.

Formal definition of the hash function is defined as follows,

Definition 6.1.3. Let p be a prime and an irreducible polynomial $r_n(x)$ in $F_p[x]$ of degree n.Let us choose the matrices $(\tilde{A}, \tilde{B}) \in G$ such that $det(\tilde{A}) \nmid r_n(x)$ and $det(\tilde{B}) \nmid r_n(x)$, then the associated hash function $H_4 : \{0,1\}^* \longrightarrow GL_3(F_q)$ are as follows,if $m = m_1 m_2 .....m_n$ be a binary string. Then $H_4(m) = h(m_1)h(m_2).....h(m_n)$

$$\text{where, } h(m_i) = \begin{cases} h(\pi_{r_n}(\tilde{A})) & \text{if} m_i = 0 \\ h(\pi_{r_n}(\tilde{B})) & \text{if} m_i = 1 \end{cases}$$

We denote the set of all associated hash functions of $(\tilde{A}, \tilde{B}) \in \mathbb{D}$ and $r_n(x)$ as $\mathbb{H}$.
The hash function H_4, which defined here is resistant to the previous attacks on the Tillch-Zemor hash function and possesses some strong properties.

6.2. Properties of the Hash Function

Now first we scrutinize the properties of the elements in $\mathbb{H}$.To obtain the local modifications property,we need the degrees of the entries of $\tilde{A}$ and $\tilde{B}$ which are small when compared to n.

6.2.1 Small Modifications Property

The main goal of our hash function H_4 is to protect the small modifications property by using degree argument.
Suppose that $\tilde{A}$ and $\tilde{B}$ generates a free monoid in $M_{3\times3}(F_p[x])$, for any polynomial generators in $\mathbb{D}$.We need to prove the following results,
(i) part(a), which is related to the degree argument in the Tillich-Zemor's construction.
(ii) part(b) and part(c) gives that our matrices in $\mathbb{D}$ satisfy a strong property.

Proposition 6.2.1. *Let $(\tilde{A}, \tilde{B}) \in M_{3\times3}(F_p[x])$ such that $\tilde{A}, \tilde{B}$ generates the free monoid $M_{3\times3}(F_p[x])$ and let $r_n(x)$ be an irreducible polynomial in $F_p[x]$.Suppose H_4 be the associated hash function defined in 6.1.3 for $(\tilde{A}, \tilde{B})$ and $r_n(x)$.Assume that $\tau = max\{deg(\tilde{A}), deg(\tilde{B})\}$ and that $m \in \{0,1\}^l$ and $m' \in \{0,1\}^k$ are different bit strings for some $0 \leqslant l, k \leqslant n \mid \tau$. Then*

- $H_4(m) \neq H_4(m^{'})$
- *If* $(\tilde{A}, \tilde{B}) \in \mathbb{D}$ *then* $H_4(m) \neq a.H_4(m^{'})$ *for any* $a \in F_q$ *such that inspecting 'a' as an element of* $F_p[x]$, $(deg(a) + k\tau) < n$
- $H_4(m) \neq a.I$ *for any* $a \in F_q$.

Proof. Let M and $M^{'}$ be the product of $\tilde{A}$'s and $\tilde{B}$'s respectively produce $H_4(m)$ and $H_4(m^{'})$ in $M_{3\times3}(F_p[x])$,before projecting M and $M^{'}$ into $GL_3(F_q)$.So that $\pi(M) = H_4(m)$ and $\pi(M^{'}) = H_4(m^{'})$. Since $l, k < n \mid \tau$ then M has degree atmost $l\tau$ and $M^{'}$ has degree atmost $k\tau$. We know that, each of the entries of M and $M^{'}$ has a degree less than n then $\pi(M) = \pi(M^{'}) \in Gl_3(F_q)$ if and only if $M = M^{'} \in M_{3\times3}(F_p[x])$.

(i) Since m and $m^{'}$ are distinct messages then the products of $\tilde{A}$'s and $\tilde{B}$'s are also distinct. Clearly , $M, M^{'} \in (\tilde{A}, \tilde{B})$ and by our hypothesis $\tilde{A}$ and $\tilde{B}$ generates a free monoid. So,

$\Longrightarrow M = x_1x_2.....x_l \neq y_1y_2.....y_k = M^{'}$, where $x_i, y_j \in \{\tilde{A}, \tilde{B}\}$,

for all $i = 1$ to $l, j = 1$ to k.

$\Longrightarrow M \neq M^{'}$ in $M_{3\times3}(F_p[x])$

$\Longrightarrow \pi(M) \neq \pi(M^{'})$ in $GL_3(F_q)$

$\Longrightarrow H_4(m) \neq H_4(m^{'})$

(ii) Since, $\tilde{A}, \tilde{B}$ in $\mathbb{D}$ then their pre-images are in $PGL_3(F_p((x)))$ generates a free subgroup of $PGL_3(F_p((x)))$ which implies that $[M] \neq [M^{'}]$

$\Longrightarrow M \neq aM^{'}$ for any $a \in F_p[x]$

Suppose,$\pi(M) = a\pi(M^{'})$ for some $a \in F_q$ in $GL_3(F_q)$, inspecting 'a' as an element in $F_p[x]$, then $M = aM^{'} + r_n(x)M^{''}$ for some $M^{''} \in M_{3\times3}(F_p[x])$

By hypothesis, $deg(a) + deg(M^{'}) < n \Longrightarrow deg(aM^{'}) < n$ and M has all the entries, each of degree less than n, which says that $M^{"} = 0$

Therefore, $M = aM^{'}$, which is a contradiction.

Hence $\pi(M) \neq a\pi(M^{'})$.Thus, $H_4(m) \neq aH_4(m^{'})$ for any $a \in F_q$.

(iii) It is not possible for the products in $\pi(M)$ and $\pi(M^{'})$ of length less than $n \mid \tau$ to get the identity.

Therefore, $H_4(m) \neq aI$ for any $a \in F_q$

Hence, $\pi(M)$ and $\pi(M^{'})$ must have order atleast $n \mid \tau$ ■

6.3. Resistance against different attacks

(i) To avert the collisions of the form $\pi(\tilde{A})^{ord(\pi(\tilde{A}))} = I$ and $\pi(\tilde{B})^{ord(\pi(\tilde{B}))} = I$,we consider that $\pi(\tilde{A})$ and $\pi(\tilde{B})$ must be of large order.

(ii) If $det(\pi(\tilde{A}))$ is primitive root then $order(\pi(\tilde{A})) \geqslant q - 1$

However, avoiding short relations attack is of the form $W(\pi(\tilde{A}), \pi(\tilde{B})) = kI$, where $k \in F_q^*$ and $W(\pi(\tilde{A}), \pi(\tilde{B})) \in \{\pi(\tilde{A}), \pi(\tilde{A}^{-1}), \pi(\tilde{B}), \pi(\tilde{B}^{-1})\}$ be a non-trivial word.To avoid such relations we proved the proposition.

Proposition 6.3.1. *Let $(\tilde{A}, \tilde{B}) \in \mathbb{D}$ and let $W(\tilde{A}, \tilde{B}) \in \{\tilde{A}, \tilde{A}^{-1}, \tilde{B}, \tilde{B}^{-1}\}$ be a non-trivial word then there exist a choice of $r_n(x)$ such that if $F_q = F_p[x] \mid < r_n(x) >$,then $W(\pi_{r_n}(\tilde{A}), \pi_{r_n}(\tilde{B})) \neq kI \in GL_3(F_q)$ for any $k \in F_q^*$*

Proof. Let $\phi = det(\tilde{A}\tilde{B}) = det(\tilde{A})det(\tilde{B}) \in F_p[x]$

We define,the localization of $F_p[x]$ at ϕ as $F_p[x]_{1/\phi} = \{\frac{t}{\phi^m} : t \in F_p[x], m \geqslant 0\}$

Since,$\frac{1}{\phi} = \frac{1}{det(\tilde{A}\tilde{B})} = \left(\frac{1}{det(\tilde{A})}\right)\left(\frac{1}{det(\tilde{B})}\right)$

we have, $\frac{1}{det(\tilde{A})} \in F_p[x]_{1/\phi}$ and $\frac{1}{det(\tilde{B})} \in F_p[x]_{1/\phi}$

We know that,$F_p[x]_{1/\phi}$ is contained in the fraction field of $F_p[x]$ and also in $F_p((x))$.

Therefore,$\tilde{A}, \tilde{B} \in GL_3(F_p[x])_{1/\phi} \subset GL_3(F_p((x)))$

We define the ideal $< r >= \{rt \ni t \in F_p[x]\}$ for any irreducible polynomial $r \in F_p[x]$.

Assume,$r \nmid \phi$ and let $r_{1/\phi}$ be the localization of r at ϕ.

(i.e) $r_{1/\phi} = \{\frac{rt}{\phi^m} \ni t \in F_p[x], m \geqslant 0\}$

Now,let us consider a surjective homomorphism from $F_p[x]_{1/\phi}$ to F_q under the irreducible polynomial.

(i.e) $\eta_r : F_p[x]_{1/\phi} \longrightarrow F_q$ induced by $x \longrightarrow a$, where a is a root of r and has a kernel $r_{1/\phi}$.

Thus,by the first isomorphism theorem gives as,$\frac{F_p[x]_{1/\phi}}{r_{1/\phi}} \cong F_q$.

Under this homomorphism, the natural images of $\tilde{A}$ and $\tilde{B}$ in $GL_3(F_q)$ is $\pi_r(\tilde{A})$ and $\pi_r(\tilde{B})$ respectively.

Since $\tilde{A}, \tilde{B} \in \mathbb{D}$, then their images $\tilde{A}, \tilde{B} \in PGL_3(F_p((x)))$ also generate a free subgroup,this says that no freely reduced (non-trivial) word in $\{\tilde{A}, \tilde{A}^{-1}, \tilde{B}, \tilde{B}^{-1}\}$ can be I or any scalar multiple kI of I for any $k \in F_q^*$.

But we observe that images of $\tilde{A}, \tilde{B}$ in $GL_3(F_q)$ under homomorphism may have $W(\pi_r(\tilde{A}), \pi_r(\tilde{B})) = kI$ for some $k \in F_q^*$ if and only if inspecting $k \in F_p[x] \subset F_p[x]_{1/\phi}$

(i.e) if W=$\begin{pmatrix} k+\alpha & \beta & \gamma \\ \delta & k+\mu & \lambda \\ \theta & \omega & k+\nu \end{pmatrix}$, where $\alpha, \beta, \delta, \gamma, \mu, \lambda, \theta, \omega, \nu \in F_p[x]_{1/\phi}$

and $\eta_r(\alpha) = \eta_r(\beta) = \eta_r(\delta) = \eta_r(\gamma) = \eta_r(\mu) = \eta_r(\lambda) = \eta_r(\theta) = \eta_r(\omega) = \eta_r(\nu) = 0$

Hence,we choose $r_n(x) \in F_p[x]$ such that any one of the $\eta_{r_n}(i)$,

$i \in \{\alpha, \beta, \delta, \gamma, \mu, \lambda, \theta, \omega, \nu\}$ must be non-zero.

We need to choose $r_n(x)$ very carefully to avert from a small set of relations. ■

6.4. Consequences of the determinant

We find unexpectedly that the leakage of the knowledge about the original bit strings by choosing f and $f^{'}$ will leads to the potential issues.We suggested the padding method to avoid these potential issues.

Suppose that $\pi(\tilde{A}), \pi(\tilde{B}) \in GL_3(F_q)$ such that $det(\pi(\tilde{A})) = \Omega$ and $det(\pi(\tilde{B})) = \omega$. If $m \in \{0,1\}^l$ be the message of length l and $M = H(m) \in GL_3(F_q)$ be the hashed values of message m,then we assume that $l = l_1 + l_2$ where l_1-number of zeros in m and l_2-number of ones in m.

6.4.1 Attacks through determinant

The leakage of any information by the determinant will leads to (our construction) some potential issues.We mean the notation $dlog_x(y)$ such that $x^k = y$ for $x, y \in F_q^*$ and $k \in \{1, 2, ...q-1\}$

(i) Finding length of the message l is efficient,when $\Omega = \omega$.

If $\Omega = \omega$ and also we find out the relation $dlog_\Omega(det(M)) \equiv l(mod(p^n - 1))$, where l is the length of the message.

$\Longrightarrow \Omega^l = det(M)$,which says that $l << p^n - 1$ (since $p^n - 1$ is much larger than any

message length).

So,that there is a possibility that the hackers can simply compute the discrete log of Ω to determine the message length l.

(ii)Leaking knowledge about l_1 or l_2 is efficient when $gcd(ord(\Omega), ord(\omega))$ is near to $ord(\Omega)$ or $ord(\omega)$.

We know that l_1 is the number of zeros in m,l_2 is the number of ones in m and also $l_1 + l_2 = l$,length of the message.

Assume that $ord(\Omega) = s$ and $ord(\omega) = t$ with $gcd(s,t) = k$

we know that, s and t must divide the order of the group F_q^*. (i.e) $ord(F_q^*) = p^n - 1$

Since,$det(M) = \Omega^{l_1}\omega^{l_2}$

$\implies det(M^s) = \Omega^{sl_1}\omega^{sl_2}$ (since, $\Omega^s = 1$)

$\implies det(M^s) = \omega^{sl_2}$

Thus, $det(M^s) = 1$ if and only if $t \mid sl_2$ which is true $\frac{t}{k} \mid l_2$

Hence a hacker can determine $det(M^s)$ and able to conclude that the l_2-number of ones in m is divisible by $\frac{t}{k}$ if $det(M^s) = 1$

(i.e) if $det(M^s) = 1$ then $\frac{t}{k} \mid l_2$ and if $det(M^s) \neq 1$ then $\frac{t}{k} \nmid l_2$

This attack would be difficult if $gcd(s,t)$ is large in comparison to s and t.

(i.e) $gcd(s,t) > s$ or $gcd(s,t) > t$ and $gcd(s,t) \neq s$ or t

For example,if ω is a primitive root.So $ord(\omega) = t = p^n - 1$ and also

$ord(\Omega) = s = \frac{p^n - 1}{2}$, then the hackers can arrive at a conclusion that the fact of being odd or even of l_2-number of ones in m,by finding out whether $det(M^s)$ is 1 or

not.

We observe that the strategy of this attack does not give any information if Ω and ω are primitive roots.

(iii) If Ω is a primitive root and $\omega = \Omega^r$ for some r such that $l_1 < r$ and $l_1 + rl_2 < p^n - 1$ then

(a) find out l_1 and l_2

(b) find out the parity of l or l_1

(a) We know that $det(M) = \Omega^k$ for some k

$\Longrightarrow det(M) = \Omega^k = \Omega^{l_1}\Omega^{rl_2}$

$\Longrightarrow k \equiv l_1 + rl_2(mod(p^n - 1))$

Since $l_1 + rl_2 < p^n - 1$ then $k = l_1 + rl_2$ and since $l_1 < r$ then $\frac{k}{r} = \frac{l_1}{r} + l_2$ where l_2 is an integer and $\frac{l_1}{r} < 1$

If a hacker can able to find out these $dlog_\Omega(det(M)) = k(mod(p^n - 1))$ and $dlog_\Omega(det(\omega)) = r(mod(p^n - 1))$

(i.e) he can efficiently able to determine $dlog_\Omega$, then he can get the k and r values.Using these values he can find out the $\lfloor\frac{k}{r}\rfloor$ to get l_2.

From these l_2 and r and also using the relation $\frac{l_1}{r} + l_2 = \frac{k}{r}$,he can easily compute l_1 also.

Thus,he would know l_1-number of zeros and l_2-number of ones in m.

(b) Since Ω is a primitive root, we know that $\omega = \Omega^r$ for some r and also we know that $det(M) = \Omega^k$ where $k = l_1 + rl_2$.(by part(a))

since $l = l_1 + l_2 \Longrightarrow l_2 = l - l_1$

$\therefore k = l_1 + r(l - l_1) = rl + (1 - r)l_1(mod(p^n - 1))$.

Hence the hackers can efficiently find out $dlog_\Omega$ to get the k and r values.We choose p is odd then $p^n - 1$ is even.Therefore,if k is even and r is even then l_1 is even and if k is odd and r is odd then l_1 is odd.

Similarly, if k is even and r is odd then l is even, and if k is odd and r is odd then l is odd

Thus,it is efficient to find out the parity of l or l_1.

Again, we notice that the above attack part(a) and part(b) can be protected by choosing Ω be a primitive root and $\omega = \Omega^r$, for some r such that $gcd(r, p^n - 1) = 1$ then ω is also a primitive root.

We suggested earlier that both Ω and ω be a primitive root then these above attacks are also avoided and also these attacks depends on being able to compute discrete logs in F_q.

6.4.2 Weakness in the Distribution of the determinant

Assume that Ω and ω are primitive roots and $\omega = \Omega^r$ for some $r > 1$ and

$gcd(r, p^n - 1) = 1$.

Let m be a bit strings of length l and $det(H(m)) = \Omega^k$ where $k = l_1 + rl_2(mod(p^n - 1))$ for some $l_1, l_2 \in \mathbb{N}_0$ and $l = l_1 + l_2$

Now,fix l then $ord(\{l_1, l_2 \in \mathbb{N}_0 \text{ and } l = l_1 + l_2\}) = l + 1$.

Hence the possible values of the determinants of the hashed values of bit strings of fixed length l is $l + 1$.

Therefore, we know that the messages of length l of the hashed values are not distributed uniformly between all possible determinants, since their determinants of the

hashed values must lie on a subset of $l+1$ possible values of $GL_3(F_q)$.

6.4.3 Padding

There is a common possibility in Cryptography to averting attacks like the weakness in the determinant as mentioned above.That weakness is cleared by padding messages with some bits to vague the original determinant of the hashed value.

For example,choose one of the standard padding scheme (PKCS-5) which multiple the message block length and added the amount of padding as required [48].

We suggested our padding as follows:

Let the elements are taken from $\mathbb{D}$ and let l be the length of the message.Now we pad our length l messages to length $2l$ bit strings with l-ones and l-zeros.Then the determinant values M of message m is $det(M) = \Omega^k = \Omega^{l(1+r)}$.Hence all the output of the determinant values are $\Omega^{l(1+r)}$.

Thus, the padding will completely remove the weakness of the determinant on security and also the distribution problem in determinant.

If choose $\pi(\tilde{A}), \pi(\tilde{B}) \in \mathbb{D}$ generates all of $GL_3(F_q)$, then the number of hash values will be $ord(\{M \in GL_3(F_q) \ni det(M) = \Omega^{l(1+r)}\}) = q^3(q^3-1)(q^2-1)$.

If $\Omega = \omega$, then any type of padding to a fixed length would remove both the weakness on security and also the distribution issues.

6.5. Condition for opting generators

Let $D = (\tilde{A}, \tilde{B}) \in \mathbb{D}$,$det(\pi(\tilde{A})) = \Omega$ and $det(\pi(\tilde{B})) = \omega$ and $\tau = max\{deg(\tilde{A}), deg(\tilde{B})\}$, then if opting $\tilde{A}, \tilde{B}$ then the following conditions are to be satisfied,

(i) n/τ is chosen to be large.

(ii) Ω, ω are primitive roots.

(iii) n is prime and either $f, f^{'}$ or $\tilde{f}, \tilde{f}^{'}$ are primitive roots of F_q.

(iv) Ω or ω is not square in F_q.

Here, condition(iv) is satisfied, if the Ω, ω are primitive roots and prime p is odd.
Suppose,prime p is odd and $\Omega = s^2$ for some $s \in F_q$.(i.e) Ω is a square in F_q.
Since $s \in F_q^*$ then $s^{q-1} = 1$.

Therefore,$\Omega^{\left(\frac{q-1}{2}\right)} = (s^2)^{\left(\frac{q-1}{2}\right)} = 1$.So $ord(\Omega) \mid \left(\frac{q-1}{2}\right)$ which implies that Ω is not the primitive root,which is a contradiction. Hence, Ω or ω is not square in F_q.

Finding a primitive root of small degree is not easy for a general case of finite field and noted that the maximum degree of such a primitive root is indeed bounded [74].For definite possibility of $r_n(x)$ will make this part easier.There are many software packages will help us to select the $r_n(x)$ as a Conway polynomial for $r_n(x)$ which assurance that the x is the primitive root.

6.6. Security on known attacks

The attacks on the Tillich-Zemor hash function are considered to verify the hash function in H_4.

(i) The attack which depends on $r_n(x)$ such that $\tilde{A}$ or $\tilde{B}$ had small order.Thus,the

probability of randomly chosen $r_n(x)$ such that $\tilde{A}$ and $\tilde{B}$ with large order is really high for Tillich and Zemor's generators [1].

Here in our case the elements are taken from $\mathbb{D}$,we know that by choosing $det(\pi(\tilde{A}))$ and $det(\pi(\tilde{B}))$ to be primitive roots, will ensure that $\pi(\tilde{A})$ and $\pi(\tilde{B})$ have order atleast $(q-1)$.

(ii) In [7], says, to prove that the small order attack presented by steinwandt et.al is also extendable to larger characteristics.The attack says that the $r_n(x)$ is analysable,in the sense that the polynomial is composed into the two non-trivial polynomials.
However, it proves that in F_2, the probability that the randomly chosen irreducible polynomial of degree n is analysed tends to 0 as n tends to infinity.These methods also extend to irreducible polynomials over F_p [8].
Hence, steinwandt et, al. noted that the small order attack could be avoided by simply choosing n to be a prime [69].
Here in our case, choosing n to be prime will exclude from the small order attack.

(iii) In [61] Petit, Lauter and Quisquater referred about that the attack used on the Morgestern hash function could possibly be used for the Zemor-Tillich hash function and similar construction by embedding the associated Cayley graphs into Morgestern graphs.But no other details of this attacks were given and this construction is not analysed further.So, this type of attack will not explored.

(iv) Geiselmann's Embedding attack was independent of the choice of irreducible polynomial $r_n(x)$ and the alternative description of this attack was discussed in [7].This attacks computing time depends on the computation time of computing the discrete

logs in $GL_3(F_{2^n})$. We know that in [4] , the discrete log problem in $GL_3(F_{p^n})$ can be reduced to the problem of computing discrete log in $F(_{p^n})^3$.

By applying Coppersmith's algorithm [31] the computation of the discrete logs in F_{p^n} when p=2 is made faster and also this algorithm applies in the case that p is fixed and n tends to infinity, and runs in subexponential time [30]. Now, Coppersmith's algorithm is the fastest known algorithm for computing discrete logs in F_{p^n}, and in $F(_{p^n})^3$ where n is prime and p is not close to n in size [2].
Generally, the computation of the discrete logs in F_{p^n} with odd characteristics is examine to be more difficult [6]. Hence this attack is efficient but not practical. Because it produces the collisions only for the long strings of zeros and ones, which is not helpful in practice.

Hence all the attacks which we discussed here are the standard attacks.These attacks have no advantage and may be even more difficult for our hash function H_4. All the attacks we discussed are runs in exponential time,choosing parameters comparable to those in cryptographic standards is expected to be sufficient to provide a secure hash function [38]. Now the NIST approved hash algorithms are SHA3-224,SHA3-256,SHA3-384,SHA3-512 [57] and these algorithms have their security strength in bits related to the numbers. For example, SHA3-512 gives 512 bits of security against a preimage or second preimage finding algorithm and 256 security bits against a collision finding algorithm.But,Mullans attack in [58] had a running time $\mathcal{O}(\sqrt{q})$ and produce collisions of length $\mathcal{O}((logq)^2/log(logq))$, which is the fastest known attack. If we choose $p^n \sim 2^{512}$ will provide equivalent security as a SHA3 family.

By choosing some suitable and satisfiable conditions to our hash functions H_4.They are secure against all the previous attacks on the Zemor-Tillich hash function and also by using padding method the potential weakness in our hash functions H_4 are removed.Thus, for a suitable choices of parameters, our hash function H_4 are collision resistant,pre-image resistant and second pre-image resistant.

6.7. Illustrations for prime p=3

For an odd prime, p=3 we consider how the polynomial generators $\tilde{A}, \tilde{B} \in \mathbb{D}$ for some simple choices of eigenvectors will look like and we note that our examples are too small to apply the hash function H_4 along with the padding using the python code (x,y).

6.7.1 Examples for p=3

For an odd prime, p=3 we consider how the polynomial generators $\tilde{A}, \tilde{B} \in \mathbb{D}$ look like for some simple choices of eigenvectors $[a : b : c], [1 : b : c], [1 : g : h]$, $[1 : \tilde{b} : \tilde{c}]$, $[1 : \tilde{g} : \tilde{h}], [\tilde{d} : 1 : \tilde{e}]$. We fix $[a : b : c] = [0 : 0 : 1]$, and choose $[1 : b : c], [1 : g : h], [1 : \tilde{b} : \tilde{c}], [1 : \tilde{g} : \tilde{h}], [\tilde{d} : 1 : \tilde{e}]$ as $[1 : 0 : 0], [0 : 1 : 0], [1 : 1 : 1]$, $[1 : 1 : 2], [1 : 2 : 1]$ in some order.

These possible choices of $[1 : b : c], [1 : g : h], [1 : \tilde{b} : \tilde{c}], [1 : \tilde{g} : \tilde{h}], [\tilde{d} : 1 : \tilde{e}]$ gives 6 different pairs of $(\tilde{A}, \tilde{B})$ which are shown in table 6.1. We name these choices of $(\tilde{A}, \tilde{B})$ as $G_i(f, f', \tilde{f}, \tilde{f}')$. Here, for our convenience, we denote $G_i(f, f', \tilde{f}, \tilde{f}')$ as $G_i(f, \tilde{f})$.

We know that the elements $[1 : 0 : 0], [0 : 1 : 0], [1 : 1 : 1], [1 : 1 : 2], [1 : 2 : 1] \in \mathbb{P}^2$ which are all distance 1 apart. So these values satisfies the condition (i) of theorem

5.2.1.

If we choose $d = 0$ then for any $f, f^{'}, \tilde{f}, \tilde{f}^{'} \in F_p[x]$ such that $f, f^{'}, \tilde{f}, \tilde{f}^{'}$ have a non-zero constant term which satisfy the condition (ii) of theorem 5.2.1.

For prime $p \geqslant 2$, the condition (iii) of theorem 5.2.1 is satisfied by the proposition 5.2.2.

Hence these choices for prime p=3, satisfies all the conditions of the theorem 5.2.1 and thus any pair of matrices $(\tilde{A}, \tilde{B})$ in table 6.1 generates a free subgroup in $GL_3(F_p((x)))$ for a prime p=3.

Let $p^n = p^3$ and denote the image of x under the projection mapping as α.With the help of the Conway polynomial algorithm, we choose $r_n(x) = x^3 - x + 1$, which guarantees that the image of x in F_q (i.e) α is both the root of $r_n(x)$ and also the primitive root of F_q.We know that $F_p[x]/ < r_n(x) > \cong F_{p^n}[\alpha]$.

Using the table 6.1 in the case p=3, $G_6(\tilde{A}, \tilde{B})$ then their determinant is $det(\tilde{A}) = 2ff^{'}$ and $det(\tilde{B}) = 2\tilde{f}\tilde{f}^{'}$.Now we have the choice of generators of $G_6(f, \tilde{f})$ were in table 6.1,

Table 6.1: $G_i(\tilde{A}), G_i(\tilde{B})$ for simple choices of eigenvectors with d=0

$G_i(\tilde{A}), G_i(\tilde{B})$	$\tilde{A}$	$\tilde{B}$	[a:b:c]	[1:b:c]	[1:g:h]	[1:$\tilde{b}$,$\tilde{c}$]	[1:$\tilde{g}$,$\tilde{h}$]	[$\tilde{d}$,1,$\tilde{e}$]
$G_1(\tilde{A}, \tilde{B})$	$\begin{pmatrix} f & 0 & 0 \\ 0 & f' & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} \tilde{f}+\tilde{f'} & \tilde{f}+2 & 0 \\ \tilde{f}+2\tilde{f'} & 2\tilde{f}+2 & \tilde{f'}+2 \\ \tilde{f'}+2\tilde{f} & \tilde{f}+2 & 2\tilde{f'}+2 \end{pmatrix}$	[0:0:1]	[1:0:0]	[0:1:0]	[1:1:1]	[1:2:1]	[1:1:2]
$G_2(\tilde{A}, \tilde{B})$	$\begin{pmatrix} f & 0 & 0 \\ 0 & f' & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 1+\tilde{f'} & \tilde{f}+2 & \tilde{f}+2\tilde{f'} \\ \tilde{f'}+2 & 1+\tilde{f} & \tilde{f}+2\tilde{f'} \\ 1+2\tilde{f'} & 2+\tilde{f} & \tilde{f}+\tilde{f'} \end{pmatrix}$	[0:0:1]	[1:0:0]	[0:1:0]	[1:2:1]	[1:1:1]	[1:1:2]
$G_3(\tilde{A}, \tilde{B})$	$\begin{pmatrix} f & 0 & 0 \\ 0 & f' & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 2\tilde{f}+2 & 1+2\tilde{f'} & \tilde{f}+2\tilde{f'} \\ 1+2\tilde{f} & 2+2\tilde{f'} & \tilde{f}+2\tilde{f'} \\ \tilde{f}+2 & 1+2\tilde{f'} & 2\tilde{f}+2\tilde{f'} \end{pmatrix}$	[0:0:1]	[1:0:0]	[0:1:0]	[1:2:1]	[1:1:2]	[1:1:1]
$G_4(\tilde{A}, \tilde{B})$	$\begin{pmatrix} f & 0 & 0 \\ 0 & f' & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 1+\tilde{f} & \tilde{f'}+2\tilde{f} & \tilde{f'}+2 \\ 1+2\tilde{f} & \tilde{f'}+\tilde{f} & \tilde{f'}+2 \\ \tilde{f}+2 & \tilde{f'}+2\tilde{f} & \tilde{f'} \end{pmatrix}$	[0:0:1]	[1:0:0]	[0:1:0]	[1:1:2]	[1:2:1]	[1:1:1]
$G_5(\tilde{A}, \tilde{B})$	$\begin{pmatrix} f+2f' & f'+f & 0 \\ 2f+2f' & f'+2f & 0 \\ f+f'+1 & f'+f & 2 \end{pmatrix}$	$\begin{pmatrix} 2 & 0 & 1+2\tilde{f'} \\ 0 & 2\tilde{f'} & \tilde{f'}+2\tilde{f} \\ 0 & 0 & 2\tilde{f} \end{pmatrix}$	[0:0:1]	[1:2:1]	[1:1:1]	[1:0:0]	[1:1:2]	[0:1:0]
$G_6(\tilde{A}, \tilde{B})$	$\begin{pmatrix} f+2f' & f'+f & 0 \\ 2f+2f' & f'+2f & 0 \\ f+f'+1 & f'+f & 2 \end{pmatrix}$	$\begin{pmatrix} 2\tilde{f} & 0 & 1-\tilde{f} \\ 0 & 2\tilde{f'} & 1-\tilde{f'} \\ 0 & 0 & 1 \end{pmatrix}$	[0:0:1]	[1:2:1]	[1:1:1]	[1:1:2]	[1:0:0]	[0:1:0]

We know that $-x^2 = x^{15}$, $-x^4 = -x^2 + x = x^{17}$, $-x^{10} = -x^2 - x = x^{23}$. Since $p^n - 1 = 26$ then $det(\pi(\tilde{A}))$ and $det(\pi(\tilde{B}))$ are distinct primitive roots.

Now using padding in the input binary strings, we execute the hash function H_4.The simple padding method is as follows, Let $m = m_1....m_n$ be a binary strings of length n.We define the padding of m bit strings as $m_1.....m_n\tilde{m}_1.....\tilde{m}_n$ where $\tilde{m}_i = 1$ if $m_i = 0$ and $\tilde{m}_i = 0$ if $m_i = 1$ for all $1 \leqslant i \leqslant n$.This padding method will remove the weakness in the determinant and exist same the determinant.

Table 6.2: Possible choices of $f, f', \tilde{f}, \tilde{f}'$

f	f'	$\tilde{f}$	$\tilde{f}'$	$det(\tilde{A})$	$det(\tilde{B})$
x	x	x^2	x^2	$2x^{15}$	x^{17}
x^2	x^2	x	x	$2x^{17}$	x^{15}
$2x$	x	x^2	x^2	x^{15}	x^{17}
x^2	x^2	$2x$	x	x^{17}	$2x^{15}$
$2x^2$	$2x^2$	x	x	$2x^{17}$	$2x^{15}$
$2x^2$	x^2	x	$2x$	x^{17}	$2x^{15}$
x	x	$2x^2$	$2x^2$	$2x^{15}$	x^{17}
x	$2x$	$2x^2$	x^2	x^{15}	$2x^{15}$
x	x	x^9	x	$2x^{15}$	x^{23}
x^9	x	x	x	$2x^{23}$	$2x^{15}$

Here along with the above parameters, we executed the hashed value of H_4 for an empty string as a test vector. The tests were executed in Intel(R) Core (TM) i3 2.00GHZ computer with 4GB of RAM using python (x,y) code.

For an empty string " ", we get the hashed value in H_4 as $\begin{pmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} \end{pmatrix}$

where,$\alpha_{11} = 4\alpha^{49} + 23\alpha^{48} + 24\alpha^{47} + 20\alpha^{46} + 10\alpha^{44} + 10\alpha^{43} + 23\alpha^{42} + 2\alpha^{41} + 9\alpha^{40} + 17\alpha^{39} + 6\alpha^{38} + 13\alpha^{37} + 2\alpha^{36} + 17\alpha^{35} + 18\alpha^{34} + 22\alpha^{33} + 16\alpha^{32} + 19\alpha^{31} + 16\alpha^{30} + 7\alpha^{29} + 19\alpha^{28} + 16\alpha^{27} + 26\alpha^{26} + 25\alpha^{25} + 3\alpha^{24} + 19\alpha^{23} + 8\alpha^{22} + 15\alpha^{21} + 14\alpha^{20} + 11\alpha^{19} + 17\alpha^{18} + 3\alpha^{17} + 18\alpha^{16} + 15\alpha^{15} + 5\alpha^{14} + 16\alpha^{13} + 21\alpha^{12} + 24\alpha^{9} + 21\alpha^{8} + 20\alpha^{7} + 18\alpha^{6} + 17\alpha^{5} + 17\alpha^{4},$

$\alpha_{12} = 14\alpha^{49} + 4\alpha^{48} + 4\alpha^{47} + 10\alpha^{46} + 12\alpha^{45} + 14\alpha^{44} + 4\alpha^{43} + 25\alpha^{42} + 3\alpha^{41} + 2\alpha^{40} + 5\alpha^{39} + 1\alpha^{38} + 20\alpha^{37} + 25\alpha^{36} + 23\alpha^{34} + 7\alpha^{33} + 2\alpha^{32} + 6\alpha^{31} + 16\alpha^{30} + 21\alpha^{29} + 21\alpha^{28} + 19\alpha^{26} + 13\alpha^{25} + 21\alpha^{24} + 8\alpha^{23} + 26\alpha^{22} + 6\alpha^{21} + 21\alpha^{20} + 19\alpha^{18} + 15\alpha^{17} + 20\alpha^{16} + 24\alpha^{15} + 15\alpha^{14} + 6\alpha^{13} + 23\alpha^{12} + 4\alpha^{11} + 21\alpha^{10} + 21\alpha^{8} + 23\alpha^{7} + 20\alpha^{6} + 7\alpha^{5}$

$\alpha_{13} = 18\alpha^{49} + 10\alpha^{47} + 10\alpha^{46} + 3\alpha^{45} + 18\alpha^{44} + 16\alpha^{43} + 20\alpha^{42} + 20\alpha^{41} + 16\alpha^{40} + 13\alpha^{39} + 8\alpha^{38} + 21\alpha^{37} + 10\alpha^{36} + 5\alpha^{35} + 5\alpha^{34} + 23\alpha^{33} + 12\alpha^{32} + 9\alpha^{31} + 3\alpha^{30} + 7\alpha^{28} + 9\alpha^{27} + 17\alpha^{26} + 9\alpha^{25} + 9\alpha^{24} + 7\alpha^{23} + 20\alpha^{22} + 3\alpha^{21} + 24\alpha^{20} + 18\alpha^{19} + 10\alpha^{18} + 23\alpha^{17} + \alpha^{16} + 17\alpha^{15} + 25\alpha^{14} + 9\alpha^{13} + 3\alpha^{12} + 3\alpha^{11} + 8\alpha^{10} + 26\alpha^{9} + 24\alpha^{8} + 22\alpha^{7} + 5\alpha^{6} + 26\alpha^{5} + 5\alpha^{3} + 5\alpha^{2} + 2\alpha^{2}$

$\alpha_{21} = 19\alpha^{49} + 11\alpha^{48} + 3\alpha^{47} + 9\alpha^{46} + 15\alpha^{45} + 26\alpha^{44} + 13\alpha^{43} + 6\alpha^{42} + 20\alpha^{41} + 7\alpha^{40} + 20\alpha^{39} + 20\alpha^{38} + 14\alpha^{37} + 4\alpha^{36} + 22\alpha^{35} + 2\alpha^{34} + 24\alpha^{33} + 1\alpha^{32} + 13\alpha^{31} + 22\alpha^{30} + 15\alpha^{29} + 24\alpha^{28} + 8\alpha^{27} + 12\alpha^{26} + 20\alpha^{25} + \alpha^{24} + 20\alpha^{23} + 3\alpha^{22} +$

$$14\alpha^{21}+8\alpha^{20}+25\alpha^{19}+21\alpha^{18}+19\alpha^{17}+9\alpha^{16}+18\alpha^{15}+19\alpha^{14}+11\alpha^{13}+14\alpha^{11}+12\alpha^{10}+2\alpha^{9}+12\alpha^{8}+5\alpha^{7}+18\alpha^{6}+11\alpha^{5}+11\alpha^{4}$$

$$\alpha_{22} = 18\alpha^{49} + 1\alpha^{48} + 2\alpha^{47} + 22\alpha^{45} + 14\alpha^{44} + 26\alpha^{44} + 13\alpha^{43} + 6\alpha^{42} + 20\alpha^{41} + 7\alpha^{40} + 20\alpha^{39} + 20\alpha^{38} + 14\alpha^{37} + 4\alpha^{36} + 22\alpha^{35} + 2\alpha^{34} + 24\alpha^{33} + 1\alpha^{32} + 13\alpha^{31} + 22\alpha^{30} + 15\alpha^{29} + 24\alpha^{28} + 8\alpha^{27} + 12\alpha^{26} + 20\alpha^{25} + \alpha^{24} + 20\alpha^{23} + 3\alpha^{22} + 14\alpha^{21}+8\alpha^{20}+25\alpha^{19}+21\alpha^{18}+19\alpha^{17}+9\alpha^{16}+18\alpha^{15}+19\alpha^{14}+11\alpha^{13}+14\alpha^{11}+12\alpha^{10}+2\alpha^{9}+12\alpha^{8}+5\alpha^{7}+18\alpha^{6}+11\alpha^{5}+11\alpha^{4}$$

$$\alpha_{23} = 10\alpha^{49} + 12\alpha^{48} + 10\alpha^{47} + 17\alpha^{46} + 10\alpha^{45} + 11\alpha^{44} + 11\alpha^{43} + 3\alpha^{42} + 23\alpha^{41} + 21\alpha^{40} + 20\alpha^{38} + 23\alpha^{37} + \alpha^{36} + 9\alpha^{35} + 24\alpha^{34} + 23\alpha^{33} + 20\alpha^{32} + 11\alpha^{31} + 19\alpha^{30} + 14\alpha^{29} + 12\alpha^{28} + 8\alpha^{27} + 25\alpha^{26} + 24\alpha^{25} + 11\alpha^{24} + 16\alpha^{23} + 21\alpha^{22} + 14\alpha^{21}+23\alpha^{20}+15\alpha^{19}+1\alpha^{18}+15\alpha^{17}+22\alpha^{16}+5\alpha^{15}+15\alpha^{13}+1\alpha^{12}+13\alpha^{11}+18\alpha^{9}+21\alpha^{8}+9\alpha^{7}+8\alpha^{6}+20\alpha^{5}+8\alpha^{4}+8\alpha^{3}+14\alpha^{2}$$

$$\alpha_{31} = 3\alpha^{49}+2\alpha^{48}+17\alpha^{47}+4\alpha^{46}+22\alpha^{45}+2\alpha^{44}+5\alpha^{43}+2\alpha^{42}+20\alpha^{40}+9\alpha^{39}+12\alpha^{38}+25\alpha^{37}+10\alpha^{36}+5\alpha^{35}+5\alpha^{34}+21\alpha^{33}+18\alpha^{32}+26\alpha^{31}+10\alpha^{30}+12\alpha^{29}+13\alpha^{28}+\alpha^{27}+16\alpha^{26}+16\alpha^{25}+5\alpha^{24}+23\alpha^{23}+3\alpha^{22}+25\alpha^{21}+6\alpha^{20}+18\alpha^{19}+2\alpha^{18}+12\alpha^{17}+26\alpha^{16}+11\alpha^{15}+24\alpha^{14}+3\alpha^{13}+12\alpha^{12}+8\alpha^{11}+14\alpha^{10}+16\alpha^{9}+8\alpha^{8}+23\alpha^{7}+12\alpha^{6}+15\alpha^{5}+12\alpha^{4}+20\alpha^{3}+4\alpha^{2}$$

$$\alpha_{32} = 8\alpha^{49} + 22\alpha^{48} + 12\alpha^{47} + \alpha^{46} + 9\alpha^{45} + 22\alpha^{44} + 2\alpha^{43} + 23\alpha^{42} + 23\alpha^{41} + 25\alpha^{40} + 2\alpha^{39} + 15\alpha^{38} + 7\alpha^{37} + 18\alpha^{36} + 18\alpha^{35} + 14\alpha^{34} + 20\alpha^{33} + 23\alpha^{32} + 6\alpha^{31} + 17\alpha^{30} + 11\alpha^{29} + 8\alpha^{28} + 1\alpha^{27} + 1\alpha^{26} + 24\alpha^{25} + 23\alpha^{24} + 20\alpha^{23} + 24\alpha^{22} + 10\alpha^{21} + 7\alpha^{20}+25\alpha^{19}+7\alpha^{18}+11\alpha^{17}+21\alpha^{16}+4\alpha^{15}+21\alpha^{14}+21\alpha^{13}+4\alpha^{12}+7\alpha^{11}+19\alpha^{10}+4\alpha^{9}+13\alpha^{8}+3\alpha^{7}+24\alpha^{6}+3\alpha^{5}+24\alpha^{4}+8\alpha^{3}$$

$$\alpha_{33} = 11\alpha^{49} + 24\alpha^{48} + 21\alpha^{47} + 7\alpha^{46} + 2\alpha^{45} + 11\alpha^{44} + 26\alpha^{43} + 13\alpha^{42} + 7\alpha^{41} + 20\alpha^{40} + 18\alpha^{39} + 6\alpha^{38} + 11\alpha^{37} + 17\alpha^{36} + 13\alpha^{35} + 26\alpha^{34} + 16\alpha^{33} + 8\alpha^{32} + 21\alpha^{31} + 16\alpha^{30} + 17\alpha^{29} + 21\alpha^{27} + 20\alpha^{26} + 21\alpha^{25} + 26\alpha^{24} + 24\alpha^{23} + 14\alpha^{22} + 7\alpha^{21} + 5\alpha^{20} + 20\alpha^{19} + 12\alpha^{18} + 12\alpha^{17} + 5\alpha^{16} + 15\alpha^{15} + 8\alpha^{14} + 23\alpha^{13} + 20\alpha^{12} + 14\alpha^{11} + 2\alpha^{10} + 24\alpha^{9} + 19\alpha^{7} + 23\alpha^{6} + 17\alpha^{5} + 3\alpha^{4} + 7\alpha^{3} + 25\alpha^{2} + 11\alpha + 10.$$

Using Free generators theorem,we can generate an infinite number of hash functions by varying p and n values.Choosing such choices of suitable parameters,we will create the best hash functions.These hash functions are security resistance to the previous attacks on the Tillich-Zemor hash function.

Conclusion

Cayley Hash functions are efficient cryptographic hash functions constructed from Cayley graphs.Hashing zero's and one's to the discrete Heisenberg group over F_p with the generators A= $\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ and B= $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$. The proposal which defined here is efficient and it attains $4n$ additions and n multiplications to hash a binary string of length n.Additionally, one of the computational properties of the Cayley hash functions are parallizable, which can enhance the efficiency of the defined proposal.Also, evaluated the randomness of the output distribution of the hashed values are uniform which reduced the occurrence of collisions.The girth of the hash function in discrete Heisenberg group over F_p gives the lower bound of the hash function H_1 is $log_2(p)$.For example,if choose p is of order 2^{256} then the defined hash functions with two generators does not have collisions except if one of the colliding bit strings of length is atleast 256.Finally, analysed the hash function with known attacks like generic attacks,subgroup attacks, elements of small order which can be averted which says that our hash function H_1 has no visible threat.

Even though the defined hash function H_1 is secured,it has one important weak-

ness, for short messages the pre-image can be calculated.The one solution is the padding method.The other solution for this weakness in H_1 is to introduce a new vectorial version of the hash function H_2.This function H_2 is very efficient and the computation of the vector-by-matrix multiplication takes only p additional requirements, which is negligible for long messages.Here also, the output distribution of the hash functions are uniform which reduce the occurrence of collisions.

The elements of neighbourhoods in $\mathbb{P}^2$ with respect to the metric are proved.Several intermediary results were obtained during the above studies.Free Generators theorem are proved as an application of Ping-Pong lemma in [73] and Free Generators theorem is applicable for a prime $p \geqslant 2$ and $d \geqslant 0$.This theorem provides an extensive choice of such generators.

Applying the Free Generators theorem,the hash functions are constructed and it retains its Modifications property using degree argument.Also verify the security properties of the hash functions using different attacks.The padding will completely remove the weakness of the determinant on security and also the distribution problem in determinant.

Finally, Cryptographic hash functions like SHA-256 and SHA-512 have a maximum input message size of $2^{64} - 1$ bits and $2^{128} - 1$ bits, respectively [67].Here our defined proposals of hash functions in the above context, takes arbitrary input to output the fixed length hash values.

There is sufficient extent for the future studies in the context of above investigations

- Keyed version of the hash function using discrete Heisenberg group can be defined and analysed along with its security.

- Cryptanalysis of the hash functions discussed in chapter 2 and chapter 5 can be determined.

- Finding condition for which $< \pi(\tilde{A}), \pi(\tilde{B}) >$ generate all of $GL_3(F_q)$ is upto the future work.

- The proper choosing of padding and finding out the small modifications property are existing in $PGL_3(F_p((x)))$ are to be analysed in future.

- The speed and distribution of the elements in $\mathbb{D}$ are to be examined in the future.

- The method appears in chapter 4 can be non-trivially extended for finding generators of free groups in $GL_r(F_p((x)))$ or $PGL_r(F_p((x)))$ for $r > 3$ could be developed.

- Applying the ideas of Free Generators theorem to construct free generators GL_3 over the local field like Q_p provided by its appropriate distance is left to the future work.

- Keyed version of the hash function can be produced and analysed using the ideas in chapter 5.

Appendices

Appendix A:Average Running time of the Hash function H_1

a=((1, 1, 0), # matrix A #

(0, 1, 0),

(0, 0, 1))

b=((1, 0, 0), # matrix B #

(0, 1, 1),

(0, 0, 1))

#inputString = raw-input("InputString.")

print('Keep input string in a file called data.txt')

#with open('data.txt', 'r') as myfile:

#inputString=myfile.read().replace('/n', '')

#myfile.close()

lines = [line.rstrip('/n') for line in open('data.txt')]

modval = input("Mod Value.")

def MatrixMul(mtx-a, mtx-b):

tpos-b = zip(*mtx-b)

rtn = [[sum(ea*eb for ea,eb in zip(a,b)) for b in tpos-b] for a in mtx-a]

return rtn

def MatrixMod(mtx-v, modval1):

rtn=[[0, 0, 0], # matrix B #

[0, 0, 0],

[0, 0, 0]]

i=0

while $i < 3$:

j=0

while $j < 3$:

rtn[i][j]=mtx-v[i][j]%modval1

j + = 1

i + = 1

return rtn

f=open(“result.txt”, “a+”)

f.write(“++++++++++++++++++++++++++++++

++++++++++++++++++++++ %s ++++++++++

++

++++++++++++++++++ /r”%(str(datetime.datetime.now())))

f.write(“A:%s B:%s Mod:%d /r /n”%(str(a),str(b),modval))

timelist = [ ]

for inputString in lines:

start = time.time()

i = 0

x=((1, 0, 0), # matrix #

$(0, 1, 0)$,

$(0, 0, 1))$

while $i < len(inputString)$:

if inputString [i] == '0':

y=a

elif inputString [i] == '1':

y=b

v = MatrixMul(x, y)

x=v

i + = 1

w = MatrixMod(v,modval)

done = time.time()

elapsed = float(done) - float(start)

#print('Time:')

#print(elapsed)

f.write("InputStr:%s Len:%d MulMatrix:%s

ModMatrix:%s Time:%f /r /n"%(inputString,

len(inputString),str(v),str(w),elapsed))

timelist.append(elapsed)

averTime = sum(timelist) / len(timelist)

print averTime

f.write("averTime: %f /r /n"%(averTime))

f.close().

Appendix B: Chi-Square Test Code for H_1

```
import time
import datetime
from scipy.stats
import chisquare
a = ((1, 1, 0), # matrix A #
(0, 1, 0),
(0, 0, 1))
b = ((1, 0, 0), # matrix B #
(0, 1, 1),
(0, 0, 1))
list12=[ ]
list13=[ ]
list23=[ ]
a12RejNo = 0
a13RejNo = 0
a23RejNo = 0
#inputString = raw-input("InputString. ")
#print('Keep input string in a file called data.txt')
#with open('data.txt', 'r') as myfile:
#inputString=myfile.read().replace(' /n', '')
#myfile.close()
def MatrixMul( mtx-a, mtx-b):
tpos-b = zip( *mtx-b)
rtn = [[ sum( ea*eb for ea,eb in zip(a,b)) for b in tpos-b] for a in mtx-a]
```

return rtn

def MatrixMod(mtx-v, modval1):

rtn=[[0, 0, 0], # matrix B #

[0, 0, 0],

[0, 0, 0]]

i=0

while $i < 3$:

j=0

while $j < 3$:

rtn[i][j]=mtx-v[i][j] % modval1

$j+ = 1$

$i+ = 1$

return rtn

def MatrixOp(cnt,inputString,modval1):

start = time.time()

i = 0

x=((1, 0, 0), # matrix X #

(0, 1, 0),

(0, 0, 1))

while $i < len(inputString)$:

if inputString[i] == '0':

y=a

elif inputString[i] == '1':

y=b

v = MatrixMul(x, y)

```
x=v
i += 1
w = MatrixMod(v,modval1)
done = time.time()
elapsed = done - start
#print ' Time:'.format(cnt, elapsed)
#print(elapsed)
f=open("result.txt", "a+")
#f.write("%d InputStr:%s Len:%d MulMatrix:
%s ModMatrix:%s Time:%.10f /r / n"%(cnt,inputString,
len(inputString),str(v),str(w),elapsed))
f.write("%d InputStr:%s Len:%d MulMatrix:%s ModMatrix:%s /r /n"
%(cnt,inputString,len(inputString),str(v),str(w)))
f.close()
list12.append(w[0][1])
list13.append(w[0][2])
list23.append(w[1][2])
return
modval = input("Mod Value.")
categ = input("Categories Value.")
d = input("Number of distributions ? (Above 1)")
notice = 'Keep input strings in a files named data1-'+ str(d) +'.txt'
print notice
n = modval*modval*modval
expFre = n/float(categ)
```

f-exp1 = [expFre for i in range(categ)]

print “expFre(SHOULD BE ABOVE 5):”.format(expFre)

#print expFre

f=open(“result.txt”, “a+”)

f.write(“————————————————%s ——-

————————————————————————————

————-/r /n”%(str(datetime.datetime.now())))

f.write(“A:%s B:%s Mod:%d Categories:%d expFre:%f No Distribution:%d /r /n”

%(str(a),str(b),modval,categ,expFre,d))

f.close()

i = 1

while $i <= d$:

list12=[ ]

list13=[ ]

list23=[ ]

filepath = ’data’+str(i)+’.txt’

f=open(“result.txt”, “a+”)

f.write(“Set : %d %s /r /n”%(i,filepath))

f.close()

print “Set:”,i

with open(filepath) as fp:

line = fp.readline()

cnt = 1

while line:

#print(“Line : ”.format(cnt, line.strip()))

```
MatrixOp(cnt,line.strip(),modval)
line = fp.readline()
cnt += 1
fp.close()
f=open("result.txt", "a+")
f.write("list12:%s list13:%s list23:%s /r /n"%(list12,list13,list23))
chi12 = chisquare(list12,f-exp = f-exp1)
chi13 = chisquare(list13,f-exp = f-exp1)
chi23 = chisquare(list23,f-exp = f-exp1)
c12,p12 = chi12
s12 = "H0 REJECTED"if c12 > p12 else "H0 ACCEPTED"
c13,p13 = chi13
s13 = "H0 REJECTED"if c13 > p13 else "H0 ACCEPTED"
c23,p23 = chi23
s23 = "H0 REJECTED"if c23 > p23 else "H0 ACCEPTED"
f.write("list12:%s chisquare12:%s %s /r /n"%(list12,chi12,s12))
f.write("list13:%s chisquare13:%s %s /r /n"%(list13,chi13,s13))
f.write("list23:%s chisquare23:%s %s /r /n"%(list23,chi23,s23))
print chi12,s12
print chi13,s13
print chi23,s23
if s12 == "H0 REJECTED":
a12RejNo += 1
if s13 == "H0 REJECTED":
a13RejNo += 1
```

```
if s23 == “H0 REJECTED”:
a23RejNo += 1
f.close()
i += 1
f=open(“result.txt”, “a+”
f.write(“a12RejNo: %d a12RejNo:%d a23RejNo:
%d /r /n”%(a12RejNo,a13RejNo, a23RejNo))
f.write(“pass proportion of a12: %f a13:%f a23:%f /r /n”
%(float(a12RejNo)/float(d),
float(a13RejNo)/float(d), float(a23RejNo)/float(d)))
f.close()
print“a12RejNo:”,a12RejNo
print“a13RejNo:”,a13RejNo
print“a23RejNo:”,a23RejNo
print('Finish').
```

Appendix C: Average Running time of the Hash function H_2

a=((1, 1, 0), # matrix A #

(0, 1, 0),

(0, 0, 1))

b=((1, 0, 0), # matrix B #

(0, 1, 1),

(0, 0, 1))

#inputString = raw-input("InputString. ")

print('Keep input strings in a file called data.txt')

lines = [line.rstrip('/n') for line in open('data.txt')]

print('Keep random const in a file called ranconst.txt')

with open('ranconst.txt', '/r') as myfile1:

Crnd=myfile1.read().replace('/n', '')

myfile1.close()

#print "inputString=",inputString

#print "Crand=", Crnd

modval = input("Mod Value. ")

def MatrixMul(mtx-a, mtx-b):

tpos-b = zip(*mtx-b)

rtn = [[sum(ea*eb for ea,eb in zip(a,b)) for b in tpos-b] for a in mtx-a]

return rtn

#3*3 Matrix mode

def MatrixMod(mtx-v, modval1):

rtn=[[0, 0, 0], # matrix B #

[0, 0, 0],

[0, 0, 0]]

i=0

while $i < 3$: j=0

while $j < 3$:

rtn[i][j]=mtx-v[i][j] %modval1

j += 1

i += 1

return rtn

#1*3 matrix mode

def MatrixMod1(mtx-v, modval1):

rtn=[0, 0, 0]

i=0

while $i < 3$:

rtn[i]=mtx-v[i]%modval1

i += 1

return rtn

def binaryString(arr):

binstr=“ ”

for x in arr:

binstr+= str(“0:b”.format(x))

#print binstr

return binstr

def Xor(a1,b1):

#print “xor ”, a , b , $bool(a)! = bool(b)$

res = $bool(a1)! = bool(b1)$

return str(int(res == True))

def str2bool(v):

return v.lower() in (“yes”, “true”, “t”, “1”)

def prependZeros(str,no):

while $no > 0$:

str = “0”+ str

no -= 1

return str

def hashFunction(a, b, inputString, modval):

i = 0

x=((1, 0, 0), # matrix X #

(0, 1, 0),

(0, 0, 1))

while $i < len(inputString)$:

if inputString[i] == ’0’:

y=a

elif inputString[i] == ’1’:

y=b

v = MatrixMul(x, y)

x=v

i += 1

w = MatrixMod(v,modval)

return v,w

f=open(“result.txt”, “a+”)

f.write(“————————————————%s ————————

————————————————————————————————/r/n”

%(str(datetime.datetime.now())))

f.write(“A:%s B:%s Mod:%d Crand:%s /r/n”%(str(a),str(b),modval,Crnd))

timelist = [ ]

for inputString in lines:

start = time.time()

v,w = hashFunction(a, b, inputString, modval)

w0 = w[0]

binStr = binaryString(w0)

hvecm-len=len(binStr)

crnd-len=len(Crnd)

if $hvecm - len > crnd - len$:

Crnd = prependZeros(Crnd, hvecm-len - crnd-len)

elif $crnd - len > hvecm - len$:

binStr = prependZeros(binStr, crnd-len - hvecm-len)

#print “hvecm=”, binStr

#print “Crand=”, Crnd

if $len(binStr)! = len(Crnd)$:

print “Error $len(binStr)! = len(Crnd)$”

lenStr = len(Crnd)

i=0

exorResult=“ ”

while $i < lenStr$:

```
#print binStr[i], Crnd[i]
exorResult = exorResult + Xor(str2bool(binStr[i]),str2bool(Crnd[i]))
i += 1
#print "exorResult", exorResult
v1,w1 = hashFunction(a, b, exorResult, modval)
#print type(w0)
#print type(w1)
t1=[[0,0,0]]
t1[0]=w0
#print t1
mulmat2=MatrixMul(t1,w1)
#print mulmat2
#modmat2 = MatrixMod1(mulmat2[0],modval)
#print modmat2
done = time.time()
elapsed = float(done) - float(start)
#print('Time:')
#print(elapsed)
f.write("InputStr:%s Len:%d MulMatrix:
%s ModMatrix:%s %s /r"%(inputString,
len(inputString),str(v),str(w),str(w0) ))
f.write("HVec(M): %s binStr: %s exorResult:
%s MulMatrix:%s ModMatrix:%s H1(M)(MUL):
%s Time:%f /r/n"%(str(t1), binStr,exorResult,str(v1),str(w1),
str(mulmat2),elapsed))
```

```
timelist.append(elapsed)
averTime = sum(timelist) / len(timelist)
print averTime
f.write("averTime: %f /r/n"%(averTime))
f.close().
```

Appendix D: Chi-Square Test Code for H_2

```
import time
import datetime
from scipy.stats import chisquare
a=( (1, 1, 0), # matrix A #
(0, 1, 0),
(0, 0, 1))
b=( (1, 0, 0 ), # matrix B #
(0, 1, 1),
(0, 0, 1))
list12=[ ]
list13=[ ]
a12RejNo = 0
a13RejNo = 0
#inputString = raw-input("InputString.")
#print('Keep input string in a file called data.txt')
#with open('data.txt', '\r') as myfile:
# inputString=myfile.read().replace('\n', '')
#myfile.close()
print('Keep random const in a file called ranconst.txt')
with open('ranconst.txt', '\r') as myfile1:
Crnd=myfile1.read().replace('\n', '')
myfile1.close()
#print "inputString=",inputString
#print "Crand=", Crnd
```

modval = input(“Mod Value.”)

categ = input(“Categories Value.”)

d = input(“Number of distributions ? (Above 1)”)

notice = ’Keep input strings in a files named data1-’+ str(d) +’.txt’

print notice

n = modval*modval*modval

expFre = n/float(categ)

f-exp1 = [expFre for i in range(categ)]

print “expFre(SHOULD BE ABOVE 5):”.format(expFre)

def MatrixMul(mtx-a, mtx-b):

tpos-b = zip(*mtx-b)

rtn = [[sum(ea*eb for ea,eb in zip(a,b)) for b in tpos-b] for a in mtx-a]

return rtn

#3*3 Matrix mode

def MatrixMod(mtx-v, modval1):

rtn = [[0, 0, 0], # matrix B #

[0, 0, 0],

[0, 0, 0]]

i=0

while $i < 3$:

j=0

while $j < 3$:

rtn[i][j]=mtx-v[i][j] %modval1

j += 1

i += 1

return rtn

#1*3 matrix mode

def MatrixMod1(mtx-v, modval1):

rtn=[0, 0, 0]

i=0

while $i < 3$:

rtn[i]=mtx-v[i] %modval1

i += 1

return rtn

def binaryString(arr):

binstr=“ ”

for x in arr:

binstr+= str(“0:b”.format(x))

#print binstr

return binstr

def Xor(a1,b1):

#print “xor”, a , b , bool(a) != bool(b)

res = bool(a1) != bool(b1)

return str(int(res == True))

def str2bool(v):

return v.lower() in (“yes”, “true”, “t”, “1”)

def prependZeros(str,no):

while $no > 0$:

str = “0”+ str

no -= 1

```
return str
def hashFunction(a, b, inputString, modval):
i = 0
x=( (1, 0, 0), # matrix X #
(0, 1, 0),
(0, 0, 1))
while i < len(inputString):
if inputString[i] == '0':
y=a
elif inputString[i] == '1':
y=b
v = MatrixMul( x, y )
x=v
i += 1
w = MatrixMod(v,modval)
return v,w
def oneLine(cnt,inputString,Crnd):
start = time.time()
v,w = hashFunction(a, b, inputString, modval)
w0 = w[0]
binStr = binaryString(w0)
hvecmlen=len(binStr)
crndlen=len(Crnd)
if hvecmlen > crndlen:
Crnd = prependZeros(Crnd, hvecmlen - crndlen)
```

```
elif crndlen > hvecmlen:
binStr = prependZeros(binStr, crndlen - hvecmlen)
#print "hvecm=", binStr
#print "Crand=", Crnd
if len(binStr) != len(Crnd):
print "Error len(binStr) != len(Crnd)"
lenStr = len(Crnd)
i=0
exorResult=" "
while i < lenStr:
#print binStr[i], Crnd[i]
exorResult = exorResult + Xor(str2bool(binStr[i]),str2bool(Crnd[i]))
i += 1
#print "exorResult", exorResult
v1,w1 = hashFunction(a, b, exorResult, modval)
#print type(w0)
#print type(w1)
t1=[[0,0,0]]
t1[0]=w0
#print t1
mulmat2=MatrixMul(t1,w1)
#print mulmat2
#modmat2 = MatrixMod1(mulmat2[0],modval)
#print modmat2
done = time.time()
```

```
elapsed = done - start
#print cnt, 'Time:', elapsed
#print(elapsed)
#print str(datetime.datetime.now())
f=open("result.txt", "a+")
f.write("-%d- \r"%(cnt))
f.write("InputStr:%s Len:%d MulMatrix:%s ModMatrix:%s
%s \r"%(inputString,len(inputString),str(v),str(w),str(w0) ))
f.write("HVec(M): %s %s Crand:%s exorResult:%s MulMatrix:
%s ModMatrix:%s H1(M)(MUL):%s Time:%f \rsetminusn"
%(str(t1), binStr,Crnd,exorResult,str(v1),str(w1), str(mulmat2), elapsed ))
f.close()
list12.append(mulmat2[0][1])
list13.append(mulmat2[0][2])
f=open("result.txt", "a+")
f.write("++++++++++++++++++++++++++
++++++++++++++++++++++++++ %s ++++++
++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++
\r"%(str(datetime.datetime.now())))
f.write("A:%s B:%s Mod:%d Categories:%d expFre:%f No Distribution:%d \r\n"
%(str(a),str(b),modval,categ,expFre,d))
f.close()
i = 1
while i <= d:
```

```
list12=[ ]
list13=[ ]
filepath = 'data'+str(i)+'.txt'
f=open("result.txt", "a+")
f.write("Set : %d %s \rsetminusn"%(i,filepath))
f.close()
print "Set:",i
with open(filepath) as fp:
line = fp.readline()
cnt = 1
while line:
#print("Line : ".format(cnt, line.strip()))
oneLine(cnt,line.strip(),Crnd)
line = fp.readline()
cnt += 1
fp.close()
f=open("result.txt", "a+")
f.write("list12:%s list13:%s \rsetminusn"%(list12,list13))
chi12 = chisquare(list12,f-exp = f-exp1)
chi13 = chisquare(list13,f-exp = f-exp1)
c12,p12 = chi12
s12 = "H0 REJECTED"if c12 > p12 else "H0 ACCEPTED"
c13,p13 = chi13
s13 = "H0 REJECTED"if c13 > p13 else "H0 ACCEPTED"
f.write("list12:%s chisquare12:%s %s \r\n"%(list12,chi12,s12))
```

```
f.write("list13:%s chisquare13:%s %s \r\n"%(list13,chi13,s13))
print chi12,s12
print chi13,s13
if s12 == "H0 REJECTED":
a12RejNo += 1
if s13 == "H0 REJECTED":
a13RejNo += 1
f.close()
i += 1
f=open("result.txt", "a+")
f.write("a12RejNo: %d a12RejNo:%d \r\n"%(a12RejNo,a13RejNo))
f.write("pass proportion of a12: %f a13:%f \r\n"%(float(a12RejNo)/float(d),
float(a13RejNo)/float(d)))
f.close()
print"a12RejNo:",a12RejNo
print"a13RejNo:",a13RejNo
print('Finish').
```

Appendix E: Average running time of the Hash Function H_3

```
import time
import datetime
a=((1,1,0), # matrix A #
(0,1,0),
(0,0,1))
b=((1,0,0), # matrix B #
(0,1,1),
(0,0,1))
d=( (1, 1, 1 ), # matrix D DeprecationWarning #
(0, 1, 1),
(0, 0, 1))
#inputString = raw-input("InputString.")
print('Keep input string in a file called data.txt')
#with open('data.txt', '\r') as myfile:
#inputString=myfile.read().replace('\n', '')
#myfile.close()
lines = [line.rstrip('\n') for line in open('data.txt')]
print('Keep Matrix C in a file called matc.txt')
matlines = [line.rstrip('\n') for line in open('matc.txt')]
#inputString = lines[0]
c= [[0,0,0], # matrix C #
[0,0,0],
```

[0, 0, 0]]

j = 0

for x in matlines:

numbers = map(int, x.split())

c[j] = numbers

j += 1

print d

modval = input(“Mod Value.”)

gval = input(“Integer.”)

def MatrixMul(mtx-a, mtx-b):

tpos-b = zip(*mtx-b)

rtn = [[sum(ea*eb for ea,eb in zip(a,b)) for b in tpos-b]

for a in mtx-a]

return rtn

#3*3 Matrix mode

def MatrixMod(mtx-v, modval1):

rtn=[[0, 0, 0], # matrix B #

[0, 0, 0],

[0, 0, 0]]

i=0

while $i < 3$:

j=0

while $j < 3$:

rtn[i][j]=mtxv[i][j]%modval1

j += 1

i += 1

return rtn

#1*3 matrix mode

def MatrixMod1(mtx-v, modval1):

rtn=[0, 0, 0]

i=0

while $i < 3$:

rtn[i]=mtx-v[i]%modval1

i += 1

return rtn

def binaryString(arr):

binstr=“ ”

for x in arr:

binstr+= str(“0:b”.format(x))

#print binstr

return binstr

def Xor(a1,b1):

#print “xor”, a , b , $bool(a)! = bool(b)$

$res = bool(a1)! = bool(b1)$

return str(int(res == True))

def str2bool(v):

return v.lower() in (“yes”, “true”, “t”, “1”)

def prependZeros(str,no):

while $no > 0$:

str = “0”+ str

```
no -= 1
return str
def hashFunction(a, b, inputString, modval):
i = 0
x=( (1, 0, 0), # matrix X #
(0, 1, 0),
(0, 0, 1))
while i < len(inputString):
if inputString[i] == '0':
y=a
elif inputString[i] == '1':
y=b
v = MatrixMul( x, y )
x=v
i += 1
w = MatrixMod(v,modval)
return v,w
def MulFunction(inputString):
i = 0
x=( (1, 0, 0), # matrix X #
(0, 1, 0),
(0, 0, 1))
while i < len(inputString):
if inputString[i] == '0':
y=a
```

```
elif inputString[i] == '1':
y=b
v = MatrixMul( x, y )
x=v
if True == devides(i+1,gval):
#print gval, "devides", i+1
y=c
v = MatrixMul( x, y )
x=v
i += 1
#print v
#print modval
w = MatrixMod(v,modval)
#print w
return v,w
def MatrixBinaryStr(mtx):
#assume 3*3 matrix
val = u"
i=0
while i < 3:
j=0
while j < 3:
val += bin(mtx[i][j])[2:]
j += 1
i += 1
```

return val

def devides(x,y):

if(x%y == 0):

return True

else:

return False

f=open(“result.txt”, “a+”)

f.write(“————————————————%s ———-

————————————————————————————-

\r\n”%(str(datetime.datetime.now())))

f.write(“a:%s b:%s c:%s modval:%d g:%d \r\n”

%(str(a),str(b),str(c),modval,gval))

timelist = []

for inputString in lines:

start = time.time()

index = 1

v,w = MulFunction(inputString)

done = time.time()

elapsed = float(done) - float(start)

print('Time:')

print(elapsed)

print v, w

f.write(“Mul: %s ModMul:%s Time :%s \r\n”%(str(v),str(w), str(elapsed)))

timelist.append(elapsed)

averTime = sum(timelist) / len(timelist)

print averTime

f.write("averTime: %f \r\n"%(averTime))

f.close().

Appendix F: Execution of H_4

```
import time
import datetime
import numpy as np
AA = ( ( [3,0], [2,0], [0]),
( [4,0] , [3,0], [0]),
( [2,1], [2,0], [2]) )
BB = ( ( [2,0,0], [0], [2,0,1]),
( [0] , [2,0,0], [2,0,1]),
( [0], [0], [2]) )
result =[[[0], [0], [0]],
[[0], [0], [0]],
[[0], [0], [0]]]
print('Keep input strings in a file called data.txt')
#inputString = raw-input("InputString. ")
lines = [line.rstrip('\n') for line in open('data.txt')]
#print "inputString=",inputString
pVal = input("p Value.")
cp = np.poly1d([1, 0, -1, 1])
print "Conway Polynomial:"
print cp
nVal = 3
print "n:",nVal
def polyMultiply(x,y):
p1 = np.poly1d(x)
```

p2 = np.poly1d(y)

#print(p1)

#print(p2)

mul = np.polymul(p1, p2)

#print “MUL:”, mul

return mul

def MatrixMul(mtx-a, mtx-b):

tpos-b = zip(*mtx-b)

rtn =$[[sum(ea * ebforea, ebinzip(a, b))forbintpos - b]forainmtx - a]$

return rtn

def MatrixMulPolyTestFail(mtx-a, mtx-b):

tpos-b = zip(*mtx-b)

rtn = $[[np.polyadd(polyMultiply(ea, eb)forea, ebinzip(a, b))$

$forbintpos_b]forainmtx - a]$

return rtn

def MatrixMulPoly(A,B):

iterating by row of A

result = [[[0], [0], [0]],

[[0], [0], [0]],

[[0], [0], [0]]]

sm = []

for i in range(len(A)):

iterating by coloum by B

for j in range(len(B[0])):

iterating by rows of B

```
for k in range(len(B)):
#print “polyMul:”, polyMultiply(A[i][k],B[k][j]
sm = np.polyadd(result[i][j],polyMultiply(A[i][k],B[k][j] ))
result[i][j] = sm #np.array(sm)
#print “sm”, result[i][j]
#print “ELEMENT”, result[i][j]
return result
def PrintPolyMatrix(A,f):
# iterating by row of A
print ’Mod Matrix \n————————————————————’
f.write(“Mod Matrix \n————————————————————”)
i = 0
while i < 3:
j=0
while j < 3:
aa = A[i][j]
kk = np.poly1d(aa)
print “Matrix Position:(“,i+1,j+1,”) \n”, kk
f.write(“\nMatrix Position:( %d %d ) \n %s \r”%(i+1,j+1,kk))
j + = 1
i + = 1
print ’\n————————————————————’
f.write(“\n————————————————————”)
#3*3 Matrix mode
def MatrixMod( mtx-v, modval1):
```

rtn=[[0, 0, 0], # matrix B #

[0, 0, 0],

[0, 0, 0]]

i=0

while $i < 3$:

j=0

while $j < 3$:

rtn[i][j]=mtx-v[i][j]%modval1

j + = 1

i + = 1

return rtn

#3*3 Matrix mode

def MatrixModPoly(mtx-v, modval1):

rtn=[[[0], [0], [0]],

[[0], [0], [0]],

[[0], [0], [0]]]

i=0

while $i < 3$:

j=0

while $j < 3$:

rtn[i][j]=np.mod(mtx-v[i][j], modval1)

j+ = 1

i+ = 1

return rtn

#1*3 matrix mode

def MatrixMod1(mtx-v, modval1):

rtn=$[0, 0, 0]$

i=0

while $i < 3$:

rtn[i]=mtx-v[i]%modval1

i + = 1

return rtn

def binaryString(arr):

binstr=“”

for x in arr:

binstr+= str(“0:b”.format(x))

#print binstr

return binstr

def Xor(a1,b1):

#print “xor”, a , b , $bool(a)! = bool(b)$

$res = bool(a1)! = bool(b1)$

return str(int(res == True))

def str2bool(v):

return v.lower() in (“yes”, “true”, “t”, “1”)

def prependZeros(str,no):

while $no > 0$:

str = “0”+ str

$no- = 1$

return str

def hashFunction(a, b, inputString, modval):

```
i = 0
x=( (1, 0, 0), # matrix X #
(0, 1, 0),
(0, 0, 1))
while i < len(inputString):
if inputString[i] == '0':
y=a
elif inputString[i] == '1':
y=b
v = MatrixMul( x, y )
x=v
i += 1
w = MatrixMod(v,modval)
return v,w
def polyProduct(a, b, inputString, modval):
i = 0
x =[[[1], [0], [0]],
[[0], [1], [0]],
[[0], [0], [1]]]
while i < len(inputString):
if inputString[i] == '0':
y=AA
elif inputString[i] == '1':
y=BB
v = MatrixMulPoly( x, y )
```

#print v

x=v

i + = 1

w = MatrixModPoly(v,modval) return v,w

def notString(inputString):

i = 0

outStr = ”

while $i < len(inputString)$:

if inputString[i] == ’0’:

outStr += ’1’

elif inputString[i] == ’1’:

outStr += ’0’

i += 1

return outStr

modval = pow(pVal, nVal)

f=open(“result.txt”, “a+”)

f.write(“————————————————%s ——\r\n”

%(str(datetime.datetime.now())))

f.write(“A:%s \nB:%s \np:

%d n:%d Mod:%d \nConvey Poly: \n%s \r”%(str(AA),str(BB),pVal,nVal, modval,

str(cp)))

timelist = []

for inputString in lines:

print “inputString ”,inputString

```
f.write("\n inputString:%s Len:%d \r"%(inputString,len(inputString)))
start = time.time()
inputString += notString(inputString)
print "m ",inputString
remul,remod = polyProduct(AA,BB,inputString, modval)
done = time.time()
elapsed = float(done) - float(start)
#print remod
print('Elapsed Time:')
print(elapsed)
f.write("m:%s Len:%d \r"%(inputString,len(inputString)))
f.write("remul:%s \nremod:%s \nTime:%f\r"%(str(remul),str(remod),elapsed))
PrintPolyMatrix(remod,f)
timelist.append(elapsed)
averTime = sum(timelist) / len(timelist)
print "averageTime:", averTime
f.write("\n averageTime: %f \r\n"%(averTime))
f.close().
```

Research Papers

1. V.Vibitha Kochamani, Lilly P.L and Joju.K.T, *Hashing with discrete Heisenberg group and graph with large girth*,In International Journal of Theoretical Physics and Cryptography, IJTPC Publication,Vol 11, May 2016.

2. Lilly P.L,V.Vibitha Kochamani, *Hashing with discrete Heisenberg group Using new generators*,In International Journal of Theoretical and Computational Mathematics,ISSN: 2395-6607,Vol 2, Issue 2, Nov.2016,pp 14-18.

3. V.Vibitha Kochamani, Lilly P.L, *Modified Form of Cayley Hash Function*, In Asian Journal of Engineering and Applied Technology ,The Research Publication, ISSN 2249-068X,Vol. 8 No. 2,April 2019 ,pp 34-36.

4. V.Vibitha Kochamani, Lilly P.L, *Security Aspects of the Cayley Hash Function using discrete Heisenberg group*,In Journal of Discrete Mathematical Sciences and Cryptography,ISSN:0972-0529 (Print) , ISSN:2169-0065 (On-line),Doi:10.1080/09720529.2019.1638613,September 2019.

5. V.Vibitha Kochamani, Lilly P.L, *Neighbourhoods in* $\mathbb{P}^2$, In International Journal of Research and Analytical Reviews, E-ISSN 2348-1269, P- ISSN 2349-5138, vol.6,issue.2,June 2019.

6. V.Vibitha Kochamani, Lilly P.L, *Vectorial version of the Cayley Hash Function using discrete Heisenberg group*, Accepted in Malaya Journal of Matematik.

7. V.Vibitha Kochamani, Lilly P.L, *Free Generators Theorem in projective general linear groups*, Communicated.

8.V.Vibitha Kochamani, Lilly P.L, *Hash Functions Using Free Generators Theorem over projective general linear group*, Communicated.

9.V.Vibitha Kochamani, Lilly P.L, *Guarding against different attacks*, Communicated.

Bibliography

[1] Abdukhalikov Kanat and Chul Kim (1998),On the security of the hashing scheme based on SL_2,*Fast Software Encryption*,Springer, 93-102.

[2] G.Adj,Alfred Menezes, Thomaz Oliveira and Francisco Rodriguez-Henriquez (2015),Computing discrete logarithms using Joux's algorithm,*ACM Comm.Computer Algebra,* 49,2,60.

[3] B.Albright (1994),*Essentials of Mathematical Statistics*,Jones and Bartlett Publishers.

[4] Alfred J.Menezes and Yi-Hong Wu (1997),The discrete logarithm problem in $GL(n, q)$,*Ars Combin*,47,23-32.

[5] Andreas Gathmann (2013),Class Notes on Commutative Algebra,Technische Universitat, Kaiserslautern.

[6] Andrew M Odlyzko (1984),Discrete logarithms in finite fields and their cryptographic significance,*Workshop on the theory and Applications of Cryptographic Techniques*,Springer,224-314.

[7] Andrew Richard Regenscheid (2007),*An algebraic hash function based on* SL_2,Ph.D. thesis,Lowa state University.

[8] Aschbacher Michael (1984),On the maximal subgroups of the finite classical groups,*Inventiones mathematicae*,76,3,469-514.

[9] E.Bach and J.O Shallit (1996),*Algorithmic number theory:Efficient algorithms*,MIT Press,1.

[10] Bart Van Rompay (2004),*Analysis and design of Cryptographic Hash Functions:MAC algorithms and Block Ciphers*,Ph.D. thesis,KU Leuven, 240.

[11] M.Bellare and P.Rogaway (1997),Collision-resistant hashing:Towards making UOWHFs practical,*Advances in Cryptology-CRYPTO'97*,Springer,470-484.

[12] Bianca Sosnovski (2016),*Cayley graphs of semi-groups and Applications to Hashing*,Ph.D. thesis,City University of New York.

[13] Brian C Hall (2004),*Lie Groups, Lie Algebras, and Representations: An Elementary Introduction*,Berlin:Springer.

[14] Bomberg Lisa, Valdimir Shpilrain and Alina Vdovina (2017),Navigating in the Cayley graph of $SL_2(F_p)$ and applications to hashing,*Semi-group Fourm*,94,2,314-324.

[15] C.Caldwell (2018),The prime pages:Prime number research,records and resources,[Available at:https://primes.utm.edu/primes/1994-2018].

[16] J.Cassaigne, T.Harju and J.Karhumaki (1999),On the undecidability of freeness of matrix semi-groups, *International Journal of Algebra and Computation* 9,3/4,295-305.

[17] D.X Charles, K.E Lauter and E.Z. Goren (2009), Cryptographic Hash functions from expander graphs,*Journal of Crytology*,22,1,93-113.

[18] Charnes chris and Josef Pieprzyk (1995),Attacking the SL_2 hashing scheme,*ASIACRYPT'94:Advances in Cryptology*,322-330.

[19] Christophe Petit, Jean-Jacques Quisquater (2016), Cryptographic Hash functions and Expander graphs:The end of the Story?, *The New Code breakers,Lecture notes in Computer Science*,9100,304-311.

[20] Christophe Petit,Jean-Jacques Quisquater,Jean-Pierre Tillich and Gilles Zemor (2009),Hard and easy components of collision search in the Zemor-Tillich hash function: New attacks and reduced variants with equivalent security,*Cryptographers Track at the RSA Conference*,182-194.

[21] Christophe Petit, Jean-Jacques Quisquater (2010), Pre-images for the Tillich-Zemor hash function,*International Workshop on Selected Areas in Cryptography*,Springer,282-301.

[22] Christophe Petit and Jean-Jacques Quisquater (2013),Rubik's for cryptographers,*Notices of the American Mathematical Society*,60,6,733-739.

[23] Christophe Petit, Jean-Jacques Quisquater (2016),ZesT:an all-purpose hash function based on Zemor-Tillich,[Accessed from:https://www.cs.bham.ac.uk/petitcz.september 2016].

[24] Clara Loh (2017),*Geometric Group Theory-An Introduction*,Springer International Publishing.

[25] Cornelia Drutu,Michael Kapovich (2013),*Lectures on Geometric Group Theory*,UC Davis Mathematics.

[26] F.Crivelli (2008),Absolute Values, Valuations and Completion, Lecture Notes,*ETH Zurich University*.

[27] W.Dai (2009),Crypto++ 5.6.0 benchmarks,[Available at: http://www. cryptopp .com/benchmarks.html].

[28] Daugles Stinson R (1995),*Cryptography theory and practice*,Second Edition,Chapman and Hall/CRC.

[29] De la Harpe. P (2000),*Topics in Geometric Group Theory*,University of Chicago Press.

[30] Don Coppersmith, Andrew M Odlzyko and Richard Schroeppel (1986),Discrete logarithms in $GF(p)$,*Algorithmica*,Springer, 1 ,1-4,1-15.

[31] Don Coppersmith (1984),Fast evaluation of logarithms in fields of characteristic two,*IEEE transactions on information theory*,30,4,587-594.

[32] Emmanuel Breuillard and Tsachik Gelander (2003),On dense free subgroups of Lie groups,*Journal of Algebra*,261,2,448-46.

[33] Geiselmann Willi (1995),A Note on the hash function of Tillich and Zemor,*Cryptography and coding*,Proceedings of the 5th IMA conference on Cryptography and Coding,C.Boyd(Ed.),Springer, Berlin,LNCS 1025,257-263.

[34] Gilles Zemor (1994),Hash functions and Cayley Graph, *Designs, Codes and Cryptography*,Springer,4,3,381-394.

[35] Gilles Zemor (1991),Hash functions and graphs with large girths, EUROCRYPT (Donald W. Davies, ed.),*Lecture Notes in Computer Science*,Springer,LNCS 547,508-511.

[36] Ph.Godlewski and P.Camion (1988),Manipulations and errors,detection and localization,Advances in cryptology-EUROCRYPT'88 (Davos Ed.),Lecture Notes in Comput.Sci.,Springer,Berlin,330,97-106.

[37] Hallam Stevens (2018),Hans Peter Luhn and the birth of the hashing algorithm,*IEEE Spectrum*,55,2,44-49.

[38] Hayley Tomkins (2018),*Alternative Generators of the Zemor-Tillich Hash Function:A Quest for Freedom in Projective Linear Groups*,University of Ottawa.

[39] S.Hoory , N.Linial and A.Wigderson (2006),Expander graphs and their applications,*Bulletin of the American Mathematical Society*,43,4,439-561.

[40] Hyungrok Jo (2017),*Cryptanalysis on Hash functions Based on Ramanujan Graphs*,Ph.D. thesis,Kyushu University.

[41] Hyungrok Jo (2018),Hash function based on Ramanujan graphs, *Mathematical Modelling for Next-Generation Cryptography,Springer*,63-79.

[42] Jean Galleir (2011),*Geometric Methods and Applications for Computer Science and Engineering*,Second-Edition, Springer.

[43] Jean-Jacques Quisquater and Marc Joye (1997), Authentication of Sequences with the SL_2 hash function:Application to video sequences,*Journal of computer security*,5,3,213-223.

[44] Jean-Pierre Tillich and Gilles Zemor (2008), Collisions for the LPS expander graph hash function,*Annual International Conference on the theory and applications of Cryptographic Techniques,Springer*,254-269.

[45] Jean-Pierre Tillich and Gilles Zemor (1994),Group-Theoretic Hash Functions,*Algebraic coding:First French-Iseraeli workshop*,Springer,90-110.

[46] Jean-Pierre Tillich and Gilles Zemor (1994),Hashing with SL_2,*Advances in Cryptology Lecture Notes in Computer Science*,Springer-Verlag,839,40-49.

[47] Jeffrey Hoffstein, Jill Pipher and Joseph.H.Silverman (2014),*An Introduction to Mathematical Cryptography*,Springer.

[48] Jonathan Katz and Yehuda Lindell (2007), *Introduction to Modern Cryptography*,Chapman and Hall/CRC.

[49] Joseph.A Gallian (2005),*Contemporary Abstract Algebra-Ninth Edition*,University of Minnesota,Cengage learning, ISBN-10: 1133599702 — ISBN-13: 9781133599708.

[50] Luca Capogna, Danielli Donatella, Pauls Scott D and Tyson Jeremy (2007),*An Introduction to the Heisenberg Group and the Sub-Riemannian Isoperimetric*,Springer Science and Business Media, Mathematics ,224.

[51] Lyoden R.C. and Schupp P.E (1997),*Combinatorial Group Theory*,Ergebnisse der Matematik undihrer Grenzgebiete,Springer-Verlag,89.

[52] Maike Massierer (2006),*Provably Secure Cryptographic Hash Functions*,The university of New South Wales.

[53] Mark Brittenham,Free groups,*University of Nebraska*,Department of Mathematics,Lincoln.[Available at:https://www.math.unl.edu/mbrittenham2/classwk/990s08/public/myasnikov.1.free.groups.pdf]

[54] A.J. Menezes , P.C. Van Oorschot, and S.A Vanstone (2001),*Handbook of applied cryptography*,CRC press.

[55] Meulenaer De Giacomo , Christophe Petit and Jean-Jacques Quisquater (2009),Hardware implementations of a variant of the Zemor-Tillich hash function:Can a provably secure hash function be very efficient?,*IACR Cryptology eprint Archive*,229.

[56] R.A. Mollin (2001),*An introduction to cryptography*,Chapman and Hall/CRC.

[57] Morris J Dworkin (2018),SHA-3 standard:Permutation-based hash and extendable-output functions,*Technical report,NIST-FIPS*,(Accessed from: https://www.nist.gov/publications/sha-3-standard-permutation-based-hash-and -extendable-output-functions,june2018).

[58] C.Mullan and B.Tsaban (2016),SL_2 homomorphic hash functions:Worst case to average case reduction and short collision search,*Designs,Codes and Cryptography*,Springer,81,1,83-107.

[59] J.Neukrich (1999),*Algebraic Number Theory*, Springer-Verlag.

[60] C.Petit, K.E.Lauter and Jean-Jacques Quisquater (2007),Cayley hashes:A class of efficient graph-based hash functions,[Available at: http://perso.uclouvain.be/christophe.petit/files/Cayley.pdf,2007.

[61] C.Petit, K.E.Lauter and Jean-Jacques Quisquater (2008),Full cryptoanalysis of LPS and Morgenstern hash functions,*International Conference on Security and Cryptography for Networks*,Springer,263-277.

[62] C.Petit (2009),*On graph-based cryptographic hash functions*,Ph.D. thesis,Universit catholique de Loyvain.

[63] C.Petit, Nicolas Veyrat-Charvillon and Jean-Jacques Quisquater (2008),Efficiency and pseudo randomness of a variant of Zemor-Tillich hash function,*IEEE International Conference on Electronics,Circuts and systems*,ICECS 2008, 906-909.

[64] Preneel.B (2003),*Analysis and Design of Cryptographic Hash Functions*,Ph.D thesis,K.U.Leuven.

[65] M.O.Rabin (1978),Digitalized signatures, *Foundations of Secure computation*,R.Lipton and R.DeMillo,Eds., Academic Press,New York,155-166.

[66] Richard Crandall, Carl Pomerance (2005),*Prime Numbers: A Computational Perspective*,Second Edition,Springer.

[67] Quynh H.Dang (2015),Secure Hash Standards (SHS),*Federal Information Processing Standards Publication*,FIPS PUB 180-4,[Available at:http://dx.doi.org/10.6028/NIST.FIPS. 180-4].

[68] V.Shpilrain (2006),Hashing with polynomials, *Proceedings of the 9th international conference on Information Security and Cryptology*,Springer,4296,2228.

[69] Steinwandt Rainer,Markus Grassl,Willi Geiselmann and Thomas Beth (2000),Weakness in the $SL_2(F_{2^n})$ hashing scheme,*Annual International Cryptology Conference*,Springer,287-299.

[70] H.Stichtenoth (1993),*Algebraic function fields and codes*,Springer.

[71] B.Sury (2010),Free groups-basics,*Lecture Notes in Advanced Training in Mathematical Schools*,Statistical -Math Unit,Indian Statistical Institute,Bangalore.

[72] M.Suzuki (1982),*Group Theory: Volume-1*,Springer-verlag,New York.

[73] Tits Jacques (1972),Free Subgroups in linear groups, *Journal of Algebra*,20,2,250-270.

[74] Victor Shuop (1990),Searching for primitive roots in finite fields,*Proceedings of the twenty-second annual ACM symposium on theory of computing*,ACM, 546-554.

[75] J.Walker,Chi-square Calculator,[Available at:https://www.forumilab.ch.rpkp .experiments.analysis.chiCalc.html].