A Study of Algebraic Methods in Asymmetric Cryptography – Algorithms, Constructions, and Attacks

Dissertation

zur

Erlangung der naturwissenschaftlichen Doktorwürde (Dr. sc. nat.)

vorgelegt der

Mathematisch-naturwissenschaftlichen Fakultät

der

Universität Zürich

von

Simran Sunil Tinani

aus

Indien

Promotionskommission

Prof. Dr. Joachim Rosenthal (Vorsitz)
Prof. Dr. Andrew Kresch
Prof. Dr. Kenny Paterson

Zürich, 2023

To my dearest parents,
Inderjeet and Sunil

Abstract

Mathematical theory constitutes the foundation for the majority of constructions and algorithms in public-key cryptography. A paramount notion in cryptography is that of a one-way trapdoor function, which is a mathematical function that can efficiently be calculated in one direction, but is difficult or computationally infeasible to invert. Algebra, number theory and algebraic geometry serve as a rich pool of well-studied presumed one-way functions, and therefore lie at the core of the construction of mainstream cryptographic protocols.

The widely used Diffie-Hellman and RSA protocols are based, respectively, on the difficulty of the discrete logarithm problem of factoring of large integers. While they have shown consistent resilience to classical attacks, Peter Shor showed an efficient theoretical attack on these systems using a quantum algorithm. Therefore, post-quantum cryptography, viz. the study of cryptosystems that are resilient to attacks by quantum (and classical) algorithms, is a critical area of research. Towards this goal, a number of standard approaches have been explored extensively, including lattice-based cryptography, code-based cryptography, and isogeny-based cryptography. Some of these have already been selected for standardization by NIST.

Nevertheless, from a long term perspective, it is interesting and important to sustain research on alternative mathematical structures, algorithms, and one-way functions for applications in post-quantum public-key cryptography. Several different frameworks have been conceived and investigated in this direction, using algebraic objects such as semigroups, nonabelian groups, semirings, rings, semigroup actions, group algebras and modifications thereof. For these proposals to gain legitimate consideration for real-world cryptography, extensive research is required on their efficiency and security properties. This thesis is an attempt in this direction.

The first prefatory chapter of this thesis provides a brief introduction to some key concepts in cryptography, and thereby sets the stage for the rest of the chapters. In Chapter 2, we study the complexity of the discrete logarithm problem in different platforms, particularly in semigroups, where inverses are not necessarily available, so the typical collision-based algorithms fail. In Chapter 3, we study the complexity of the conjugacy search problem, which has repeatedly been proposed as a one-way function, in some potential nonabelian groups. In Chapter 4, we provide a cryptanalysis of a key exchange system based on a form of modified group algebras. Finally, Chapter 5 studies some deterministic methods of producing collisions in generalizations of some well-known group-based hash functions.

Acknowledgements

First and foremost, I extend my deep gratitude to my PhD advisor, Prof. Joachim Rosenthal, for giving me the opportunity to pursue this doctorate, and for trusting my ability to succeed in this colossal endeavour. Joachim has been an immensely supportive mentor, whose knowledge, insight and experience have been pivotal to guiding the course of my PhD. He has given me utmost academic freedom to explore my curiosity and creativity, and to work on topics I have found most engrossing.

Apart from his academic mentorshop, Joachim's moral support has been vital to my success in completing this thesis. His words of encouragement have greatly shaped my confidence in doing research, and my fortitude during the times of self-doubt that are familiar to almost every PhD student. I greatly appreciate his equal willingness to discuss and provide advice on non-academic concerns, from career matters and legal and administrative issues, to life in Zürich and ideas for weekend trips. I have learnt a great amount, not only about mathematics, but also history and politics, from the many lunches and coffee breaks with him. Joachim's optimism, enthusiasm, kindness, and cheerful attitude have lightened up countless of my days.

I also extend heartfelt thanks to my second advisor, Dr. Carlo Matteotti, who has consistently taken keen interest in my research, provided helpful feedback, and motivated me to achieve my best. I have always looked forward to the insights that come from research discussions with Carlo and Joachim, and have felt all the more inspired after them. Carlo's presence, his rigorous proofreading of, and feedback towards my work have been immensely uplifting, and I am grateful to have had such a kind, hard-working and encouraging person as a mentor during my PhD.

I also acknowledge and thank the Cyber-Defence Campus for their financial support towards my PhD, and the other opportunities for engagement, interaction, learning, and sharing my research that they offered me.

I further thank the thesis evaluation referees for their careful review of this thesis. I am also grateful to my PhD committee members, Prof. Kenny Patterson and Prof. Andrew Kresch, who patiently sat through all my committee meetings and gave me advice and input on my progress. I extend special thanks to Kenny, whose engaging research in practical cryptography, and willingness for discussions thereof, have been a source of inspiration for me throughout.

While talking of mentors, I cannot forget to appreciate the role of the people who instilled in me the love for mathematics in my early stages as a student. I express my heartfelt indebtedness to my first ever mentor in mathematics research, late Prof.

I.B.S Passi, to whom I will forever credit my passion for algebra. Prof. Passi, who unfortunately passed away in 2021, was for years an invaluable source of knowledge, inspiration, and motivation. I also extend thanks to my master thesis advisor Prof. Kapil Paranjape and mentor Dr. Amit Kulshreshtha, who have advised and guided me in terms of both academic and career matters, and believed in my capabilities throughout.

This acknowledgement would be dreadfully meaningless without reference to my most paramount mentors and best friends since birth, the primary pillars of my strength and hope, to the two people to whose credit I ascribe all that I am today and all that I am capable of becoming. It is hard to express in words my gratitude to my parents, whose unconditional support and love throughout the years have shaped my person and helped me live up to the best of my abilities. I am privileged to have been brought up with their emphasis on high education, critical thinking, and scientific temper, and their judgement-free inspiration to explore my own creativity and interests. They have supported me, appreciated and taken active interest in my endeavours, dedicated selfless amounts of time and energy to my needs, believed in me unvaryingly, and encouraged me tirelessly. To them, I dedicate this thesis.

I am also thankful to the members of my research group, who have become dear friends along the way. I have shared countless discussions, laughs, gossips and memories with Julia, Niklas, Zita, Beatricia, Silvia, and Michael, Abhinaba, and former members of "the coolest office ever". My initial academic elder siblings, Gianira, Karan, Violetta, and Julia, greatly helped me find comfort, both academically and personally, when I initially moved to Zürich. I thank all of them for being the amazing people that they are. I also thank all my other friends, particularly: Bahar, whose unconditional support and companionship have been pivotal in my life; Sai Praveen, my favourite fellow mathematician and a loyal confidant for almost a decade; and Amruta, who first hosted me in Zürich during my PhD interview and has helped me out on numerous occasions; and Soheb, who has motivated me to aim high, and yet also taught me the importance of slowing down.

I also thank all the other members of the department for their cooperation, friendliness and warmth, which have made ours a pleasant workplace. With special gratitude, I acknowledge all the staff members whose hard work and dedication keep the institute running smoothly. Of special mention are the IT Whiz Carsten, the secretaries Grit, Franziska and Bettina, and the graduate school coordinator Jessica, who have been ever-ready to help out in times of technical and administrative issues.

न चोरहार्यं न राजहार्यं
न भ्रतृभाज्यं न च भारकारि।
व्यये कृते वर्धति एव नित्यं
विद्याधनं सर्वधनप्रधानम्॥

Which cannot be stolen by thieves, nor taken away by kings
nor split among brothers; nor is it a burden upon the shoulders.
When spent, it grows continuously.
Indeed, the wealth of knowledge is the most superior among all riches.

Contents

Chapter 1

Introduction

Cryptography is the science and art of securing communication from unauthorized access by transforming information into a form that is unintelligible to an unapproved entity. Modern cryptography is a highly consequential field of study and practice, and a critical component of the rapid digitalization process we currently witness, since it facilitates secure data storage, message exchange, financial transactions, and military communication across insecure channels such as the internet. It plays a crucial role in each of the aspects of the now well-known CIA triad model of information security:

1. Confidentiality: Information is kept secret from unauthorized parties.
2. Integrity: Messages are not modified in transit.
3. Authentication: The sender of a message is as claimed.

While the formal study of cryptography is fairly recent, the subject has a rich and intriguing history. The first section of this chapter provides a brief summary of a selection of historical events that have shaped cryptography in its present form.

1.1 History

The generation, storage and transport of written information together form one of the key defining characteristics of human civilization. Imaginably, the need to protect the confidentiality of some of this information would have arisen shortly after, if not in parallel, to the phenomenon of its creation and storage. It is therefore not surprising that cryptography has been around for thousands of years in a vast range of different forms.

The use of ciphers dates back at least to 1900 BCE in Ancient Egypt, where hieroglyphs were used to encode messages on tombs and monuments. The Spartans of Ancient Greece (c. 400 BCE) are known to have used a transposition cipher to protect military communications. The Caeser Cipher, one of the simplest and most widely known encryption techniques was used by Julius Caeser (100-44 BCE) for military purposes during the Roman times.

A variety of different ciphers have been developed across history, many emerging to enhance the chances of preserving security. In a substitution cipher, each letter of the message is replaced by another letter or symbol. These are the oldest known ciphers, of which the shift Caeser cipher is an example. In transposition ciphers, the order of the letters of the alphabet is permuted to generate the secret text.

Various methods of steganography, i.e. the practice of hiding a secret message inside of something that is not secret, were developed during the Middle Ages (c. 1000-1500 CE), including invisible ink and messages hidden in wax tablets. Polyalphabetic ciphers, under which multiple secret alphabets are used to encode a message according to some key, are believed to have been developed during the Renaissance (c. 1500-1600 CE).

Structured methods for cryptanalysis, i.e. the decoding of secret messages by an unintended, unauthorized entity (without prior access to the key), seem to have emerged later on. In the Arab world of c. 800 CE, one finds the development of the cryptanalytic technique of frequency analysis, which involves analyzing the frequency of letters in a message to break a substitution cipher.

The use of ciphers and cryptanalysis in political and military applications became increasingly pervasive after the widespread deployment of long-range communication methods during the eighteenth, nineteenth, and early twentieth centuries. The British interception and decoding of the secret Zimmermann Telegram sent by Germany to Mexico during World War I was responsible for propelling the earlier neutral United States into the war.

During World War II, Germany used its machine cipher, the Enigma machine, for secret communication. This was a device containing a series of spinning rotors whose paths generated a complicated polyalphabetic cipher. The British, aided by Polish cryptographers, deciphered a large number of messages encrypted on Enigma machines. This played a crucial role in the victory of the Allies in the war.

Up to this point, every known form of cryptography employed used secret predecided keys. The seminal development of public-key systems pioneered by Diffie and Hellman [27] in 1976, combined with the coming of the digital age, has led to a massive overhaul of the field of cryptography and an explosion in applications in secure online communication, financial transactions, and military communication. Modern cryptography comprises a vast and rich field of study and practice, whose significance continues to expand rapidly in parallel with our progress into the information age.

1.2 Mathematics and Cryptography

Cryptography comprises an expansive and deep interdisciplinary subject, drawing from the fields of mathematics, computer science and engineering. Mathematical theory constitutes the foundation for the majority of constructions and algorithms in cryptography, particularly those found in public-key cryptography. Concepts from algebra, number theory and algebraic geometry lie at the core of mainstream cryptographic protocols, as well as methods for cryptanalysis. Concepts from probability theory and statistics are used to analyse the security of cryptographic systems.

One-way trapdoor functions, which are defined formally in Section 1.5, are derived from hard mathematical problems whose solutions are established through years of mathematics research to be computationally too expensive to be implemented without access to the trapdoor information. In order to establish a cryptosystem from a one-way trapdoor function, one needs, in addition, a suitable structural and algorithmic framework, an efficient representation of information and an efficient method for computing with the one-way function.

Several of the structural frameworks, methods, algorithms and one-way functions used in public-key cryptography, as well as methods for cryptanalysis and security verification come from algebra, number theory and algebraic geometry. Mathematical one-way functions arising from these disciplines form the basis of today's most used public-key cryptosystems, the Diffie-Hellman and RSA protocols, which are disucussed in more detail in Section 1.5. Algebraic structures such as finite fields and elliptic curve groups are used as the base structures for these systems, allowing efficient arithmetic on data and keys.

1.2.1 New one-way functions and post-quantum cryptography

The one-way functions underlying the Diffie-Hellman and RSA protocols are, respectively, the discrete logarithm problem and the factoring of large integers. While the Diffie-Hellman and RSA protocols have shown consistent resilience to classical attacks, it was shown in [90] that there exists an efficient (polynomial time) attack on these systems using a quantum algorithm. The dire consequence of this groundbreaking result is that many present-day public-key cryptosystems would be broken when a quantum computer with sufficient computational power is invented. For this reason, post-quantum cryptography, viz. the study of (public-key) cryptosystems that are resilient to attacks by quantum (as well as classical) algorithms, is a critical area of research.

Towards this goal, a number of approaches have been explored, including lattice-

based cryptography, code-based cryptography, multivariate cryptography, and isogeny-based cryptography. In 2016, NIST (National Institute of Standards and Technology) launched the Post-Quantum Cryptography Standardization program to standardize potentially quantum-secure cryptographic primitives. As of the last official update in 2022, lattice-based key exchange, encryption, and signature schemes have been selected for standardization, while some code-based and isogeny-based alternatives are also still being considered.

1.3 Computational Complexity

Before we delve in to the mathematical theory of cryptography, we first quickly elucidate some important terminology in complexity theory. This is important to clarify the context to the reader when references to the efficiency of algorithms is made later on.

Definition 1.1 (Order Notation). Let $f(x)$ and $g(x)$ be positive-valued functions of x. We say that "f is big-$\mathcal{O}$ of g" and write $f(x) = \mathcal{O}(g(x))$ if there are positive constants c and C such that $f(x) \leq cg(x) \; \forall \; x \geq C$.

When talking about the complexity of an algorithm, we are interested in knowing the number of steps (time) and sometimes, the storage space required for the completion of the algorithm. These are both typically calculated relative to the size of the input to the algorithm, usually measured in bits.

An algorithm is said to be polynomial time (or to have polynomial complexity) if for an input size of $\mathcal{O}(k)$ bits, there is a constant $c \geq 0$ such that the algorithm takes $\mathcal{O}(k^c)$ steps. On the other hand, if for inputs of size $\mathcal{O}(k)$ bits, there is a constant $c > 0$ independent of k such that the algorithm takes $\mathcal{O}(e^{ck})$ steps, then the running time is exponential. An algorithm has subexponential complexity if for input size k, its running time is $\mathcal{O}(2^{k^\epsilon})$ for some $\epsilon > 0$. In the context of cryptography, an algorithm referred to as efficient or fast is typically polynomial time, whereas exponential time algorithms are normally infeasible to execute.

A problem is said to be polynomial (resp. exponential) time if the best known algorithm for its solution is polynomial (resp. exponential) time. A problem is said to be subexponential time if for for an input size of k bits, for every $\epsilon > 0$ there exists an algorithm which solves the problem in time $O(2^{k^\epsilon})$. In other words, the running time grows faster than any polynomial function, but slower than any exponential function.

Having described this terminology, we are now ready to explore the theory of mathematical cryptography in more depth.

1.4 Private Key Cryptography

Private-key cryptography, also known as secret-key or symmetric-key cryptography, is the branch of cryptography under which the communicating parties encrypt and decrypt their messages using a single shared secret key that the adversary does not possess. A prerequisite for this is that the communicating parties need to have a secure medium to exchange the secret key beforehand.

Denote by M, K and C the sets of plaintext messages, keys, and ciphertext messages, respectively. Encryption and decryption may be viewed respectively as functions e, d

$$e : K \times M \to C$$

$$d : K \times C \to M$$

such that for all $k \in K,\ m \in M$, $d(k, e(k, m)) = m$. It is in general assumed that an adversary knows the method for encryption and decryption, i.e. the functions e and d. The lack of knowledge on the key $k \in K$ used is what prevents the adversary from decrypting a message.

For (K, M, C, e, d) to be a successful cipher, it must satisfy the following properties:

1. Efficient evaluation: For a key $k \in K$ and plaintext $m \in M$, it must be easy to compute the ciphertext $c = e(k, m)$ and the plaintext $d(k, c)$.
2. Infeasible illegitimate decryption: Given ciphertexts

   $$c_1 = e(k, m_1), c_2 = e(k, m_2), \ldots, c_n = e(k, m_n)$$

   encrypted using the key $k \in K$, it must be very difficult to compute any of the corresponding plaintexts

   $$m_1 = d(k, c_1), \ldots, m_n = d(k, c_n)$$

   without knowledge of k.

Some other desirable properties typically demanded in practice are:

3. Known Plaintext Security: Given ciphertexts

   $$c_1 = e(k, m_1), c_2 = e(k, m_2), \ldots, c_n = e(k, m_n) \in C$$

   encrypted using the key $k \in K$ along with their corresponding plaintexts

   $$m_1, \ldots, m_n,$$

   it must be infeasible to decrypt any ciphertext $c = e(k, m)$, $c \neq c_i\ \forall i, 1 \leq i \leq n$ that is not in the given list, without knowing k.

4. Chosen Plaintext Security: An attacker who is allowed to use an oracle to decrypt any chosen set of ciphertexts

$$c_1 = e(k, m_1), c_2 = e(k, m_2), \ldots, c_n = e(k, m_n) \in C,$$

is unable to decrypt a ciphertext $c = e(k, m)$, $c \neq c_i$ $\forall i$, $1 \leq i \leq n$ not in the given list, without knowing k.

Some popular private-key ciphers are Data Encryption Standard (DES), Advanced Encryption Standard (AES), Triple DES, Blowfish, and RC4. The security of most efficient modern private key cryptosystems relies on repeated application of various mixing operations that are combined in a way that they are hard to reverse without the private key.

As mentioned before, private key cryptography cannot be used by itself for information exchange between people who have not exchanged a key beforehand. However, private key cryptosystems are used extensively for internet communications in conjunction with public key cryptosystems. In fact, since private key cryptography tends to be significantly more efficient than public key cryptography, it is typically the preferred method for message encryption, while a public key cryptosystem is used a priori for key exchange. In the main chapters of this thesis, we are concerned exclusively with public key systems.

1.5 Public Key Cryptography

The revolutionary 1976 paper of Diffie and Hellman [27] first introduced the idea of public-key cryptography to the mainstream. It was the first public demonstration of the exchange of confidential information between two people who can communicate only via a channel that is being monitored by an adversary. In a public key cryptosystem, the communicating parties, call them Alice and Bob, each have two keys, one kept private and the other published publicly.

Using a key exchange scheme, Alice and Bob can communicate in public, yet establish a shared, secret key which is known only to the both of them. In an encryption scheme, the public key of Alice can also be used by anyone in the world to encrypt messages and send them to Alice, but only she can decrypt them with her private decryption key. Similarly, a digital signature scheme allows Alice to use her keys to attach proof of her identity to her messages, allowing the receiver to validate the source and integrity of data.

A paramount notion in cryptography is that of a one-way trapdoor function. A one-way function, put loosely, is a mathematical function that can easily and efficiently

be calculated in one direction, but is difficult or computationally infeasible to invert. The existence of a true one-way function has not been proven, but many functions have been proposed to be one-way, and are used as such with this assumption.

Moreover, if there exists a piece of additional information, called the trapdoor information, which allows an efficient inversion, the function is called a one-way trapdoor function. One-way trapdoor functions are used extensively to conceal information in public-key cryptographic systems, to ensure that an adversary cannot invert the function and thereby decrypt the message, whereas the intended receiver (who has the trapdoor information) can easily do so.

We will see in this section that one-way functions and one-way trapdoor functions play a key role in constructing public-key protocols. We begin by describing the Diffie-Hellman key exchange protocol.

1.5.1 Discrete log-based encyrption

Protocol 1.1 (Diffie-Hellman Key Exchange).

1. *The two parties, Alice and Bob, agree on a large prime number, p and an integer g having large prime order in $\mathbb{F}_p^*$. Typically, g is chosen to be a primitive root modulo p.*
2. *Alice chooses a secret integer a and computes $A = g^a \pmod{p}$. She sends A to Bob. Her secret key is a, and her public key is A.*
3. *Bob chooses a secret integer b and computes $B = g^b \pmod{p}$. He sends B to Alice. His secret key is b, and his public key is B.*
4. *Alice computes her shared secret key, $K_A = B^a \pmod{p}$.*
5. *Bob computes his shared secret key, $K_B = A^b \pmod{p}$.*

Note that $K_A = B^a = (g^b)^a = g^{ba} = g^{ab} = (g^a)^b = A^b = K_B \pmod{p}$. Thus, both Alice and Bob end up with the same shared secret key $K = K_A = K_B$, which can then be used for symmetric encryption.

We note that the Diffie-Hellman key exchange protocol is efficient to implement: the exponentiation operation g^x in a finite group can be done with the square-and-multiply method, which takes a small multiple of $\log_2(x)$ group multiplications to compute.

The security of the Diffie-Hellman key exchange relies on the computational difficulty of the discrete logarithm problem in a finite field $\mathbb{F}_p$, which is presumed to be a one-way function for large primes p. We state this problem in its general form below.

Definition 1.2 (Discrete Logarithm Problem (DLP)). Let G be a finite cyclic group with generator g and let h be an element of G. The discrete logarithm problem is the problem of finding an exponent x such that $g^x = h$ in G. The number x (computed modulo the order of G) is called the discrete logarithm of h to the base g and is denoted by $\log_g(h)$.

To find one of the private keys a or b in the Diffie-Hellman key exchange, the adversary needs to solve the discrete logarithm problem with base g in the group $G = \mathbb{F}_p^*$. However, the difficulty of obtaining the shared key is equivalent to the following algorithmic problem.

Definition 1.3 (Diffie-Hellman Problem (DHP)). Let G be a finite cyclic group with generator g let $h_1 = g^{n_1}$ and $h_2 = g^{n_2}$ be elements of G, provided so that the values of the exponents n_1 and n_2 are concealed. The Diffie-Hellman problem requires finding the element $g^{n_1 n_2}$ in G.

It is clear that a solution to the DLP also gives a solution to the DHP, so the DHP is at most as difficult as the DLP. Whether the converse holds is, however, not known. While it is believed that the Diffie-Hellman and Discrete Logarithm Problems are equivalent, a proof is not known.

In their original paper [27], Diffie and Hellman use the platform group $G = \mathbb{F}_p^*$ for key exchange. However, it later turned out that using a group $G = E(\mathbb{F}_p)$ of points on an elliptic curve over a finite fields provides better security. Arithmetic in the groups $\mathbb{F}_p^*$ and $E(\mathbb{F}_p)$ is fast, so the protocol is efficiently implementable in both. The best known general algorithm to solve the DLP in $\mathbb{F}_p^*$ for any p is subexponential, while in a suitably chosen elliptic curve group, it is exponential, taking $\mathcal{O}(\sqrt{|G|})$ steps, which has been shown to be optimal. The elliptic curve Diffie-Hellman is currently the state-of-the-art algorithm for classical key exchange.

We note that the Diffie-Hellman protocol can be used only for exchanging keys, but requires additional apparatus if it is to be used for message encryption or authentication. In 1985, Taher ElGamal showed the construction of a public-key encryption cryptosystem based on the discrete logarithm problem, building on the ideas of Diffie and Hellman.

Protocol 1.2 (**ElGamal Encryption Scheme**).

1. *The two parties, Alice and Bob, agree on a large prime number, p and and a primitive root g* $\mod p$.

2. *Alice chooses a secret integer a and computes $A = g^a \pmod p$. She sends A to Bob. Her secret key is a, and her public key is A.*

3. *Bob chooses a random secret integer b and computes his public key $B = g^b \pmod p$ and the shared secret key $K = A^b \pmod p$.*

4. *Encryption: To encrypt his message m, Bob computes the ciphertext $C = m \cdot K \pmod p$ and sends the pair (B, C) to Alice.*

5. *Decryption: Alice computes the shared key $K = B^a \pmod p$ and the plaintext message $m = C \cdot K^{-1} \pmod p$.*

It can be shown easily that the difficulty of decrypting the message for an adversary is equivalent to the difficulty of solving the Diffie-Hellman problem. The security of the ElGamal system is therefore based on the difficulty of the discrete logarithm problem.

1.5.2 Algorithms for the discrete logarithm problem

As discussed, the security of the Diffie-Hellman protocol relies on the difficulty of solving the discrete logarithm problem in the group $G = \mathbb{F}_p^*$. For general groups, brute force provides an obvious theoretical solution. One can compute the list of values g, g^2, $g^3, \ldots$ until equality with h is reached. If g has order n, then this algorithm is guaranteed to find the solution in at most n exponentiation steps. For sufficiently large n, this is not practically feasible with the computing power available today. At present, it is recommended that the prime p is at least 2048 bits long.

Another approach for a general group is collision-based algorithms, which rely on finding matches between lists of elements, and using these to solve the DLP. Below, we describe one such algorithm, Shanks' Baby-Step Giant-Step Algorithm [89]. Let G be a group and let $g \in G$ be an element of order $N \geq 2$.

Algorithm 1.1: Shanks' Baby-Step Giant-Step Algorithm

1: Set $n = 1 + \left\lfloor \sqrt{N} \right\rfloor$.
2: Create two lists, $L_1 = \{1, g, g^2, g^3, \ldots, g^n\}$,
$L_2 = \{h, hg^{-n}, hg^{-2n}, hg^{-3n}, \ldots, hg^{-n^2}\}$
3: Find a match between the two lists, say $g^i = hg^{-jn}$.
4: Return $x = i + jn$. Clearly, x is a solution to $g^x = h$.

This algorithm solves the discrete logarithm problem $g^x = h$ in $\mathcal{O}(\sqrt{N} \log N)$ steps using storage size of $O(\sqrt{N})$. Clearly, although both are exponential time, this approach is far more efficient than the brute force algorithm. Pollard's rho algorithm [81] is another collision-based method, with the same $\mathcal{O}(\sqrt{N} \log N)$ time complexity, but with no storage space requirement. It works by iteratively generating a sequence

of group elements, and eventually detects a cycle in this sequence, which it then uses to solve the discrete logarithm problem.

1.5.3 Factoring-based encryption

We now describe the RSA public-key cryptosystem [84], which, like ElGamal, is used for secure communication and encryption. It is named after its inventors, Ron Rivest, Adi Shamir, and Leonard Adleman. The RSA cryptosystem has been widely used for secure communication, and remains one of the most popular public-key cryptosystems in use today.

Protocol 1.3 (RSA Encryption Scheme).

1. *Key generation: Alice selects two large secret primes p and q and computes $N = pq$. She chooses encryption exponent e with $\gcd(e, (p-1)(q-1)) = 1$ and publishes (N, e) as the public key.*

2. *Encryption: Bob selects his plaintext m and computes $c = m^e \pmod N$ using Alice's public key. He sends c to Alice.*

3. *Alice computes d satisfying $ed = 1 \pmod{(p-1)(q-1)}$. She can do this in polynomial time using the Euclidean algorithm, since she knows $(p-1)(q-1)$ from her knowledge of p and q.*

4. *Decryption: Alice computes $m' = c^d \pmod N$.*

We have $m' = c^d \pmod N = (m^e)^d \pmod N = m^{ed} \pmod N = m \pmod N$, where the last step follows from the fact that $|(\mathbb{Z}/n\mathbb{Z})^*| = \phi(N) = (p-1)(q-1)$, so since $m \in (\mathbb{Z}/n\mathbb{Z})^*$ and $ed = 1 \pmod{\phi(N)}$, we must have $m^{ed} = m \pmod N$. Therefore, Alice recovers the correct message sent by Bob.

Recall that the security of the Diffie–Hellman and the ElGamal systems relies on the difficulty of solving equations of the form $a^x = b \pmod p$ for x unknown. On the other hand, the RSA algorithm relies on the principle that solving an equation $x^e = c \pmod N$ for x is easy if one knows the prime factorization of N, and difficult otherwise. When N is a prime, it can be done using the Euclidean algorithm, which computes the inverse of e modulo N. The only known way for an attacker to determine the private key from the public key is to factor N into its prime factors. The security of this cryptosystem thus relies on the difficulty of factoring large integers. The function $x \to x^e \pmod N$ is a one-way trapdoor function, with the trapdoor information being the factorization of N.

1.5.4 Algorithms for Factoring

There is clearly a straightforward brute force attack, in which the adversary tries every prime below $\lceil\sqrt{N}\rceil$ to find a factorization. To avoid brute force attacks on RSA, it is necessary for the primes p and q to be large enough. At present, p and q are recommended to be no shorter than 2048 bits. There are a number of algorithms for the factorization of integers, which therefore also naturally extend to attacks on the RSA scheme. Pollard's $p-1$ method [81] is a probabalistic algorithm which uses a variation of Floyd's algorithm [31] for finding cycles in sequences. It works efficiently for a certain subset of numbers, demonstrating the existence of insecure RSA moduli which otherwise appear to be secure.

The quadratic sieve algorithm [82] is one of the most powerful factorization methods known today, and has been used to factor several RSA challenge numbers. It fixes a smoothness bound B and a factor base of prime numbers less than B, and searches for numbers x such that $x^2 - n$ is smooth over the factor base. The process involves two fundamental steps: sieving, which eliminates multiples from a fixed set of integers, and linear algebra, which solves a set of congruence equations. These solutions are then combined to form a factorization of the original composite number.

The number field sieve [55] is a generalization of the quadratic sieve algorithm (QS) which allows for more complex factorizations. It is the quickest known algorithm for factoring integers larger than a few hundred digits, and is a highly consequential algorithm in both cryptography and computational number theory. It can also be used to solve the discrete logarithm problem.

1.5.5 Digital Signatures

A digital signature is a mathematical algorithm used to verify the authenticity and integrity of electronic documents or messages. Digital signatures are commonly used in electronic transactions, such as online banking or e-commerce, to ensure that sensitive information is transmitted securely and that the sender's identity is verified.

They can also be used to provide non-repudiation, which means that the sender cannot deny having sent the message or document once it has been signed with their private key. This protects both communicating parties from being deceived by each other by attaching to each message a proof of its origin. This property is impossible to achieve in a symmetric-key setting in which both Alice and Bob use the same secret key and thus have the same capabilities.

Under a digital signature protocol, the sender generates a signature from the message and the private key, using a signature algorithm. Typically, a signature algorithm

works in two steps: by calculating a hash of the message, and then encrypting it using the private key. The signature can be verified using the sender's public key. The primary security property required from a signature scheme is unforgeability, or "existential unforgeability", which means that a party without access to the private key is not able to forge a valid signature on a message.

Each of the three popular public-key algorithm families, namely integer factorization, discrete logarithms and elliptic curves, allows us to construct digital signatures.

Protocol 1.4 (RSA Signature Scheme).

1. *Key generation: Alice selects large primes p and q and computes $n = pq$ and an integer e such that $1 < e < (p-1)(q-1)$ and $\gcd(e, (p-1)(q-1)) = 1$. She computes an integer d such that $de = 1 \pmod{(p-1)(q-1)}$. The pair (n, e) is her public key, and d is her private key. The hash function h is also published publicly.*

2. *Signature Generation: Alice first computes a hash value $h(M)$ of her message M and then computes the signature $s(M)$ as $s(M) = h(M)^d \pmod{n}$. She sends the pair $(M, s(M))$ to Bob. Alternatively, she may also choose to encrypt M before sending it.*

3. *Signature Verification: On receiving the pair $(M, s(M))$, Bob computes the hash value $h(M)$ and $v = s(M)^e \pmod{n}$. If $v = h(M)$, then Bob knows that the message has not been tampered with and that it was indeed sent by Alice.*

As in the RSA encryption scheme, an attacker who can factorize n can compute the decryption exponent d, and thereby forge signatures on any message and impersonate Alice. However, there exists another attack on the scheme in the form described above, if the message is not hashed before it is signed. In this case, note that if an attacker Eve starts with any number $s \in (\mathbb{Z}/n\mathbb{Z})^*$, she can compute $y = s^e$ and generate a valid message-signature pair (y, s). Bob, on receiving this pair, will verify the signature and assume it comes from Alice.

Even though this attack requires Eve to choose the signature first, and therefore allows no control on the semantics of the message, it is unacceptable that the process verifies forged signatures. For this reason, RSA signature is rarely used in this form, particularly without pre-hashing the messages. In practice, only certain message formats are allowed (through a process called padding), allowing the verifier to distinguish between valid and invalid messages. The probabilistic signature scheme RSA-PSS is an extension of RSA which is used in practice.

The ElGamal signature scheme is a digital signature scheme based on the discrete logarithm problem. It consists of two main algorithms: a key generation algorithm

and a signature generation and verification algorithm.

Protocol 1.5 (**ElGamal Signature Scheme).**

1. *Key generation: Alice chooses a large prime p, a primitive root $\alpha pmodp$, and a secret key $x \in \{1, \ldots, p-2\}$. She computes $\beta = \alpha^x \pmod p$. Her public key is (p, α, β) and her private key is x.*

2. *Signature generation: Alice chooses an ephemeral secret key $k \in \{1, \ldots, p-2\}$ and computes $r = \alpha^k \pmod p$. She chooses her message m and computes the hash $h = \text{hash}(m)$. Then, she computes $s = (h - xr)k^{-1} \pmod{p-1}$. Her signature is (r, s).*

3. *Signature verification: Bob, on receiving the message m and (r, s), computes $h = \text{hash}(m)$, $u_1 = hs^{-1} \pmod p$, $u_2 = rs^{-1} \pmod p$, and $v = \alpha^{u_1}\beta^{-u_2} \pmod p$. The signature is treated as valid if and only if $v = r$.*

The security of the ElGamal signature scheme relies on the difficulty of discrete logarithm problem in $\mathbb{F}_p^*$. An attacker who can solve the DLP can compute both the private key d from β and the ephemeral key k from r, and use these to sign arbitrary messages on behalf of the signer. It can also be shown that the reuse of the ephemeral key k to sign two different messages allows an attacker to easily compute both the private keys d and k and therefore to freely forge signatures. As for the RSA signature scheme, there is also an existential forgery attack on this scheme, under which an attacker can select the signature first, and then generate its corresponding message.

The ElGamal signature algorithm is rarely used in practice in the form above. The Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA) are used for practical applications, due to shorter signature lengths and better security properties. We only state the DSA here.

Protocol 1.6 (**Digital Signature Algorithm).**

1. *A trusted party chooses and publishes large primes p and q satisfying $p = 1 \pmod q$ and an element g of order q modulo p.*

2. *Key generation: Alice chooses secret signing key $a \in \{1, \ldots, q-1\}$ and computes $A = g^a \pmod p$. She publishes the verification key A.*

3. *Signature generation: Alice computes the hash of her message m, $h = \text{hash}(m)$ as an element of $\mathbb{F}_q$. She chooses an ephemeral key $k \in \{2, \ldots, q-1\}$. She computes signature $r = g^k \pmod p \pmod q$ and $s = (h + ar)k^{-1} \pmod q$.*

For the DSA to be secure, one needs to protect against two different discrete logarithm attacks in $\mathbb{F}_p^*$ and $\mathbb{F}_q^*$. The reuse of the ephemeral key in this case also leads

to a complete breakdown, so it must be ensured that the ephemeral key is used only once and then discarded.

1.6 Hash Functions

Let $\mathcal{A}$ be an alphabet and $\mathcal{A}^*$ denote the set of all finite-length words in $\mathcal{A}$ and $\mathcal{A}^n$ denote the set of all words up to length n in $\mathcal{A}$. A length n hash function, or compression function, is a map $\mathcal{A}^* \to \mathcal{A}^n$ which takes messages of arbitrary length to fixed-length message digests. A hash function $h : \mathcal{A}^* \to \mathcal{A}^n$ is called a cryptographic hash function if it satisfies the following properties:

- Collision-resistance: it is computationally infeasible to find a pair (x, x') of distinct messages such that $h(x) = h(x')$. Such a pair (x, x') is said to produce a collision of h.
- Second pre-image resistance: given a message x, it is computationally infeasible to find another message $x \neq x'$ such that $h(x) = h(x')$.
- One-wayness: given a hash value $y \in \mathcal{A}^n$ it is computationally infeasible to find a pre-image x such that $h(x) = y$.

Hash functions are also often required to exhibit the avalanche effect, under which a small modification in the message text causes a big change in the hash value. This prevents the hash value from revealing any information about the message string, and ensures no visible correlation between the hash values of related strings.

1.6.1 SHA

The Secure Hash Algorithms (SHA), specified in FIPS (U.S. Federal Information Processing Standard) 180-4, comprises a family of cryptographic hash functions designed by the US National Security Agency (NSA) and standardized by NIST [71]. These algorithms are based on block ciphers and comprise of various data preprocessing and processing transformations such as breaking data into chunks, padding, circular shift operations, bitwise operations such as logical AND, OR and XOR, and recombining processed data through concatenation.

Within this family, the SHA-1, SHA-2, and SHA-3 families were successively designed with increasingly stronger encryption in response to hacker attacks. There have been successful attacks on the collision-resistance of SHA-1 (which produces 160-bit digests), through brute force efforts. For this reason, the length of message digests has been increased to 224- or 256-bit digests in SHA-2. SHA-0 and SHA-1 are both now obsolete, while the SHA-2 hash function is implemented in some widely

used security applications and internet protocols. The interested reader is referred to the reports [69, 70, 71, 72] for more information on the SHA family.

1.6.2 Applications in cryptography

Hash functions have a number of applications. They are often used in password storage, so that an attacker with access to the password database can only see the hashed value and not the actual password, and therefore, cannot hack the users' accounts. This also renders this process more efficient, since it is no longer necessary to store arbitrary-length passwords.

Hash functions are also used for verifying the integrity of files transmitted over the internet. The receiver of the file may download both the file and its hash, recompute its hash value and compare it to the hash received. The equality of these two values shows that the message has not been tampered with, even in otherwise unnoticeable ways. As seen in Section 1.5.5, cryptographic hash functions are also used as components of digital signatures. Hash functions are also used in the construction of MACs (Message Authentication Code), for which they are combined with a secret key prior to application to the message. MACs are used to ensure the integrity of a message and to verify that it was sent by an authorized sender.

1.6.3 Security

There are several types of attacks on hash functions. Without detailed descriptions, we list some of these below.

1. Collision Attacks: the attacker tries to find two different inputs that result in the same hash output, thereby trying to violate collision-resistance. For instance, the attacker may hash roughly $2^{n/2}$ inputs for a hash function of length n, and exploit the birthday paradox, which implies that two of these inputs that have the same hash value with a probability of 50 percent.

2. Preimage Attacks: the attacker tries to find an input that hashes to a fixed output thereby trying to violate preimage-resistance. This allows them to substitute the malicious message for a legitimate one, and disrupt the integrity. If attempted by brute force for a hash function producing an n-bit output, one can find preimages for in approximately 2^n evaluations.

3. Length Extension Attacks: the attacker targets hash functions with a built-in secret key or seed value. Given the hash output of a message, the attacker constructs a new message by appending additional data to the original message and generates a valid hash output without knowing the secret key or seed value.

4. Side-Channel Attacks: the attacker exploits weaknesses in the implementation of the hash function, rather than the hash function itself.

1.7 Cryptographic protocols over the internet

Secure communication over public networks like the internet is facilitated by a coalescence of private-key encryption systems, public-key systems for key exchange and signatures, and cryptographic hash functions. Due to variation in functionality, efficiency and security properties, it is only in a careful sync that these different primitives may achieve the full power of the internet protocols in current use. We list a few of the most significant internet protocols for secure communication below.

- Transport Layer Security (TLS): provides security for internet communication by encrypting data and verifying the identity of the communicating parties.
- Secure Shell (SSH): provides secure remote access to network devices over an unsecured network
- Pretty Good Privacy (PGP): provides secure email communication by encrypting the email messages and verifying the sender's identity.
- Internet Protocol Security (IPsec): provides secure communication at the IP layer of the network by encrypting and authenticating IP packets.

1.8 Summary of Contributions in this Thesis

Current classical and post-quantum systems are thus altogether based on a relatively small number of one-way functions and mathematical structures (lattices, codes, elliptic curves, finite fields). While the hardness of breaking these systems has undergone thorough scrutiny and rigorous research, the risk of a novel, efficient attack in the future always looms. From a long term perspective, it is therefore interesting and important to sustain research on alternative mathematical structures, algorithms, and one-way functions that can be applied to public-key cryptography.

A number of different frameworks have been conceived and investigated, including, but not limited to, algebraic objects such as semigroups, non-abelian groups, semirings, rings, group algebras and modifications thereof. Various abstract and concrete algorithmic problems within these structures have been constructed or rediscovered, and proposed as one-way functions, and been used to build novel cryptosystems. For these proposals to gain legitimate consideration for real-world cryptography, extensive research is required on their efficiency and security properties. This thesis is an attempt in this direction.

In this thesis, we study various algebraic objects, and related algorithmic problems, cryptographic constructions, and cryptanalytic methods. The present prefatory chapter of the thesis is aimed at providing a brief introduction to some key concepts in cryptography, and thereby setting the stage for the rest of the chapters in the thesis. Below, we briefly summarize the chapter-wise themes and contributions.

Chapter 2

In Chapter 2, we study the complexity of the discrete logarithm problem in a semigroup, where inverses are lacking, so the typical collision-based algorithms fail.

The main contribution of this chapter is a deterministic algorithm for computing the discrete logarithm of an element y in a semigroup S with respect to some torsion base element $x \in S$. The time complexity of our algorithm is $\mathcal{O}(\sqrt{N_x}(\log N_x)^2)$, where N_x is the cycle length of x. The previously known algorithms for the discrete logarithm problem in a semigroup [10, 64] are both probabilistic and may fail with a small likelihood. We also perform an analysis of the complexity and success probability of these algorithms. Further, we use our algorithm to adapt the Pohlig-Hellman algorithm to semigroups. The material in Section 2.2 is adapted from the paper [102] co-authored by me.

Another endeavour in this chapter is to study the discrete logarithm problem in the infinite polynomial semirings $S_6[x]$ and $S_{20}[x]$, from an experimental perspective. These are infinite semigroups with non-torsion zero divisors, so in general, the discrete logarithm problem does not have a straightforward formal solution, and the algorithm mentioned above does not apply. However, we show with experiments that the discrete logarithm problem has an easy heuristic solution in this case.

In the final section of the chapter, we describe the Semigroup Action Problem and some observed limitations of the ideas in [34] for reductions extending the attack methods of Pohlig and Hellman to semigroup actions. We argue that these methods do not necessarily lead to a direct and effective attack protocol in the general case.

Some of the material in this chapter is adapted from the paper [102].

Chapter 3

In Chapter 3, we study the complexity of the conjuagacy search problem, which has repeatedly been proposed as a one-way function, in some classes of non-abelian platform groups.

The main contribution of this chapter gives a polynomial time reduction from the conjugacy search problem to (a polynomial number of) discrete logarithm prob-

lems, in two classes of groups: finite polycyclic groups with two generators, and matrix groups over finite fields. This theory is used to cryptanalyse the systems proposed in [41], [93], and [105]. We further introduce a concept called efficient C-decomposability, which we show is sufficient for the CSP in a central product of groups to reduce to individual CSP's in the components. We use this concept to demonstrate a polynomial time solution of the CSP in any extraspecial p-group.

Finally, this chapter also gives a broad overview of some of the platforms groups and computational problems employed in non-abelian group-based cryptography, some of the common protocols for key exchange and signatures, and the general and specific methods of cryptanalysis. This part serves as a survey, and contains no new results.

Some of the material in this chapter is adapted from the preprints [100, 101].

Chapter 4

In Chapter 4, we provide a full cryptanalysis of the key exchange system in [24], which is based on a modified form of group algebras. This is done by providing a classical polynomial time algebraic solution to the underlying algorithmic problem. For this, we produce an algebraic reduction of the underlying problem to a set of simultaneous equations over $\mathbb{F}_q$ involving circulant matrices. We show that in a majority of cases, they can be solved by linear algebra in polynomial time.

We also provide an argument to show that our attack algorithm has a high success rate, and verify this experimentally for the parameters proposed by the authors. We also show that despite the use of a non-commutative structure, this algorithmic problem is equivalent to a commutative semigroup action problem.

The material in this chapter has been adapted from the preprint [99].

Chapter 5

Chapter 5 focuses on devising structured, deterministic methods for producing collisions in algebraic hash functions that may be seen as generalized forms of the well-known Zémor and Tillich-Zémor hash functions.

For the generalized Zémor hash, we extend existing hash values in $SL_2(\mathbb{F}_p)$ into triangular or diagonal form by multiplying with products of the form $A_0^m A_1^n$, where A_0 and A_1 denote fixed generators of $SL_2(\mathbb{F}_q)$. We also discuss the application of this method to produce collisions, and the feasibility and efficiency thereof. Our method thus provides an alternate deterministic approach to the method for finding triangular hashes in [79].

For generalized Tillich-Zémor hash functions over $\mathbb{F}_{p^k}$ for $p \neq 2$, we relate the generator matrices to a polynomial recurrence relation, and accordingly provide conditions for collisions. We also describe a method to maliciously design the system so as to facilitate easy collisions, in terms of this polynomial recurrence relation.

On simplifying the general criteria, and through experiments, our general conclusion is that it is very difficult in practice to achieve the theoretical collision conditions efficiently, in both the generalized Zémor and the generalized Tillich-Zémor cases. Therefore, although the techniques are interesting theoretically, in practice the collision-resistance of the generalized Zémor functions is reinforced.

Closing It has been my humble strive that this project would add constructively to the theoretical and practical knowledge in the budding area of algebraic cryptography, and provide insight on future directions for research and implementation.

Chapter 2

Discrete Logarithm Problem in Various Algebraic Platforms

2.1 Introduction

Let G be a group and assume $x, y \in G$ are two elements of the group. We refer to x as the base element. The discrete logarithm problem (referred to henceforth as DLP) asks for the computation of an integer $m \in \mathbb{Z}$ (assuming such integers exist) such that $x^m = y$. The DLP plays an important role in a multitude of algebraic and number theoretic cryptographic systems. Its use was introduced in the Diffie-Hellman protocol for public key exchange [27] and has since seen a tremendous amount of development, generalisations and extensions [59].

Many modern-day systems for public key exchange use the discrete logarithm problem in a suitable group. The most commonly used groups have been the multiplicative group of finite fields and the group of points on an elliptic curve. The DLP in Jacobians of hyperelliptic curves and more general abelian varieties has also been studied extensively [20]. In [60] and [32], the DLP over the matrix group $\mathrm{GL}_n(\mathbb{F}_q)$ was studied and shown to be no more difficult than the DLP over a small extension of $\mathbb{F}_q$, and consequently less efficient in terms of key sizes for the same security level.

In this chapter, we express complexities using group/semigroup multiplications as one fundamental step. Thus, an exponentiation x^e is performed in $\mathcal{O}(\log e)$ steps. We will use the fact that for two lists of length n in which a match exists, a match can be found in $\mathcal{O}(n \log n)$ steps using standard sorting and searching algorithms (for details, the interested reader may refer to [22]).

For a general finite group of order N, there exist algorithms that solve the DLP in $\mathcal{O}(\sqrt{N} \log N)$ steps. Such algorithms are said to produce a square root attack. The most well-known examples are Shanks' Baby Step-Giant Step algorithm [89] and the Pollard-Rho algorithm [81]. Note that Shanks' algorithm is a deterministic algorithm having time complexity $\mathcal{O}(\sqrt{N} \log N)$ and space complexity $\mathcal{O}(\sqrt{N})$. In contrast, Pollard's algorithm is a probabilistic algorithm having time complexity $\mathcal{O}(\sqrt{N} \log N)$ group multiplications and space complexity $\mathcal{O}(1)$. If N is unknown, a simple modification of these algorithms would achieve a time complexity

of $\mathcal{O}(\sqrt{N}(\log N)^2)$.

Elliptic curve groups have been widely implemented in practice since for a carefully selected elliptic curve group the best known classical algorithm for solving DLP has running time $\mathcal{O}(\sqrt{N}\log N)$, where N is the group order. This is in contrast to many other finite groups such as the multiplicative group of a finite field and the group of invertible matrices over a finite field where algorithms with subexponential running time are known [2].

In cryptography, the Diffie-Hellman protocol using a finite group has been generalized to situations where the underlying problem is a discrete logarithm problem in a semigroup or even to situations where a semigroup acts on a set [49, 58]. The interested reader will find more material in a recent survey by Goel et al. [36].

Summary of Contributions

This chapter studies the difficulty of solving the discrete logarithm problem in algebraic platforms that are different from groups.

The main contribution of this chapter, Section 2.2 is a deterministic algorithm for computing the discrete logarithm of an element y in a semigroup S with respect to some torsion base element $x \in S$. The time complexity of our algorithm is $\mathcal{O}(\sqrt{N_x}(\log N_x)^2)$, where N_x is the order of x. This is the same as the time complexity for the Pollard-rho and Baby Step-Giant Step algorithms in a group, where the order N_x of the base element is unknown. The previously known algorithms for the discrete logarithm problem in a semigroup [64], [10] are both probabilistic and may fail with a small likelihood. We also perform an analysis of the complexity and success probability of these algorithms. In this section we also use our algorithm to adapt the Pohlig-Hellman algorithm to semigroups. The material in Section 2.2 is adapted from the paper [102] co-authored by me.

In Section 2.3, we use an experimental lens to study the discrete logarithm problem in the infinite polynomial semirings $S_6[x]$ and $S_{20}[x]$. These are infinite semigroups with zero divisors, so in general, the degree of a power of a polynomial cannot be calculated formally. The elements are also non-torsion, so the algorithm in Section 2.2 does not apply. However, we show that the discrete logarithm problem has an easy heuristic solution. Our experiments demonstrate that given a polynomial p and an exponent e, the degree of the resulting polynomial $q := p^e$ lies within a small range around the estimate $\lceil(\deg(q)/\deg(p))\rceil$.

In Section 2.4, we describe the Semigroup Action Problem, which was first proposed as a generalization of the DLP in [58]. In [34], the authors define reductions which may lead to a method of attacks on cryptosystems based on semigroup actions, by

extending the attack methods of Pohlig and Hellman [80] for the DLP in groups. We discuss some observed limitations of these ideas, and argue that these methods do not necessarily lead to a direct and effective attack protocol in the most general case.

2.2 The discrete logarithm problem in a semigroup

A semigroup is a set of elements with an associative binary operation. Having discussed groups, it is naturally interesting to ask whether the DLP also has a square root attack in more generalized structures such as semigroups. Since the best algorithms for the DLP all make use of the existence of inverses, it is unclear whether they can be generalized to a semigroup. However, when a special type of semigroup element, called a torsion element, is used as the base, it turns out that the DLP is reducible in polynomial time to the DLP in a finite group.

A torsion element is one whose powers eventually repeat to form a cycle, and will be defined more precisely in Section 2.2.1. This section also elaborates more on why the standard collision-based algorithms are not directly adaptable to the semigroup case. A semigroup in which every element is torsion is called a torsion semigroup. The DLP in semigroups with a torsion base element, in a classical setting, was first discussed by Chris Monico [64] in 2002, and later in a paper by Banin and Tsaban [10] in 2016. While the discussion in the present chapter is entirely on classical algorithms, it is also worth mentioning the paper [19], where the authors independently provide a quantum algorithm that solves the DLP in a torsion semigroup.

Both the algorithm of Monico and the one of Banin and Tsaban are probabilistic and might fail with low probability. Further, some of their methods are heuristic, dependent on an oracle or some additional assumption, and their success rates and expected number of steps are either conjectured or stated loosely. It is therefore of interest to come up with an algorithm which deterministically computes the discrete logarithm in a semigroup. In this regard we like to make some analogy to the problem of determining if an integer is a prime number, a problem of great importance in cryptography. Nowadays in practice the algorithm of Miller and Rabin [63, 83] has been used for many years. Still it was a great result when Agrawal, Kayal and Saxena [3] came up with a deterministic polynomial time algorithm to achieve this goal.

A key step in finding the discrete logarithm in a semigroup is computing the cycle length of an element. Both the algorithms of [10] and [64] rely on computing some multiple of the cycle length, and then removing "extra" factors by taking gcd's until

the cycle length is obtained. Once the cycle length value is obtained, the discrete logarithm may easily be computed with a few more simple steps. While Monico does not provide further elaboration on how this is done, the paper by Banin and Tsaban bridges this knowledge gap by showing how the problem is reduced to a DLP in a group once the cycle length and start values are known.

Denote by N_x the order of x (formally defined in Definition 2.4). The complexity of the algorithm in [10] is $\mathcal{O}(\sqrt{N_x}(\log N_x)^2 \log\log N_x)$, and the complexity of the one in [64] is $\mathcal{O}(\sqrt{N_x}(\log N_x)^2)$. While both of the existing methods seem to succeed with high probability for practical values, we show that the process of taking successive gcd's/factors is unnecessary, and that one can deterministically find the cycle length. The main contribution of this paper will be a deterministic algorithm for computing the discrete logarithm of an element y in some semigroup S with respect to some torsion base element $x \in S$. The time complexity of our algorithm is $\mathcal{O}(\sqrt{N_x}(\log N_x)^2)$.

2.2.1 Preliminaries

A semigroup S is a set together with an associative binary operation. Like in group theory where a torsion group consists of elements of finite order only we define:

Definition 2.1 (Torsion Element). Let S be a semigroup. An element $x \in S$ is called a torsion element if the sub-semigroup $\langle x\rangle := \{x^k \mid k \in \mathbb{N}\}$ generated by x, is finite. S is called a torsion semigroup if every $x \in S$ is a torsion element.

Throughout the chapter the following definitions will be assumed:

Definition 2.2 (Cycle Start). Let $x \in S$. The cycle start s_x of x is defined as the smallest positive integer such that $x^{s_x} = x^b$ for some $b \in \mathbb{N}$, $b > s_x$.

Definition 2.3 (Cycle Length). Let $x \in S$. The cycle length L_x of x is defined as the smallest positive integer such that $x^{s_x+L_x} = x^{s_x}$.

Definition 2.4 (Element order). Let $x \in S$. With notation as above, we define the order N_x of x as the cardinality of the sub-semigroup $\langle x\rangle$. Note that $N_x = s_x + L_x - 1$.

Definition 2.5 (Semigroup DLP). Let S be a semigroup and $x \in S$. The semigroup DLP is defined as follows. Given $y \in \langle x\rangle := \{x^k \mid k \in \mathbb{N}\}$, find all $m \in N$ such that $x^m = y$.

We state below a key result first proved in [10].

Lemma 2.1 ([10]). Let S be a semigroup and $x \in S$ be an element with cycle start s_x. The set of powers $G_x = \{x^{s_x+k} \mid k \geq 0\}$ of x forms a finite cyclic group. The

identity element of G_x is given by x^{tL_x}, where t is the minimum positive integer such that $x^{tL_x} \in G_x$.

The following result is stated in [64] in a slightly different formulation. We provide an equivalent proof based on the group structure of G_x.

Lemma 2.2 ([64]). Let $x \in S$ have cycle start s_x and cycle length L_x. For all integers $n, m \geq s_x$, we have $x^m = x^n \iff n \equiv m \pmod{L_x}$.

Proof. We can assume without loss of generality that $n \geq m$, and so we can write $n = m + kL_x + u$, with $k \geq 0$ and $0 \leq u < L_x$. First suppose that $n \equiv m \pmod{L_x}$, i.e. $u = 0$. Since $m, n \geq s_x$, we have $x^n = x^{m+kL_x} = x^m$.

Conversely, if $x^n = x^m$, write $n_1 = n - s_x \geq 0$, and $m_1 = m - s_x \geq 0$. We have

$$x^{s_x+m_1} = x^{s_x+n_1} = x^{s_x+m_1+kL_x+u} = x^{s_x+m_1+u}.$$

Now, without loss of generality, $m_1 \geq s_x$, because if not, one can always increment m_1 and n_1 by multiples of L_x until this happens. So, we can assume that x^{m_1} lies in G_x and is thus invertible. We multiply by the inverse on both sides to finally get

$$x^{s_x} = x^{s_x+u}.$$

Thus, we must have $u = 0$ or $n \equiv m \pmod{L_x}$, as required. □

Remark 2.1. It becomes clear from the above discussion that the standard collision-based algorithms for order and discrete log computations in a group do not adapt directly to a general semigroup. Collision-based algorithms for the computation of the order N of a group element x (for instance, see [95]) are based on the principle that whenever N can be expressed in the form $N = A - B$ for non-negative integers A and B, the collision $x^A = x^B$ always occurs. However, this principle does not work in a semigroup, where there are two independent components of the order. More specifically, for a semigroup element x with cycle length L_x and cycle start s_x, whenever L_x may be expressed in the form $A - B$ for non-negative integers A and B, the equality $x^A = x^B$ holds if and only if $A, B \geq s_x$. As an example, consider a semigroup element x with cycle length $L_x = 12$ and cycle start $s_x = 5$. Then, $L_x = 15 - 3$, but $x^{15} \neq x^3$. Thus without prior knowledge of the cycle start, the semigroup order N_x or cycle length L_x cannot directly be found using the same collision-based algorithms for groups.

Similarly, collision-based algorithms fail for discrete log computations in a semigroup. As an example, consider a semigroup element x with cycle length $L_x = 15$ and cycle start $s_x = 10$, and suppose that the discrete log of $y = x^5$ is to be found.

Then $y \cdot x^6 = x^{11} = x^{26}$ is obtained as a collision. However, unlike in the group case, the conclusion $y = x^{26-6} = x^{20}$ is wrong since $x^5 \neq x^{20}$. This happens because even though x is torsion and forms a cycle of powers, it is not invertible.

This concludes the prerequisite knowledge on torsion elements in semigroups. In the next section, we study the existing probabalistic algorithms for cycle lengths, and analyse their assumptions, working and complexities.

2.2.2 Existing Probabalistic Agorithms

Banin and Tsaban's Algorithm

In this section, we study the probabalistic algorithm described in [10] for computing the cycle length of a torsion element in a semigroup. While the authors of the original paper describe their theory only for torsion semigroups, it will become clear that the same discussion holds true for any semigroup when the base element chosen is torsion.

Let S be a semigroup and x be a torsion element of S. Let s_x denote the cycle start of x and L_x its cycle length. Then, recall from Lemma 2.1 that $G_x := \{x^{s_x}, x^{s_x+1}, \ldots, x^{s_x+L_x-1}\}$ is a cyclic group, and that it has order L_x. The authors of [10] assume the availability of a 'Discrete Logarithm Oracle' for the group G_x, which returns values $\log_x h$ for $h \in G_x$. They state that these values need not be smaller than the group order but are polynomial in the size of G_x and the element x. The representation of the identity in G_x is unknown, and a method to compute inverses is not available.

The authors claim that the well-known algorithms for discrete logarithm computations in groups do not explicitly require inverses, or can easily be modified to work without the use of inverses. While it is true that these algorithms make use of mainly the existence of inverses rather than their explicit computation, we believe that the fact that easy modification is possible is not immediate without some justification. In fact, it will become clear in the later sections that the modified Baby-Step-Giant-Step algorithm devised by Monico [64] (and also the deterministic algorithm presented in Section 2.2.3) is a crucial and non-trivial part of any such modification.

We make the following observation from the proof of Lemma 1 found in [10]. For any $k \geq 0$, denote by v_k the smallest positive integer such that

$$v_k L_x \geq 2s_x + k.$$

We then have $x^{v_k L_x - s_x - k} \in G_x$ and

$$x^{s_x+k} x^{v_k L_x - s_x - k} = x^{v_k L_x} = x^{tL_x}, \tag{2.1}$$

so the inverse of the element x^{s_x+k} of G_x is given by $x^{v_k L_x - s_x - k}$. In particular, the computation of inverses requires prior knowledge of the cycle start and cycle length. As will be explained below, the cycle start may be computed only once the value of the cycle length is known, using a binary search. This explains why the authors in [10] insist that their Discrete Logarithm Oracle does not need to use the computation of inverses.

Below, we describe Algorithm 2.1, which is the algorithm suggested in [10] to compute the order of the group G_x, i.e. the cycle length L_x of x.

Algorithm 2.1: Banin-Tsaban Algorithm for Cycle Length

Input A semigroup S and a torsion element $x \in S$; a DLP oracle for finite cyclic groups

Output The cycle length L_x of x

1: Initialize $i, j, g, L_x \leftarrow 1$, $N >> s_x + L_x$. Fix bounds $r > 1, s > 1$.
2: **while** $j < s$
   1. Fix a random $z \in \{\lfloor N/2 \rfloor, \ldots, N\}$ and set $h = x^z$.
   2. **while** $i < r$
      (a) Choose a random number $k_i > 0$.
      (b) Use the DLP oracle to compute $k_i' = \log_h(h^{k_i})$.
      (c) Set $g \leftarrow \underset{j \le i}{\gcd}(k_j - k_j') = \gcd\left(\underset{j<i}{\gcd}(k_j - k_j'),\ k_i - k_i{}'\right)$.
      (d) Set $i \leftarrow i + 1$.
   3. **end while**
   4. Set $L_x \leftarrow \mathrm{lcm}(L_x, g)$, $j \leftarrow j + 1$.
6: **end while**
7: Return L_x.

We first note that the authors state complexities in terms of L_x, which are valid when the bound N for N_x is known. If the algorithm fails for a value of N, the authors suggest to double N and try again. In this case, which we will assume from now on, we assert that the complexities need to be taken in terms of N_x instead of L_x. The oracle may be assumed to have the standard complexity of $O(\sqrt{N_x}\log N_x)$ steps for discrete logarithm calculations.

Step (2.2.c) takes $\mathcal{O}(\log(\max_{j \le i}(k_j - k_i)) = \mathcal{O}(\log N_x)$ integer operations by the assumption on the oracle, which does not contribute to the total complexity. Thus, the total complexity of Step (2.2) comes from the oracle alone, and is $\mathcal{O}(\sqrt{N_x}\log N_x)$. Now, the authors of [10] remark that r and s can be taken to satisfy $r = \mathcal{O}(1)$ and $s = \mathcal{O}(\log\log N_x)$. Thus, the total complexity is $\mathcal{O}(\log N_x)$ times the complexity of Algorithm 2.1, and thus $\mathcal{O}(\log\log(N_x)\log N_x)$ times the complexity of Step (2.2).

Therefore, we get the total complexity of $\mathcal{O}(\log\log N_x(\log N_x)^2\sqrt{N_x})$.

Finally, in Algorithm 2.2, we present the application of the binary search method to find the cycle start once L_x is known. This algorithm is formulated as below for this purpose in [10], though the idea to use a binary search is also originally mentioned in [64].

Algorithm 2.2: Calculating Cycle Start (Binary Search)

Input A semigroup element x with cycle length L_x
Output Cycle start s_x of x

1: Initialize $s_x \leftarrow 1$
2: **while** $x^{s_x+L_x} \neq x^{s_x}$ **do**
  $s_x \leftarrow 2s_x$
3: **end while**
4: Set $a \leftarrow s_x/2$
5: **while** $|a - s_x| \geq 2$
  $c \leftarrow (a + s_x)/2$
  if $x^{c+L_x} \neq x^c$ **then**
    $a \leftarrow c$
  else
    $s_x \leftarrow c$
6: **end while**

Lemma 2.3. Let N_x be the order of the element x. Then Algorithm 2.2 requires

$$\mathcal{O}\left((\log N_x)^2\right).$$

steps.

Proof. Each of Steps (2) and (5) involves $\mathcal{O}(\log N_x)$ rounds, each of which computes requires $\mathcal{O}(\log N_x)$ semigroup multiplications and one comparison. The total complexity is thus $\mathcal{O}\left((\log N_x)^2\right)$. □

Monico's Algorithm

In his PhD thesis [64], Chris Monico provides a probabilistic algorithm (described below as Algorithm 2.3) that calculates the cycle length of an element in a finite ring of order N. This algorithm makes use of the multiplicative semigroup structure of the finite ring, and of the availability of the explicit bound N for every cycle length, and is in fact applicable to any semigroup where such a bound N is available. In this subsection, we analyse this algorithm, provide a more concrete bound on its success

rate, and compute its complexity in terms of N. We will discuss this algorithm in terms of torsion semigroups, as opposed to finite rings.

Algorithm 2.3: Monico's Baby-Step Giant-Step for Cycle Length

Input A finite semigroup S with $|S| = N$ and an element $x \in S$
Output The cycle length L_x of x

1: Set $m = \lceil \sqrt{N} \rceil$. Choose a prime $q > N$.
2: For $0 \leq i \leq m$, compute and store in a table the pairs $(i; x^{q+im})$. Sort the table by the second component.
3: Find the least positive integer b_1 such that x^{q+b_1} is in the table: $x^{q+b_1} = x^{q+a_1 m}$. (Note: $0 < b_1 < m$).
4: Find the least positive integer b_2 such that x^{2q+b_2} is in the table: $x^{2q+b_2} = x^{q+a_2 m}$. (Again, $0 < b_2 < m$).
5: Compute $g = \gcd(a_1 m - b_1, a_2 m - b_2 - q)$.
6: For each divisor d of g below some bound B, do the following.
  If $x^{N+g/d} = x^N$:
    set $g \leftarrow g/d$;
7: Output $L_x = g$ and stop.

We first note that if $L_x > m$ and the table in Step (2) has repeated entries $x^{q+i_1 m} = x^{q+i_2 m}$, then numbers b_1 and b_2 may not exist below m. In this case the algorithm needs to be modified to take $g \leftarrow (i_1 - i_2)m$. However, whenever this case does not arise, it can be shown that steps 3 and 4 are always successful in finding a collision.

We further remark that in Step (6), the list of divisors of g is kept fixed, while g is updated to g/d whenever the condition is satisfied. In the subsequent steps, non-divisors of g/d can be immediately discarded. However, the end result depends on the order in which divisors are tested, which the algorithm does not mention explicitly. However, we note that it is, in fact, possible to restrict the testing to only the prime power divisors of g below B, and with this setting, the optimal performance is obtained by taking divisors in decreasing order. We will assume this set-up for the rest of the analysis. For completeness, we restate Algorithm 2.3 with the above clarifications in Algorithm 2.4.

Note that Step (2) involves $\mathcal{O}(\log N)$ multiplications to compute x^q and another $\mathcal{O}(\sqrt{N})$ multiplications to compute $x^q, x^q \cdot x^m, x^q \cdot x^{2m}, \ldots, x^q \cdot x^{m^2}$. Step 3 involves at most m multiplications $x^{q+1} = x^q \cdot x, x^{q+1} \cdot x, \ldots, x^{q+m-1}$, with complexity $\mathcal{O}(\sqrt{N})$, and match-finding with the first list, with complexity $\mathcal{O}(\sqrt{N} \log N)$ with standard sorting and search algorithms. The same is true for Step (4). Step (5) has complexity $\mathcal{O}(\log \max(a_1 m - b_1, a_2 m - b_2 - q)) = \mathcal{O}(\log N)$ and so does not contribute to the overall complexity. Step (6) involves B iterations of a multiplication and an

Algorithm 2.4: Restated: Monico's Baby-Step Giant-Step for Cycle Length

Input A finite semigroup S with $|S| = N$ and an element $x \in S$
Output The cycle length L_x of x

1: Set $m = \lceil \sqrt{N} \rceil$. Choose a prime $q > N$.
2: For $0 \leq i \leq m$, compute and store in a table the pairs $(i; x^{q+im})$. Sort the table by the second component. If a collision $x^{q+i_1 m} = x^{q+i_2 m}$ occurs, set $g = (i_1 - i_2)m$ and go to Step (6).
3: Find the least positive integer b_1 such that x^{q+b_1} is in the table: $x^{q+b_1} = x^{q+a_1 m}$. (Note: $0 < b_1 < m$).
4: Find the least positive integer b_2 such that x^{2q+b_2} is in the table: $x^{2q+b_2} = x^{q+a_2 m}$. (Again, $0 < b_2 < m$).
5: Compute $g = \gcd(a_1 m - b_1, a_2 m - b_2 - q)$.
6: Fix a bound B and compute all the divisors of g below B. Denote these by $d_1 > d_2 > \ldots > d_r$.
7: For $i = 1, \ldots, r$, do the following.
  If $x^{N+g/d_i} = x^N$:
    set $g \leftarrow g/d_i$;
8: Output $L_x = g$ and stop.

exponentiation $x^{g/d}$, and thus has a time complexity of $\mathcal{O}(B(\log g+1)) = \mathcal{O}(B \log N)$ multiplications.

In the original work, Monico states that the bound B of Algorithm 2.3 can always be chosen so that $B < \sqrt{a_1 m - b_1}$. We remark that this claim does not hold in the current setting of the algorithm. For example, with a cycle length value of 4, and $a_1 m - b_1 = 104$, $a_2 m - b_2 - q = 52$, we get $g = 52$. If $B < \sqrt{a_1 m - b_1} = \sqrt{104} < 11$, then we would only test divisors d below 11, and would never factor out 13 to obtain the true cycle length. For such a bound to work, one needs to modify the algorithm to test both divisors d and g/d in Step (6). However, we will show in Lemma 2.4 that it is almost always sufficient to take B to be a reasonably large fixed constant, thus the complexity of Step (6) can be counted as $\mathcal{O}(\log N)$, and does not contribute to the overall complexity. Thus, the overall time complexity is $\mathcal{O}(\sqrt{N} \log N)$. If N is unavailable, the algorithm can also be modified to update the value of N by doubling at each step until a large enough value is found. In this case, Algorithm 2.3 has a total complexity of $\mathcal{O}(\sqrt{N_x}(\log N_x)^2)$.

Further, Monico suggests a modification to the above algorithm, viz. to find several such a_i and b_i and compute all the gcd's. It is clear that this suggestion is exactly the method used in Banin and Tsaban's algorithm as discussed above.

We now analyse the probability of success. The algorithm first looks for a collisions of the form $x^{q+a_1m} = x^{q+b_1}$. The working principle is that in this case, the cycle length L_x divides $a_1m - b_1$. Similarly, if also $x^{q+a_2m} = x^{2q+b_2}$ then $g = \gcd(a_1m - b_1, a_2m - b_2 - q)$ is a multiple of L_x.

So far, the process is essentially the same in both Algorithms 2.1 and 2.3: while the former uses a discrete logarithm oracle to obtain multiples of the cycle length, the latter directly finds these multiples by finding collisions. However, in Algorithm 2.3, we do not proceed with computing multiple factors of L_x, but work with the fixed multiple g of L_x, whereas in Algorithm 2.1 this multiple shrinks several times.

Algorithm 2.3 then proceeds by fixing a bound B and iterating over every number d below B to check if $d \mid g$. If yes, it executes the next part, i.e. checks if $x^{N+g/d} = x^N$, and if this holds, it sets $g \leftarrow g/d$. Note that if the factorization of the number g is known (or if g can be factored in time negligible compared to $O(\sqrt{N})$, then we do not need this fixed bound B, and can instead iterate over every prime factor d of g. It is well-known that the number of prime factors of g counted with multiplicity is $\mathcal{O}(\log g)$, so Step (5) of the algorithm can find L_x in $\mathcal{O}(\log N)$ steps. However, in general, factoring g may be difficult, so we assume from here on that the algorithm proceeds by fixing a bound B for the divisors of g. Below we analyse the probability of the algorithm succeeding in terms of B and g.

Lemma 2.4. The probability that Algorithm 2.4 succeeds is bounded below by $\left(1 - \frac{1}{B}\right)^{\log g}$ where g is defined in Step 5 of Algorithm 2.4.

Proof. We write $g = L_x \cdot F$ for some number F and suppose that the algorithm fails. This means that there is a divisor, and hence also a prime power divisor of F, which the algorithm fails to factor out. Let p be a prime dividing F, α_p denote its largest power dividing F, and β_p be its largest power below the fixed bound B. So, we have $p^{\alpha_p} \mid F$, $p^{\alpha_p+1} \nmid F$, $p^{\beta_p} \leq B$, $p^{\beta_p+1} > B$.

Note that the number of times the algorithm divides g by p is

$$\sum_{i=1}^{\beta_p} i = \beta_p \cdot (\beta_p + 1)/2.$$

Since divisors are taken in decreasing order, we must have $\beta_p \cdot (\beta_p + 1)/2 < \alpha_p$ if the algorithm fails. So, the algorithm succeeds as long as $\beta_p \cdot (\beta_p + 1)/2 \geq \alpha_p$ for every prime divisor p of F. Thus, the probability of success for the algorithm can be bounded below by

$$\prod_{p \mid g} \text{Prob}\left(\frac{\beta_p \cdot (\beta_p + 1)}{2} \geq \alpha_p\right).$$

Write $v_p = \frac{\beta_p(\beta_p+1)}{2}$ for simplicity. We may assume that g is a random multiple of L_x below the bound B, so F is a random number in $\left\{1, \ldots, \frac{B}{L_x}\right\}$. We have,

$$\begin{aligned}\text{Prob}(\alpha_p \leq v_p) =& 1 - \text{Prob}(p^{v_p+1} \mid F) \\ =& 1 - \left(\frac{B/L_x}{p^{v_p+1}(B/L_x)}\right) \\ =& 1 - 1/p^{v_p+1} = 1 - \frac{1}{p^{\frac{\beta_p(\beta_p+1)}{2}+1}}.\end{aligned}$$

Hence, a lower bound for the probability of the algorithm's success is

$$\prod_{p|F}\left(1 - \frac{1}{p^{\frac{\beta_p\cdot(\beta_p+1)}{2}+1}}\right).$$

Now, we have,

$$\begin{aligned}p^{\beta_p+1} > B \iff \frac{1}{p^{\beta_p+1}} < \frac{1}{B} \\ \implies 1 - \frac{1}{p^{\frac{\beta_p(\beta_p+1)}{2}+1}} > 1 - \frac{1}{B^{\frac{\beta_p}{2}}p} > 1 - \frac{1}{B},\end{aligned}$$

where the last inequality follows from the following argument. If $\beta_p \geq 2$, then it clearly holds. If $\beta_p = 0$, this means by definition that $p > B$, and so again, the inequality holds. Finally, if $\beta_p = 1$, then $p \leq B$ and $p^2 > B$, or $p > B^{1/2}$, and so $B^{1/2}p > B$. Thus, in every case, one has $1 - \frac{1}{B^{\beta_p/2}p} > 1 - \frac{1}{B}$.

We further make the following observation. Let $\omega(n)$ denote the number of distinct prime divisors of integer n (note, however, that the same statement also holds if counted with multiplicity). Then clearly, $2^{\omega(n)} \leq n$, and so, taking logarithms, $\omega(n) \leq \log_2 n$.

Collecting all the above results, we conclude that the probability of success Prob (success) of Algorithm 2.3 is bounded below as follows.

$$\begin{aligned}\text{Prob (success)} \geq& \prod_{p|F}\left(1 - \frac{1}{B}\right) \\ =& \left(1 - \frac{1}{B}\right)^{\omega(F)} \geq \left(1 - \frac{1}{B}\right)^{\log F} \\ \geq& \left(1 - \frac{1}{B}\right)^{\log g}.\end{aligned}$$

□

Note that this bound shows that Algorithm 2.3 is indeed successful with overwhelming probability, as conjectured by the author. For example, with $B = 10^6$, even when g is several orders of magnitude larger than B, say $g = 2^{4000}$, the probability of success is greater than 99.6 percent, by the bound derived in Lemma 2.4.

2.2.3 Deterministic Solution of the DLP

The solution of the DLP in a semigroup involves two parts: the calculation of the cycle length L_x and cycle start s_x of the base element x. These values are needed to find the discrete log.

Deterministic Algorithm for Cycle Length Computation

We now present our deterministic algorithm for the computation of the cycle length. It works by finding a suitable collision, and also guarantees finding the actual cycle length rather than just a multiple of it, in a fixed number of steps.

Algorithm 2.5: Deterministic Algorithm for Cycle Length

Input A semigroup S and a torsion element $x \in S$.
Output Cycle length L_x of x

1: Initialize $N \leftarrow 1$.
2: Set $q \leftarrow \lceil \sqrt{N} \rceil$.
3: Compute, one by one, $x^N, x^{N+1}, \ldots, x^{N+q}$ and check for the equality $x^N = x^{N+j}$ at each step $j \geq 1$. Store these values in a table as pairs $(N+j, x^{N+j})$, $0 \leq j < q$. If $x^N = x^{N+j}$ for any $j < q$, then set $L_x \leftarrow j$ and end the process. If not, proceed to the next step.
4: For $0 \leq i \leq q$, compute, one by one, the values $x^{N+q}, x^{N+2q}, \ldots, x^{N+iq}$ and at each step i, look for a match in the table of values calculated in Step (3).
5: Suppose that a match $x^{N+iq} = x^{N+j}$ is found, and i is the smallest integer such that this happens. Set $L_x \leftarrow iq - j$ and end the process.
6: If no match is found in steps 3 or 5, set $N \leftarrow 2 \cdot N$ and go back to Step (2).

Theorem 2.1. Let S be a semigroup and $x \in S$ a torsion element with order N_x. If an upper bound on N_x is known, Algorithm 2.5 returns the correct value of the cycle length L_x with

$$\mathcal{O}\left(\sqrt{N_x} \cdot (\log N_x)^2\right)$$

steps. The total space complexity is $\mathcal{O}\left(\sqrt{N_x}\right)$ semigroup elements.

Proof. We first assume $N \geq \max(L_x, s_x)$ and show that Steps (1-5) succeed in finding L_x. We have $q = \lceil \sqrt{N} \rceil$. If $L_x < q$, then the equality $x^N = x^{N+L_x}$ is found

in the first step and the statement of the theorem follows. Else if $L_x \geq q$, we can write uniquely

$$L_x = iq - j,$$

for some positive integers $i > 0$, $0 \leq j < q$. Now, we must have $i \leq q$, because otherwise if $i \geq q + 1$, we would have

$$L_x \geq (q+1)q - j > q^2 + q - q = q^2 \geq N,$$

a contradiction.

We have

$$\begin{aligned} & L_x = iq - j,\ 0 < i \leq q, 0 \leq j < q \\ \implies & N + j + L_x = N + iq \\ \implies & x^{N+j} = x^{N+j+L_x} = x^{N+iq}, \end{aligned}$$

where the last step follows because $N > s_x$ by assumption. So, such a collision always occurs between elements of the two lists in the algorithm.

We now claim that for the smallest such integer i computed in Step (5) of Algorithm 2.5, $L_x = iq - j$. To see this, let i be the smallest positive integer such that

$$x^{N+j} = x^{N+iq}.$$

Also let $L_x = i'q - j'$, $0 < i' \leq q$, $0 \leq j' < q$. We have already shown above that such integers i' and j' exist for our choice of N. By the definition of L_x, we must have $L_x \mid iq - j$. Now suppose that $i' > i$. Then,

$$\begin{aligned} i'q - j' \geq & (i+1)q - j' \\ = & iq + (q - j') > iq \\ \geq & iq - j. \end{aligned}$$

But, $L_x = i'q - j' \mid iq - j$, so we must have $iq - j = i'q - j'$. Since $i' > i$, this means that

$$q \leq (i' - i)q = (j' - j) < j',$$

which is a contradiction because $0 \leq j' < q$. So, we must have $i' = i$, $j' = j$. This proves the claim.

We have shown above that the algorithm finds the correct cycle length when $N > \max(s_x, L_x)$. Since the algorithm doubles the value of N until a match is found, it always terminates and outputs the correct cycle length. We now look at the time complexity.

For a given N, Step (2) involves one exponentiation, or $\mathcal{O}(\log N)$ multiplications to find x^N and then at most another $q = \mathcal{O}(\sqrt{N})$ multiplications and equality checks for $x^N \cdot x, x^N \cdot x^2, \ldots, x^N \cdot x^q$. This step also needs a storage space of at most $q = \mathcal{O}(\sqrt{N})$ elements. Step 5 needs one exponentiation or $\mathcal{O}(\log N)$ multiplications to find x^q, and then another $q = \mathcal{O}(\sqrt{N})$ multiplications to find $x^{N+q} \cdot x^q, x^{N+q} \cdot x^{2q} \ldots, x^{N+q^2}$. Finding matches in Steps (3) and (5) can be done in $\mathcal{O}(q \log q) = \mathcal{O}(\sqrt{N} \log \sqrt{N})$ comparisons with the use of sorting and efficient look-up methods. Thus, clearly, steps 1 to 5 in algorithm 2.5 have a total complexity of $\mathcal{O}(\sqrt{N} \log N)$.

Moreover, the algorithm starts at $N = 1$ and doubles N until the cycle length is found, i.e. until $N > \max(s_x, L_x)$. Thus, the number of times steps 2 to 5 are performed is

$$\lceil \log (\max (L_x, s_x)) \rceil = \mathcal{O} (\max (\log (L_x), \log(s_x))) = \mathcal{O}(\log N_x).$$

Thus, the total number of steps involved is

$$\mathcal{O}\left(\sqrt{N_x} \cdot (\log N_x)^2\right).$$

Clearly, Step (3) involves the storage of $q = \lceil \sqrt{N} \rceil = \mathcal{O}\left(\sqrt{\max(s_x, L_x)}\right) = \mathcal{O}\left(\sqrt{N_x}\right)$ elements, so this value gives the total space complexity. This completes the proof. □

Remark 2.2. If a bound N on the order N_x is known a priori, then Algorithm 2.5 can clearly be completed in a single round, with time complexity $\mathcal{O}\left(\sqrt{N} \cdot (\log N)\right)$.

Remark 2.3. For the case of a group, there exist better algorithms for the computation of the order of an element even when the total group order is unbounded. For instance, Algorithm 3.3 in [95] uses a growth function $d(t)$, which generalizes the square root function used above, to compute the order N of a group element x, and achieves time and space complexities of $\mathcal{O}\left(\sqrt{N}\right)$, thus eliminating the additional $\log N$ multiplier introduced by the method in Algorithm 2.5. Under this method, one has functions $g(t)$ and $b(t)$ of the stage t, which dynamically replace the constants N and q, respectively.

However, this method fails when used for a general semigroup due to the presence of two independent unknown components of the order. To see this, note that the algorithm would need to be modified for a semigroup as follows. At stage t, one has $g(t-1) \leq N_x < g(t)$. On the completion of the baby steps, one has a table with the powers $x^{g(t)}, x^{g(t)+1}, \ldots, x^{g(t)+b(t)}$ (the addition of $g(t)$ is necessary in the semigroup case to ensure that the loop is entered). The giant

steps compute $x^{g(t)+g(t-1)+b(t)}, x^{g(t)+g(t-1)+2\cdot b(t)}, \ldots, x^{g(t)+g(t-1)+d(t)\cdot b(t)} = x^{2g(t)}$. Now, while N_x is guaranteed to have a unique expression as $g(t-1)+ib(t)-j$ with $0 < i \leq d(t)$ and $0 \leq j \leq b(t)$, this does not necessarily lead to a collision. In fact, if $b(t) < L_x < g(t-1)$ and $2L_x > g(t) = g(t-1)+d(t)\cdot b(t)$, then neither the baby steps nor the giant steps leads to a collision, and the cycle length is never found (note that this can happen only if $L_x > s_x$). Moreover, if a collision $x^{g(t)+g(t-1)+ib(t)} = x^{g(t)+j}$ is obtained in the giant step phase, the only conclusion that can be drawn is that $L_x \mid g(t-1)+ib(t)-j$. If instead we forced the condition $g(t-1) \leq N_x < g(t)$, a collision again may never occur because there is no control on the cycle start (For instance, in matrix semigroups over finite simple semirings, the cycle start is often found to be much larger than the cycle length. In such cases, adapting group-based algorithms would fail). See Remark 2.1 for further details.

Experimental Results for Cycle Length Computations We used Algorithm 2.5 to compute cycle length values in several common semigroups, such as matrix semigroups over finite fields, matrix semigroups over the finite simple semiring S_{20} (see [110] for a construction and [58] for the addition and multiplication tables), and the symmetric and alternating groups (where the cycle length is precisely the order of the element). We further used the obtained cycle lengths to compute the cycle start values using Algorithm 2.2. The working code may be found at https://github.com/simran-tinani/semigroup-cycle-length.

Solving the DLP once the Cycle Length is known

In this section, we demonstrate the solution of the DLP for a torsion element x in the semigroup S once the cycle length is known. As before let N_x be the order of the sub-semigroup $\langle x \rangle$, let L_x be the cycle length of the torsion element x (which we assume is already computed) and let $y \in \langle x \rangle$ be an element.

In [10], the authors demonstrate the next steps in solving for $\log_x(y)$, via a reduction to a DLP in the group G_x, once L_x and s_x are known. The procedure is described in Algorithm 2.6 below, which has been adapted from the original formulation in [10].

Since authors of [10] do not give an explicit proof of correctness Step 5 in Algorithm 2.6, we provide it in Theorem 2.2. Before this, we will need the following technical result.

Lemma 2.5. Let L_x be the cycle length of $x \in S$, and n, a, and a' be fixed positive integers. Suppose that $x^{bL_x+n} = x^a \in G_x$, where b is the minimum number such that $x^{bL_x+n} \in G_x$, and $x^{n-cL_x} = x^{a'} \in G_x$, where c the maximum number such that $x^{n-cL_x} \in G_x$. Then

$$bL_x + n \leq a, \text{ and } n - cL_x \leq a'.$$

Algorithm 2.6: Algorithm for Discrete Logarithm

Input A semigroup S, a torsion element $x \in S$, with cycle length L_x and cycle start s_x, and $y \in S$ with $y = x^m$.

Output The discrete logarithm m of y with base x.

1: Compute $t = \left\lceil \frac{s_x}{L_x} \right\rceil$ and define $x' = x^{tL_x+1} \in G_x$.
2: Find the minimum number $0 \leq b \leq t$ such that $y' = y \cdot x^{bL_x} \in G_x$ using binary search.
3: Use Shanks' Baby-Step Giant-Step algorithm for the group $\langle x' \rangle \subseteq G_x$ to compute $m' \in \{0, 1, \ldots, L_x - 1\}$ such that $(x')^{m'} = y'$.
4: Find the maximum number $c \geq 0$ such that $x^{(tL_x+1)m'-cL_x} \in G_x$ using binary search.
5: Return $m = m'(tL_x + 1) - (b + c)L_x$.

Proof. First let $x^{bL_x+n} = x^a$ with b minimal such that $x^{bL_x+n} \in G_x$. Suppose, to the contrary, that $bL_x + n > a$. We must have, by the minimality of b, $x^{(b-1)L_x+n} \notin G_x$, so $(b-1)L_x + n < a$.

$$\begin{aligned} \text{But,} \quad & x^{bL_x+n} = x^a \in G_x \\ \implies & bL_x + n - a = kL_x, \ k \geq 1 \\ \implies & (b-k)L_x + n = a \\ \implies & x^{(b-k)L_x+n} = x^a \in G_x, \ k \geq 1. \end{aligned}$$

This is a contradiction to the minimality of b. So, $bL_x + n \leq a$. Now suppose that $x^{n-cL_x} = x^a \in G_x$, with c maximal, and suppose that $n - cL_x > a'$. We argue as above:

$$\begin{aligned} & L_x \mid n - cL_x - a' \\ \implies & n - (k+c)L_x = a', \text{ for some } k \geq 1 \\ \implies & x^{n-(k+c)L_x} = x^{a'} \in G_x, \end{aligned}$$

which is a contradiction to the maximality of c. Thus $n - cL_x \leq a'$. □

Theorem 2.2. Let S be a semigroup, $x \in S$ a torsion element and $y \in \langle x \rangle$ any element. Assume the cycle length L_x and cycle start s_x of x are known. Then Algorithm 2.6 returns the correct values of the discrete logarithm $m = \log_x(y)$ in $\mathcal{O}\left(\sqrt{L_x} + (\log N_x)^2\right)$ semigroup multiplications, with a required storage of $\mathcal{O}\left(\sqrt{L_x}\right)$ semigroup elements.

Proof. We use the notations of Algorithm 2.6, and also write $n = \log_x y$. We will show that the output m is equal to the correct discrete logarithm value n. Recall that we have a group G_x, generated by $x' := x^{tL_x+1}$, and with identity x^{tL_x}. The parameter t is given by the formula $t = \left\lceil \frac{s_x}{L_x} \right\rceil$. Inverses in G_x can be computed in polynomial time using the formula (2.1).Note that membership in G_x can be tested with one equality check: $y \in G_x \iff y \cdot x^{L_x} = y$. There are now two cases:

1. When $y \in G_x$, we have $b = 0$. Here, it is possible to use Shanks' Baby Step-Giant Step algorithm [89] which is a deterministic algorithm and which requires $\mathcal{O}\left(\sqrt{L_x}\right)$ semigroup multiplications and storage space $\mathcal{O}\left(\sqrt{L_x}\right)$, in order to compute $\log_{x'}(y)$. This is done in Step (3). From this value, $n = \log_x(y)$ is readily computed, as shown below. Note that in this case, $\log_x(y)$ is determined modulo L_x.

2. When $y \notin G_x$, Algorithm 2.6 first computes, using binary search, the smallest power b of x^{L_x} such that the product $y \cdot x^{bL_x}$ lies in the group G_x, and then proceeds as in case 1 via the Baby Step-Giant Step algorithm to find the discrete logarithm m' of $y \cdot x^{bL_x}$ with base x' (i.e. $(x')^{m'} = y \cdot x^{bL_x}$). Note that in this case, the value of $\log_x(y)$ is less than s_x, and is thus determined uniquely in $\mathbb{N}$. Again, the time and space complexity are both $\mathcal{O}\left(\sqrt{L_x}\right)$.

In both cases above, we have the maximal value c such that $x^{m'(tL_x+1)-cL_x} \in G_x$, and so $c \leq L_x + s_x + 1 = N_x + 1$, since $m' \leq L_x$ and $tL_x \leq L_x + s_x$. We also clearly have $b \leq t \leq N_x$. Since the computations of both b and c are done via binary searches, they contribute $\mathcal{O}((\log N_x)^2)$ steps to the overall time complexity. Now,

$$x^{m'(tL_x+1)-cL_x} = x^{m'(tL_x+1)} = (x')^{m'} = x^{bL_x+n}.$$

Applying Lemma 2.5 to the above equation, we must have

$$m'(tL_x + 1) - cL_x \leq bL_x + n, \text{ and } bL_x + n \leq m'(tL_x + 1) - cL_x.$$

Therefore, $bL_x + n = m'(tL_x + 1) - cL_x$, or $n = m'(tL_x + 1) - (b + c)L_x$, which is precisely equal to m, the value returned by the Algorithm 2.6. Thus, $m = n$. This completes the proof. □

Combining Theorem 2.1, Lemma 2.3 and Theorem 2.2 we arrive at the main proposition of the paper:

Proposition 2.1. Let S be a semigroup, $x \in S$ a torsion element and $y \in \langle x \rangle$ any element. The discrete logarithm $m = \log_x(y)$ can be computed deterministically in

$$\mathcal{O}\left(\sqrt{N_x} \cdot (\log N_x)^2\right)$$

steps, with a required storage of $\mathcal{O}\left(\sqrt{N_x}\right)$ semigroup elements.

Proof. For the solution, one begins by finding L_x. This can be done using Algorithm 2.5 and according to Theorem 2.1 this requires $\mathcal{O}\left(\sqrt{N_x} \cdot (\log N_x)^2\right)$ steps and the storage of $\mathcal{O}\left(\sqrt{N_x}\right)$ elements.

By Lemma 2.3 the computation of the cycle start s_x is achieved in $\mathcal{O}((\log N_x)^2)$ semigroup multiplications, which does not contribute to the overall cost of the algorithm.

By Theorem 2.2, the discrete logarithm m can then be retrieved using Algorithm 2.6, in $\mathcal{O}\left((\log N_x)^2 + \sqrt{L_x}\right)$ steps, with a required storage of $\mathcal{O}\left(\sqrt{L_x}\right)$ semigroup elements.

As $L_x \leq N_x$, the overall complexity is dominated by the computation of the cycle length, and the proof of the result is now clear. □

Solving the DLP once the Factorization of the Cycle Length is known

We mentioned in the introduction that for a general group of order N the best general known algorithms for solving the discrete logarithm problem have complexity $\mathcal{O}(\sqrt{N})$ operations.

In case the order N has a prime factorization into small primes there is the famous Pohlig–Hellman algorithm [80] for solving the DLP whose complexity is dominated by the largest prime factor in the integer factorization of N.

In case that we have available the integer factorization of the cycle length L_x we can adapt the Pohlig–Hellman algorithm for groups to a Pohlig–Hellman algorithm for solving the DLP in a semigroup. Algorithm 2.7 represents this adapted Pohlig–Hellman algorithm.

Theorem 2.3. Let S be a semigroup, $x \in S$ a torsion element and $y \in \langle x \rangle$ any element. Assume the cycle start s_x of x is known and assume the integer factorization of the cycle length L_x is known to be $L_x = \prod_{i=1}^{r} p_i^{e_i}$. Then Algorithm 2.7 computes the discrete logarithm $\log_x y$ requiring $\mathcal{O}\left(\sum_{i=1}^{r} e_i \left(\log L_x + \sqrt{p_i}\right) + (\log N_x)^2\right)$ steps. The space complexity of the algorithm consists in $\mathcal{O}\left(\sum_{i=1}^{r} e_i \sqrt{p_i}\right)$ semigroup elements.

Proof. Steps 1 and 2 are in analogy to the corresponding steps of Algorithm 2.6. Steps 3 to 5 represent the Pohlig–Hellman algorithm for groups with the implied complexity dominated by the largest prime factor p_i of the integer factorization of L_x (for a reference on Pohlig–Hellman in groups, see in [45, Theorem 2.32]). It

Algorithm 2.7: Pohlig–Hellman Algorithm for solving the Discrete Logarithm Problem in a Semigroup

Input A semigroup S, a torsion element $x \in S$, with cycle length $L_x = \prod_{i=1}^{r} p_i^{e_i}$ and cycle start s_x, and $y \in S$ with $y = x^m$

Output The discrete logarithm m of y with base x

1: Compute $t = \left\lceil \frac{s_x}{L_x} \right\rceil$ and define $x' = x^{tL_x+1} \in G_x$.

2: Find the minimum number $0 \leq b \leq t$ such that $y' = y \cdot x^{bL_x} \in G_x$ using binary search.

3: **for** $i \in \{1, \ldots, r\}$

1. Compute the values $x_i' := (x')^{L_x/p_i^{e_i}}$, $y_i' := (y')^{L_x/p_i^{e_i}}$, and $\gamma_i := (x_i')^{p^{e_i-1}}$.
2. Calculate the inverse z_i of ${x_i}'$ in G_x using (2.1).
3. Set $k \leftarrow 0$ and $n_0 \leftarrow 0$.
4. **while** $k < e_i$ **do**
   (a) Compute $y_k' = (y_i' z_i^{n_k})^{p^{e_i-1-k}} \in \langle \gamma_i \rangle$.
   (b) Use Shanks' Baby-Step Giant-Step algorithm for the group $\langle \gamma_i \rangle \subseteq G_x$ to compute $d_k \in \{0, 1, \ldots, p_i - 1\}$ such that ${\gamma_i}^{d_k} = {y_k}'$.
   (c) Set $n_{k+1} \leftarrow n_k + p_i^k d_k$, and $k \leftarrow k + 1$.
5. **end while**
6. Set $m_i := n_{e_i}$.

4: **end for**

5: Use the Chinese Remainder Theorem to solve the congruence equations

$$m' \equiv m_i \pmod{p_i^{e_i}}, \quad \forall\, i \in \{1, \ldots, r\}$$

uniquely for $m' \pmod{L_x}$. This gives the discrete logarithm of y' with respect to the base x' in the group G_x.

6: Find the maximum number $c \geq 0$ such that $x^{(tL_x+1)m'-cL_x} \in G_x$ using binary search.

7: Return $m = m'(tL_x + 1) - (b + c)L_x$.

follows that the running time of the algorithm is $\mathcal{O}\left(\sum_{i=1}^{r} e_i \left(\log L_x + \sqrt{p_i}\right)\right)$ steps. The computation of b and c require in addition $(\log N_x)^2$ steps. The total space complexity is $\mathcal{O}\left(\sum_{i=1}^{r} e_i \sqrt{p_i}\right)$ semigroup elements and that completes the proof. □

2.3 DLP in an infinite polynomial semiring

In general when considering polynomials in cryptography, one considers finite polynomial rings obtained by quotienting polynomial rings by an irreducible polynomial (for example, finite fields). In this section, we explore exponentiation, and subsequently, the complexity of the DLP in an infinite polynomial semiring. First observe that if R is an integral domain, then the DLP in the ring $R[x]$ is trivially solved by simply dividing the degrees of the two given polynomials to obtain the exponent. However, in the case where R is not an integral domain, and particularly, if R is not a ring but a semiring, the degree of $f(x)^n$ is not necessarily equal to $\deg(f(x)) \cdot n$.

We will consider two finite semirings S_6 and S_{20}, which were found and described in [110]. Both S_6 and S_{20} are finite simple semirings, and up to isomorphism the unique simple semirings with respectively 6 and 20 elements. The addition and multiplication tables of these can be found below in Tables 2.1, 2.2.a and 2.2.b.

+	0	1	2	3	4	5
0	0	1	2	3	4	5
1	1	1	1	1	1	5
2	2	1	2	1	2	5
3	3	1	1	3	3	5
4	4	1	2	3	4	5
5	5	5	5	5	5	5

·	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	1	2	3	4	5
2	0	2	2	0	0	5
3	0	3	4	3	4	3
4	0	4	4	0	0	3
5	0	5	2	5	2	5

Table 2.1: Arithmetic in S_6

With computational experiments, we investigated the statistical properties of exponentiation in the polynomial rings $S_6[x]$ and $S_{20}[x]$. For this, we manually programmed the addition and multiplication tables, defined the corresponding polynomial semirings, and implemented polynomial exponentiation through square and multiply. Further, we used a function to generate random polynomials over the semiring, calculate their degrees, exponentiate by random integer, and then compared the degrees of the original and resulting polynomials. Some selected results can be found in table 2.3.

+	0	a	b	c	d	e	f	g	h	i	j	k	ℓ	m	n	o	p	q	r	1
0	0	a	b	c	d	e	f	g	h	i	j	k	ℓ	m	n	o	p	q	r	1
a	a	a	b	c	d	e	f	g	h	i	j	k	ℓ	m	n	o	p	q	r	1
b	b	b	b	c	e	e	f	g	h	i	k	k	ℓ	m	n	o	p	q	r	1
c	c	c	c	c	f	f	f	h	h	i	ℓ	ℓ	ℓ	1	n	p	p	q	r	1
d	d	d	e	f	d	e	f	g	h	i	j	k	ℓ	m	n	o	p	q	r	1
e	e	e	e	f	e	e	f	g	h	i	k	k	ℓ	m	n	o	p	q	r	1
f	f	f	f	f	f	f	f	h	h	i	ℓ	ℓ	ℓ	1	n	p	p	q	r	1
g	g	g	g	h	g	g	h	g	h	i	m	m	1	m	n	o	p	q	r	1
h	h	h	h	h	h	h	h	h	h	i	1	1	1	1	n	p	p	q	r	1
i	i	i	i	i	i	i	i	i	i	i	n	n	n	n	n	q	q	q	r	n
j	j	j	k	ℓ	j	k	ℓ	m	1	n	j	k	ℓ	m	n	o	p	q	r	1
k	k	k	k	ℓ	k	k	ℓ	m	1	n	k	k	ℓ	m	n	o	p	q	r	1
ℓ	ℓ	ℓ	ℓ	ℓ	ℓ	ℓ	ℓ	1	1	n	ℓ	ℓ	ℓ	1	n	p	p	q	r	1
m	m	m	m	1	m	m	1	m	1	n	m	m	1	m	n	o	p	q	r	1
n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	q	q	q	r	n
o	o	o	o	p	o	o	p	o	p	q	o	o	p	o	q	o	p	q	r	p
p	p	p	p	p	p	p	p	p	p	q	p	p	p	p	q	p	p	q	r	p
q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	q	r	q
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
1	1	1	1	1	1	1	1	1	1	n	1	1	1	1	n	p	p	q	r	1

Table 2.2.a: Addition in S_{20}

·	0	a	b	c	d	e	f	g	h	i	j	k	ℓ	m	n	o	p	q	r	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
a	0	0	0	0	0	0	0	0	0	0	a	a	a	a	a	b	b	b	c	a
b	0	0	0	0	a	a	a	b	b	c	a	a	a	b	c	b	b	c	c	b
c	0	a	b	c	a	b	c	b	c	c	a	b	c	b	c	b	c	c	c	c
d	0	0	0	0	0	0	0	0	0	0	d	d	d	d	d	g	g	g	i	d
e	0	0	0	0	a	a	a	b	b	c	d	d	d	e	f	g	g	h	i	e
f	0	a	b	c	a	b	c	b	c	c	d	e	f	e	f	g	h	h	i	f
g	0	0	0	0	d	d	d	g	g	i	d	d	d	g	i	g	g	i	i	g
h	0	a	b	c	d	e	f	g	h	i	d	e	f	g	i	g	h	i	i	h
i	0	d	g	i	d	g	i	g	i	i	d	g	i	g	i	g	i	i	i	i
j	0	0	0	0	0	0	0	0	0	0	j	j	j	j	j	o	o	o	r	j
k	0	0	0	0	a	a	a	b	b	c	j	j	j	k	ℓ	o	o	p	r	k
ℓ	0	a	b	c	a	b	c	b	c	c	j	k	ℓ	k	ℓ	o	p	p	r	l
m	0	0	0	0	d	d	d	g	g	i	j	j	j	m	n	o	o	q	r	m
n	0	d	g	i	d	g	i	g	i	i	j	m	n	m	n	o	q	q	r	n
o	0	0	0	0	j	j	j	o	o	r	j	j	j	o	r	o	o	r	r	o
p	0	a	b	c	j	k	ℓ	o	p	r	j	k	ℓ	o	r	o	p	r	r	p
q	0	d	g	i	j	m	n	o	q	r	j	m	n	o	r	o	q	r	r	q
r	0	j	o	r	j	o	r	o	r	r	j	o	r	o	r	o	r	r	r	r
1	0	a	b	c	d	e	f	g	h	i	j	k	ℓ	m	n	o	p	q	r	1

Table 2.2.b: Multiplication in S_{20}

As expected, given a polynomial p and an exponent e, the degree of the resulting polynomial $q := p^e$ was, in all cases, smaller than the quantity $\deg(p) \cdot e$. We also computed the value of $\lceil(\deg(q)/\deg(p))\rceil$, which is the best average guess for the value of e if the semiring is considered to not have many zero divisors.

For many instances in $S_6[x]$, this guess actually matched the value of e, and for all instances, the difference was a very small quantity. In $S_6[x]$, for most instances, $\lceil(\deg(q)/\deg(p))\rceil$ was different from the true value of the exponent e. However, the maximum difference between these two quantities still stays very small. Thus, in both cases an adversary only has to try a very small number of possible exponents in order to solve the DLP.

$\deg p$	e	$\deg q := p^e$	$\deg p \cdot e$	$\left\lceil \frac{\deg q}{\deg p} \right\rceil$
59	83	4815	4897	82
40	14	547	560	14
43	17	715	731	17
80	29	2292	2320	29
67	62	4093	4154	62
39	69	2623	2691	68
75	42	3109	3150	42
22	62	1303	1364	60
23	53	1167	1219	51
84	93	7720	7812	92
83	97	7955	8051	96
9	9	73	81	9
91	62	5581	5642	62
96	90	8551	8640	90
29	98	2745	2842	95
55	6	325	330	6
35	59	2007	2065	59
17	13	209	221	13
39	15	571	585	15
96	28	2661	2688	28

$\deg p$	e	$\deg q := p^e$	$\deg p \cdot e$	$\left\lceil \frac{\deg q}{\deg p} \right\rceil$
14	82	1067	1148	77
17	36	577	612	34
10	71	640	710	64
8	58	407	464	51
20	70	1331	1400	67
16	36	541	576	34
11	34	341	374	31
19	31	559	589	30
61	22	1321	1342	22
64	63	3970	4032	63
98	95	9216	9310	95
87	26	2237	2262	26
44	23	990	1012	23
13	97	1165	1261	90
32	98	3039	3136	95
28	96	2593	2688	93
100	83	8218	8300	83
59	94	5453	5546	93
28	53	1432	1484	52
39	92	3497	3588	90

Table 2.3: Exponentiation in $S_6[x]$ (left) and $S_{20}[x]$ (right)

2.4 Semigroup actions on a set

Let X be any set. A semigroup S is said to act (from the left) on X if there is a function (called the S-action)

$$S \times X \to X$$

such that for every $s, t \in S$ and $x \in X$, $(st) \cdot x = s \cdot (t \cdot x)$. In [58], abelian semigroup actions were used to construct a generalized Diffie-Hellman-like public key exchange protocol based on a generalization of the discrete logarithm problem, which the authors call the semigroup action problem.

Definition 2.6 (Semigroup Action Problem). Given a semigroup G acting on a set X and elements $x, y \in X$, find $g \in G$ such that $gx = y$, given that such a g exists.

Protocol 2.1 (Extended Diffie-Hellman Key Exchange).
Let X be a finite set, G be an abelian semigroup, and ϕ a G-action on X. The Extended Diffie-Hellman key exchange in (G, X, ϕ) is the following protocol:

1. *Alice and Bob publicly agree on an element $s \in S$.*
2. *Alice chooses $a \in G$ and computes her public key as. Her private key is a.*
3. *Bob chooses $b \in G$ and computes his public key bs. His private key is b.*
4. *Their common secret key is $a(bs) = (a \cdot b)s = (b \cdot a)s = b(as)$*

This construction was generalized and further investigated in several works [34, 35, 58]. In fact, when the semigroup is an abelian group, this construction may be seen as an abstract formulation of some isogeny-based cryptographic problems [4, 23, 94]. In [34], the authors show Pohlig-Hellman type reductions which may lead to a method of attacks on cryptosystems based on semigroup actions. For this, they define the notion of reductions.

Definition 2.7. Let S and T be semigroups acting respectively on sets X and Y. A reduction is a tuple (f, F, G) of maps $f : S \to T$ and $F, G : X \to Y$ such that for all $s \in S$ and $x \in X$, we have $f(s) \cdot G(x) = F(s \cdot x)$. A reduction is called effective if the maps f, F, G are efficiently computable and $1 < |T| < |S|$.

Given a semigroup action problem instance $y = sx$ the authors of [34] suggest that the existence of an "effective" reduction (f, F, G) can serve as an attack. In particular, given a semigroup action problem instance $x, y \in X$ where $y = s \cdot x$, and a reduction (f, F, G), one has another semigroup action problem instance $G(x), F(y) \in Y$ where $F(y) = f(s) \cdot G(x)$. Thus, an adversary who can solve the semigroup action problem in T can restrict the search in S to preimages of the solutions in T under f. However, we observe that for these redutions to be truly practically effective, some further assumptions are needed. In general, it does not seem that reductions by themselves in general comprise a general and effective attack method in all cases.

Firstly, we observe that the reduced problem is not exactly an instance of the semigroup action problem because there is an added constraint: the adversary needs to

find $t \in \mathrm{Im}(f) \subseteq T$ such that $F(y) = tG(x)$. Thus, effectively the adversary has to solve a simultaneous semigroup action problem and a subset membership problem in T. For some actions this constraint could add a dimension of extra complexity.

Further, once one solves the reduced semigroup action problem for $t \in \mathrm{Im}(f) \subseteq T$, one still needs to search in $f^{-1}(t) \subseteq S$ for a solution to the original problem. So, this set must be significantly smaller than (polynomial size in) S, or there should be sufficiently many solutions within this set to render the probability of finding a solution high enough. As an extreme example, if the solution $s \in S$ of the original semigroup action problem is unique, then the adversary has to brute force ove all of $f^{-1}(t) \subseteq S$. Note that in the case of the DLP, the preimage $f^{-1}(t)$ itself is unique, and easily computable by the Chinese Remainder Theorem. This convenience is likely to be absent for other semigroup actions.

Another important constraint is that it is not enough for $|T|$ to simply be smaller than $|S|$, it needs to be smaller by orders of magnitude for the complexity of a generic solution to the reduced semigroup action problem to be lower than the original. Thus, in general the condition $|T| < |S|$ does not seem sufficient to define the reduction as effective. In order to benefit from reduction to the group action case, the adversary must be able to get there in polynomially many reductions, which, apriori, does not seem to always be possible.

Intuitively, the two preceding constraints are conflicting. The smaller T is in comparison to S, the larger the sizes of the preimages $f^{-1}(t)$ would be on average. Suppose that S is a semigroup and T is a subsemigroup of S much smaller than S. Then, to reduce the problem from S to T one either needs to apply a large number of reductions, or a small number of "highly effective" reductions, where the size shrinks significantly at each step. For the latter case, $f^{-1}(t)$ is likely to be too big for brute force search. Thus, it seems possible that in some cases these reductions are not very effective, and in fact that a clever choice of s and S do not allow for any benefit from performing them at all.

As an example, suppose that each reduction f_i is such that the preimage size magnifies by a factor of 2, i.e. $|f_i^{-1}(T_i)| = 2|T_i|$. Then, for $f = f_1 f_2 \dots f_n$ a composition of reductions, the search space in S has size 2^n. Thus, while one may end up with a group action at the end, the remaining problem need not be negligible, and can in fact dominate the algorithm for the final solution.

Chapter 3

Public-Key Cryptography in Non-Abelian Groups

3.1 Introduction

The construction and realization of cryptographic systems that resist quantum attacks presently constitutes an important area of research. Apart from lattice-based, multivariate, and code-based cryptography, it has been proposed recently to use the rich structure of non-abelian groups to construct quantum-secure protocols for public key exchange, message encryption, and authentication. Since Shor's quantum algorithm relies on the solution of the Hidden Subgroup Problem for finite abelian groups, it is believed that non-commutative groups may offer a promising avenue. Some recent surveys on this emerging field, called group-based cryptography, can be found in [66] and [30].

The most prominent algorithmic problem employed for constructing non-abelian protocols is the Conjugacy Search Problem (henceforth written CSP). While the Discrete Logarithm Problem (henceforth written DLP) in a group G requires the recovery of the exponent n when given the group elements g and $h = g^n$, the CSP requires the recovery of a conjugator $x \in G$, given the elements g and $h = x^{-1}gx$. To reflect this analogy, it is common to use the notation $g^x := x^{-1}gx$ for $g, x \in G$, which we also adopt in this chaoter. If the conjugator is restricted to lie in a subgroup $A \subseteq G$, we refer to the problem as an A-restricted CSP.

The first and most prominently known protocols constructed based on the CSP were by Ko-Lee [52] and Anshel, Anshel and Goldfeld (AAG) [5] and both of whose underlying problems is a specific restricted CSP.

Protocol 3.1 (Ko et al. [52]). Let G be a finitely generated group, with subgroups A and B that commute element-wise, i.e. $ab = ba\ \forall a \in A, \forall b \in B$. Choose a base element $w \in G$. The parameters G, A, B, and w are public.

1. Alice chooses a secret element $a \in A$, and publishes $w^a = a^{-1}wa$.
2. Bob chooses a secret element $b \in B$, and publishes $w^b = b^{-1}wb$.

3. Alice computes $K_A = (w^b)^a$, and Bob computes $K_B = (w^a)^b$.

Since a and b commute, we have a common shared secret $K_A = K_B = a^{-1}b^{-1}wab$.

Note that the Ko et al. protocol has a major disadvantage in that it still requires the elements of the chosen subgroups to commute with each other. This condition narrows down the possibilities for the platforms that can be used drastically, and also makes it difficult to find a suitable platform. The following protocol from [5] shows a workaround to this hindrance.

Protocol 3.2 (**Anshel-Anshel-Goldfeld**). Let G be a group, and $N_1, N_2 \in \mathbb{N}$, $1 \leq L_1 \leq L_2$, and $L \in \mathbb{N}$ be parameters.

1. Alice randomly generates an N_1-tuple of words $\bar{a} = (a_1, \ldots, a_{N_1})$ in G each of length between L_1 and L_2. The tuple $\bar{a}$ is called Alice's public set.
2. Bob randomly generates an N_2-tuple of words $\bar{b} = (b_1, \ldots, b_{N_2})$ in G each of length between L_1 and L_2. The tuple $\bar{b}$ is called Bob's public set.
3. Alice randomly generates $A = a_{s_1}^{\epsilon_1} \ldots a_{s_L}^{\epsilon_L}$, where $0 < s_i < N_1$ and $\epsilon_i = \pm 1$ (for each $1 \leq i \leq L$). A is Alice's private key.
4. Bob randomly generates $B = b_{t_1}^{\delta_1} \ldots b_{t_L}^{\delta_L}$, where $0 < t_i < N_1$ and $\delta_i = \pm 1$ (for each $1 \leq i \leq L$). B is Bob's private key.
5. Alice computes $b_i' = A^{-1}b_iA$ $(1 \leq i \leq N_2)$ and transmits them to Bob.
6. Bob computes $a_i' = B^{-1}a_iB$, $1 \leq i \leq N_1$ and transmits them to Alice.
7. Alice computes $K_A = A^{-1}a_{s_1}'^{\epsilon_1} \ldots a_{s_L}'^{\epsilon_L}$. Clearly $K_A = A^{-1}B^{-1}AB$.
8. Alice computes $K_B = b_{t_L}'^{-\delta_L} \ldots b_{t_1}'^{-\delta_1}B$. Clearly $K_B = A^{-1}B^{-1}AB$.

$K = K_A = K_B$ is the shared secret key.

The security of the protocol clearly reduces to the following mathematical problem.

Definition 3.1 (Commutator Key Exchange Problem). Let G be a group. Let $a_1, \ldots, a_k, b_1, \ldots, b_k \in G$. Let $a \in \langle a_1, \ldots, a_k \rangle$, $b \in \langle b_1, \ldots, b_k \rangle$. Given $a_1, \ldots, a_k, b_1, \ldots, b_k$, $a_1^b, \ldots, a_k^b, b_1^a, \ldots, b_k^a$, compute $a^{-1}b^{-1}ab$.

In order to break the system and compute K, it is sufficient for the eavesdropper to solve (subgroup-restricted SCSP) find one of the following:

- an element $A' \in \langle a_1, \ldots, a_{N_1} \rangle$ such that $\bar{b}' = A'^{-1}\bar{b}A'$,
- an element $B' \in \langle b_1, \ldots, b_{N_2} \rangle$ such that $\bar{a}' = B'^{-1}\bar{a}B'$.

Note that in order for the secret keys A and B to stay hidden during the transmission process, or more generally, for the commutator key exchange problem to be difficult, one requires a suitable representation of group elements, from which the required decomposition into the given generators cannot be deduced easily. A well-defined format for the representation of group elements is called a normal form.

The authors of both these systems proposed as platforms the Braid groups B_N, which will be described in Section 3.4. However, a number of attacks [46, 67, 103] show that the braid groups are not suitable platforms. Nevertheless, the possibility of finding another potential non-abelian platform group for CSP-based protocols is still open to research. Some other groups that have been proposed for use are polycyclic groups, metabelian groups, p-groups, Thompson groups, and matrix groups.

When the underlying platform group is linear (i.e. embeds faithfully into a matrix group over a field), several polynomial time attacks exist, which focus on retrieving the private shared key without solving the CSP [12, 54, 65, 103]. However, in general the computation of an efficient linear representation may pose a serious roadblock for an adversary. Further, many of these attacks are impractical to implement for standard parameter values. Such attacks are also typically protocol-specific, and this always leaves open the possibility of constructing a different protocol, again based on the CSP, where the known attacks are avoided. So far, the true difficulty of the CSP in different platforms has not been sufficiently investigated.

Summary of Contributions

This chapter is focused on the study of the algorithmic complexity of the conjugacy search problem in some classes of non-abelian platform groups.

The main contribution of this chapter, Section 3.2, gives a polynomial time reduction from the conjugacy search problem to (a polynomial number of) discrete logarithm problems, in two classes of groups. We produce a polynomial time solution for the CSP in a finite polycyclic group with two generators, and show that a restricted CSP is reducible to a DLP. In matrix groups over finite fields, we use the Jordan decomposition of a matrix to produce a polynomial time reduction of an A-restricted CSP, where $A \subseteq GL(\mathbb{F}_q)$ is a cyclic subgroup, to a set of DLPs over an extension of $\mathbb{F}_q$. This theory is used to cryptanalyse the systems proposed in [41, 93, 105]. A direct consequence of our results is that the security of a protocol based on a cyclic-restricted conjugacy search problem in linear platform group, essentially depends on the difficulty of computing a representation of the platform and a set of DLPs. We believe that our methods and findings are likely to allow for several other heuristic attacks in the general case. The material in Section 3.2 is adapted from the preprint [101] co-authored by me.

In Section 3.3 we introduce a concept called efficient C-decomposability, which we show is sufficient for the CSP in a central product of groups to reduce to individual CSP's in the components. This shows the importance of exercising caution when selecting a platform group for a CSP-based system: central products with the efficient C-decomposability property must be avoided, and a chosen platform must in a way be "atomic". We use this concept to demonstrate a polynomial time solution of the CSP in any extraspecial p-group. The result is practically relevant since several non-abelian groups are constructed by combining smaller groups by taking direct, central, and semidirect products (see, for example, [15, 25]). The material in Section 3.3 is adapted from the preprint [100] authored by me.

Section 3.4 is a broad overview of some of the platforms groups and computational problems employed in non-abelian group-based cryptography, and some of the common protocols for key exchange and signatures. In this section, we also discuss the general and specific methods of cryptanalysis of non-abelian group-based systems. This section serves more as a survey, and contains no new results.

3.2 Complexity of CSP in some polycyclic and matrix groups

In this section, we study the complexity of various versions of the CSP in two well-known classes of linear groups: polycyclic groups and matrix groups over finite fields. Polycyclic groups were suggested for cryptographic use in [29], where some evidence was provided for resistance to some known attacks. Matrix groups are important for any non-abelian system, since whenever the platform group is linear, an efficient faithful representation reduces the underlying problem to one in a matrix group. Several proposed non-abelian cryptosystems use platforms that are special instances of polycyclic or matrix groups, and employ problems either equivalent to, or easier than, the versions of the CSP discussed in this chapter.

Our results show that security of the independent systems in [41, 93, 105] relies on a problem at most as hard as a set of DLPs, so that the CSP here offers no novel security feature. More generally, a direct consequence of our results is that a CSP-based protocol in a linear platform with efficient representation must ensure that the conjugators come from a subgroup with more than two generators in order to have any potential security benefit. On the other hand, the case of finite polycyclic groups with two generators shows that a larger subgroup A often offers an attacker with more flexibility, rendering the A-restricted CSP easier. We believe that our findings are likely to allow for several other heuristic attacks in the general case, when the conjugators are not carefully chosen. The algebraic reductions demonstrated in this

chapter may also prove useful in future cryptanalytic techniques for non-abelian protocols over different platform groups.

3.2.1 Polycyclic groups

In [29], polycyclic groups were suggested as a potential platform for conjugacy-based cryptography, and some evidence was provided for these groups resisting the length-based attacks afflicting braid group-based systems. A survey of polycyclic group-based cryptography can be found in [40]. This section discusses the complexity of some special cases of the CSP in polycyclic groups. We start by introducing polycyclic groups and some of their basic properties, and in particular briefly discuss the complexity of performing the group operations.

Definition 3.2 (Polycyclic Group). A polycyclic group is a group G with a subnormal series $G = G_1 > G_2 > \ldots > G_{n+1} = 1$ in which every quotient G_i/G_{i+1} is cyclic. This series is called a polycyclic series.

Definition 3.3 (Power-Conjugate Presentation). Let G be a group with generators $a_1, a_2, \ldots, a_n$. Let $I \subseteq \{1, 2, \ldots, n\}$ denote a list of indices and $m_i > 1$ be integers corresponding to elements $i \in I$. A power-conjugate presentation is a group presentation of the form

$$\begin{aligned} G = \langle a_1, a_2, \ldots, a_n \mid & a_i^{m_i} = w_{ii},\ i \in I, \\ & a_j^{a_i} = w_{ij},\ 1 \leq i < j \leq n, \\ & a_j^{a_i^{-1}} = w_{-ij},\ 1 \leq i < j \leq n, i \notin I\rangle, \end{aligned} \tag{3.1}$$

where the words w_{ij} are of the form $w_{ij} = a_{|i|+1}^{l(i,j,|i|+1)} \ldots a_n^{l(i,j,n)}$, with $l(i,j,k) \in \mathbb{Z}$, and $0 \leq l(i,j,k) < m_k$ if $k \in I$.

Lemma 3.1 ([47]). G is polycyclic if and only if it has a polycyclic presentation.

Define $G_i = \langle a_i, a_{i+1} \ldots a_n\rangle$, $1 \leq i < n$, $G_{n+1} = \langle 1\rangle$. The presentation in (3.1) is called *consistent* if $|G_i/G_{i+1}| = m_i$ whenever $i \in I$, and the G_i/G_{i+1} is infinite whenever $i \notin I$.

Definition 3.4 (Geodesic Form). Let G be generated by a set of symbols X, and $a_i \in X \cup X^{-1}$, $1 \leq i \leq n$. A word $w = a_1 a_2 \ldots a_n$ is said to be in geodesic form if there is no shorter word that represents the same group element.

Definition 3.5 (Normal Form). Given a consistent polycyclic presentation (3.1) for a group G, every element a of G can be represented uniquely in the form $a = a_1^{e_1} a_2^{e_2} \ldots a_n^{e_n}$ where $e_i \in \mathbb{Z}$, $0 \leq e_i \leq m_i$ for $i \in I$. This is called the normal form of a.

Henceforth, as is standard, we will use the normal form to represent words in polycyclic groups, and also assume that the presentations we deal with are all consistent. Given a word w in G, the process by which minimal non-normal subwords are reduced to normal form using the relations in (3.1) is called *collection*.

Many different strategies for this process have been suggested, but the best-known performance in most cases is achieved by the Collection from the Left Algorithm [106], and its improvement in [33]. Under this, a word

$$w = (x_1^{\alpha_1} x_2^{\alpha_2} \dots x_n^{\alpha_n})(x_{i_1}^{\beta_1} \dots \dots x_{i_t}^{\beta_t})$$

is represented in two parts: collected part (as a vector $(\alpha_1, \alpha_2, \dots, \alpha_n)$) and uncollected stack of generator powers $s = (x_{i_1}^{\beta_1}, \dots, x_{i_t}^{\beta_t})$. To collect w into normal form, a finite number of generator powers from the uncollected part are iteratively pushed onto the collected part by using the relations from the definition.

In general, the running time of this algorithm depends on the exponents appearing across the whole process. Due to the dependence on the exponents in the intermediate steps, the complexity remains a rough and unclear estimate. It is clear that collection is a key part of the group operations since it is involved in both multiplication and inversion of words in G, and so in general operations in a polycyclic group may be inefficient. However, in many special cases, the complexity can be bounded. We discuss the complexity of computing in polycyclic groups for some cases, including all finite polycyclic groups, in the next subsection.

Complexity of Group Operations

Throughout this section, G denotes a polycylic group given by the presentation (3.1). Below, we begin our discussion by showing that collected words in the last two generators can be multiplied, inverted and exponentiated, using explicit formulas. For simplicity, write the relations in x_{n-1} and x_n as $x_n^{x_{n-1}} = x_n^L, x_n^{x_{n-1}^{-1}} = x_n^D$ for fixed $L, D \in \mathbb{Z}$.

Lemma 3.2. We have a formula to collect any word of the form $x_n^i x_{n-1}^j$. For any $A, B \in \mathbb{Z}$,

$$x_n^B x_{n-1}^A = \begin{cases} x_{n-1}^A {x_n}^{BL^A} \text{ if } A \geq 0, \\ x_{n-1}^A {x_n}^{BD^{-A}} \text{ if } A < 0. \end{cases} \tag{3.2}$$

The following formula allows the computation of the product of r words given in normal form. It is easily verified by induction on r.

Lemma 3.3. Let $r \geq 1$ and $w_i = x_{n-1}^{A_i} x_n^{B_i}$ for $1 \leq i \leq r$, and $w = w_1 w_2 \dots w_r$.

Define

$$k_j = \begin{cases} 1, A_j \geq 0 \\ 0, A_j < 0, \end{cases} \qquad A_i^+ = \sum_{j=i+1}^{r} A_j k_j,\ A_i^- = \sum_{j=i+1}^{r} A_j(1-k_j).$$

Then, $w = x_{n-1}^A x_n^B$, with $A = \sum_{i=1}^{r} A_i$, and $B = \sum_{i=1}^{r} B_i L^{A_i^+} D^{A_i^-}$

Lemma 3.4. Let $K > 0$ and consider an element $w = x_{n-1}^A x_n^B$. Write $E(A) = L$, if $A \geq 0$, $E = D$ if $A < 0$, $F(A) = L$, if $A < 0$, $F(A) = D$ if $A \geq 0$ ($E_j = k_j L + (1-k_j)D$). Then

$$(x_{n-1}^A x_n^B)^K = x_{n-1}^{KA} x_n^{B\left(\frac{E^{|KA|}-1}{E^{|A|}-1}\right)},$$

$$(x_{n-1}^A x_n^B)^{-K} = x_{n-1}^{-KA} x_n^{-BF^{|KA|}\left(\frac{E^{|KA|}-1}{E^{|A|}-1}\right)}.$$

Thus, operations on normal words in x_{n-1} and x_n have complexity $\mathcal{O}(1)$. We now consider words in the three generators x_{n-2}, x_{n-1}, x_n.

Define constants $A_j^{(k)}, B_j^{(k)}$ with $x_{n-2}^{-k} x_j x_{n-2}^k = x_{n-1}^{A_j^{(k)}} x_n^{B_j^{(k)}}$ for $j = n-1, n$, and $k \in \mathbb{Z}$. Clearly, $A_j^{(\pm 1)}$ and $B_j^{(\pm 1)}$ can be read from the group presentation.

Given $A = A_j^{(k)}$ and $B = B_j^{(k)}$ for any $k \geq 0$, we obtain $A_j^{(k+1)}$ and $B_j^{(k+1)}$ (resp. $A_j^{(k-1)}$ and $B_j^{(k-1)}$ if $k < 0$) in time $\mathcal{O}(1)$ by computing

$$x_{n-2}^{-1}(x_{n-1}^A x_n^B)x_{n-2} = (x_{n-1}^{x_n})^A (x_3^{x_1})^B = w_1^A w_2^B$$

which requires two substitutions from the presentation, two exponentiations and one word multiplication, of words in x_{n-1} and x_n. The square and multiply method can be used for subsequent exponents, giving a total complexity of $\mathcal{O}(\log k)$ for computing $(A_j^{(k)}, B_j^{(k)})$. Note that if $n \geq 4$ the complexity of computing

$$x_{n-3}^{-1}(x_{n-2}^A x_{n-1}^B x_n^C)x_{n-3}$$

depends on all of the exponents A, B, C, and their intermediate values, and from this case onward nothing concrete can be said about the complexity in general.

Below, we consider the case where we have a general bound N for the exponents on each generator at each step of collection. For instance, if each generator x_i has finite order m_i, the exponents on x_i can always be reduced in polynomial time, so we can assume without loss of generality that $N = \max(m_i)$. In particular this holds for all finite polycyclic groups.

Proposition 3.1. Let $n \geq 4$. For $j \leq n-3$, the complexity of multiplying two words, and of inverting a single word, in the generators $x_j,\ x_{j+1}, \ldots, x_n$ is $\mathcal{O}(\frac{(n-j)!}{2}(\log N)^{2(n-j-2)+1})$.

Proof. Note that from the above discussion we have a complexity of $\mathcal{O}(\log N)$ for multiplication and inversion of words in x_{n-2}, x_{n-1} and x_n. For $j \leq n-3$ we have

$$\begin{aligned} x_j^{-1}\ (x_{j+1}^{A_{j+1}}\ x_{j+2}^{A_{j+2}}\ \ldots\ x_n^{A_n})x_j =& (x_{j+1}^{x_j})^{A_{j+1}}\ (x_{j+2}^{x_j})^{A_{j+2}}\ \ldots\ (x_{j+1}^{x_j})^{A_n} \\ =& w_{j+1,j}^{A_{j+1}}\ w_{j+2,j}^{A_{j+2}}\ \ldots\ w_{n,j}^{A_n} \\ =& \overline{w_{j+1,j}}\ \overline{w_{j+2,j}}\ \ldots\ \overline{w_{n,j}} \\ =& w \end{aligned}$$

where the expression in the second line is obtained through $(n-j)$ substitutions from the presentation, the expression in the third line is computed in $\sum_{i=j+1}^{n} O(\log A_i)$ word multiplications in $x_{j+1}, \ldots, x_n$, and the final value w is computed in $(n-j)$ word multiplications in $x_{j+1}, \ldots, x_n$. So, $x_j^{-1}(x_{j+1}^{A_{j+1}} x_{j+2}^{A_{j+2}} \ldots x_n^{A_n})x_j$ can be computed in $\mathcal{O}((n-j)\log N)$ word multiplications in $x_{j+1}, \ldots, x_n$, and so

$$x_j^{-K}(x_{j+1}^{A_{j+1}} x_{j+2}^{A_{j+2}} \ldots x_n^{A_n})x_j^K$$

can be computed in $\mathcal{O}((n-j)\log K \log N)$ word multiplications in $x_{j+1}, \ldots, x_n$. Thus, two normal words in $x_j,\ x_{j+1}, \ldots, x_n$ can be multiplied by plugging in the value of the conjugation by a power of x_j and then performing a multiplication of two words in $x_{j+1}, \ldots, x_n$. So, the total complexity is $\mathcal{O}((n-j)(\log N)^2)$ word multiplications in $x_{j+1}, \ldots, x_n$. The result on multiplication then easily follows by backwards induction on $j \leq n-3$. Similarly, a normal word in $x_j,\ x_{j+1}, \ldots, x_n$ can be inverted by performing an inversion of a word in $x_{j+1}, \ldots, x_n$ and then plugging in the value of the conjugation by a power of x_j. So, one must perform $\mathcal{O}((n-j)(\log N)^2)$ word multiplications in $x_{j+1}, \ldots, x_n$ and one inversion of a word in $x_{j+1}, \ldots, x_n$. It may be easily verified that this inversion too has an overall complexity of $\mathcal{O}(\frac{(n-j)!}{2}(\log N)^{2(n-j-2)+1})$. □

CSP in a Polycyclic Group with two generators

We consider the case $n = 2$, with two generators x_1 and x_2. Throughout, we will write $N_1 = \mathrm{ord}(x_1)$ and $N_2 = \mathrm{ord}(x_2)$ as the respective orders of x_1 and x_2 in G, which are both allowed to be infinite. We have two relations $x_1^{-1}x_2x_1 = x_2^L$ and $x_1x_2x_1^{-1} = x_2^D$ (the second is redundant if and only if N_1 is finite, in which case $D = L^{N_1-1}$). Note that if N_2 is finite then $\gcd(L, N_2) = 1$, since if not, writing $L_2 = \gcd(L, N_2) \neq 1$, we have $x_1^{-1}x_2^{N_2/L_2}x_1 = 1$, or $x_2^{N_2/L_2} = 1$, a contradiction.

The following lemma is a consequence of Lemma 3.2 for the solution of the CSP.

Lemma 3.5. The conjugated word $(x_1^c x_2^d)^{-1}(x_1^a x_2^b)(x_1^c x_2^d)$ can be collected to $x_1^g x_2^h$ with $g = a$ and

$$h = \begin{cases} -dL^a + bL^c + d; \text{ if } c,\ a \geq 0 \\ -dL^a + bD^{-c} + d; \text{ if } c < 0,\ a \geq 0 \\ -dD^{-a} + bL^c + d; \text{ if } c \geq 0,\ a < 0 \\ -dD^{-a} + bD^{-c} + d; \text{ if } c,\ a < 0 \end{cases}$$

Theorem 3.1. If $N_2 = \text{ord}(x_2)$ is finite, the CSP has a polynomial time solution in G_2.

Proof. Suppose we are given an instance of the CSP, i.e. an equation

$$(x_1^c x_2^d)^{-1}(x_1^a x_2^b)(x_1^c x_2^d) = x_1^e x_2^f,$$

where we want to solve for c and d. Then, from Lemma 3.5, $a = e \pmod{N_1}$ and the CSP is reduced to solving a modular equation for two unknowns c and d.

If $a \geq 0$, we have $f + d(L^a - 1) = bL^c$, or $f + d(L^a - 1) = bD^{-c}$. Writing $b_1 = \gcd(b, N_2)$, we see that a solution for L^c (resp D^{-c}) exists if and only if $d(L^a - 1) = -f \pmod{b_1}$. Writing $a_1 = \gcd(b_1, L^a - 1)$, a solution d for $d(L^a - 1) = -f \pmod{b_1}$ exists if and only if $a_1 \mid f$. By construction, a solution (c, d) exists, so both these conditions are satisfied. Further, a solution d to $d(L^a - 1) = -f \pmod{b_1}$ is given by $d = -(f/a_1)((L_a - 1)/a_1)^{-1} \pmod{b_1/a_1}$. Write $d = -(f/a_1)((L_a - 1)/a_1)^{-1} + Mb_1/a_1$ for some $M \in \mathbb{Z}$ which we may choose. Then,

$$M(L^a - 1)/a_1 = (f + d(L^a - 1))/b_1 = \begin{cases} (b/b_1)L^c,\ c \geq 0 \\ (b/b_1)D^{-c},\ c < 0 \end{cases}$$

Writing $A = (b/b_1)^{-1}((L^a - 1))/a_1$ (clearly $\gcd(A, N_2) = 1$), we may take $M = A^{-1} \pmod{N_2}$, so that a solution is given by $c = 0$. Then $d = (L^a - 1/a_1)^{-1}(-f + b)/a_1))$.

Similarly, a solution can be obtained for the case $a < 0$ when $N_1 = \infty$. Thus, in both cases, a solution of the CSP involves a fixed number of applications of the Euclidean algorithm, and so has polynomial time complexity. □

Theorem 3.2. If $N_2 = \text{ord}(x_2)$ is finite, the $\langle x_1 \rangle$-restricted CSP in G_2 reduces to a DLP. Further, the elements can be chosen so that it is exactly equivalent to a DLP in $(\mathbb{Z}/N_2\mathbb{Z})^*$.

Proof. Here we have the exponents from Lemma 3.5, with $d = 0$, so the CSP reduces to the retrieval of the exponent c where $f = bL^c \pmod{N_2}$. Here, for a solution to exist, $b_1 = \gcd(b, N_2)$ divides f, and we have $f/b_1 = L^c \pmod{N_2/b_1}$,

so the adversary must solve the DLP with base $L \pmod{N_2/b_1}$, for the exponent c. Choosing the base element so that b satisfies $\gcd(b, N_2) = 1$, the restricted CSP is exactly equivalent to a DLP in $(\mathbb{Z}/N_2\mathbb{Z})^*$. □

Remark 3.1. If $N_2 = \infty$, then the CSP in G_2 reduces to an exponential Diophantine integer equation $f = -dL^a + bL^c + d$. As far as the author's knowledge goes, there is no known standard technique for solving such equations, and trial and error would perhaps be the best method (for a general reference see [91]). For instance, the adversary may try different values of c until $f - bL^c$ is a multiple of $L^a - 1$, and subsequently solve for d. On the other hand, the $\langle x_1 \rangle$-restricted CSP in this case has an easy solution: the adversary solves $f = bL^c$ for c simply by taking the real number base-L logarithm of $f/b \in \mathbb{Z}$.

CSPs in some other polycyclic groups

$\langle x_1 \rangle$ -restricted CSP in a polycyclic group with three generators Here it will be convenient to write $s = x_1$, $t_1 = x_2$, $t_2 = x_3$ in the group presentation (3.1). Also, write $S = \langle s \rangle$ and $T = \langle t_1, t_2 \rangle$, $\theta = \mathrm{ord}(s)$, $\theta_1 = \mathrm{ord}(t_1)$, $\theta_2 = \mathrm{ord}(t_2)$ as the respective orders in G. Then T is a polycyclic group with two generators, and so from Lemma 3.2 we have $t_2^B t_1^A = t_1^A t_2^{BL^A}$, $A, B \in \mathbb{Z}$. Further, S acts on T via $s^{-1} t_1 s = t_1^{a_1^{(1)}} t_2^{a_2^{(1)}}$, $s^{-1} t_2 s = t_1^{a_1^{(2)}} t_2^{a_2^{(2)}}$ for fixed integers $a_i^{(j)}$, $i, j \in \{1, 2\}$.

Representing an element $t_1^A t_2^B$ of T as a tuple $(A, B) \in \mathbb{Z}/\theta_1\mathbb{Z} \times \mathbb{Z}/\theta_2\mathbb{Z}$ and writing $(t_1^A t_2^B)^{s^i} = t_1^{A_i} t_2^{B_i}$ we can describe the action of S as a recurrence relation given by $(A_0, B_0) = (A, B)$,

$$(A_{i+1}, B_{i+1}) = \left(a_1^{(1)} A_i + a_1^{(2)} B_i \pmod{\theta_1},\ a_2^{(1)} L^{A_i a_1^{(2)}} \frac{L^{A_i a_1^{(1)}} - 1}{L^{a_1^{(1)}} - 1} + a_2^{(2)} \frac{L^{B_i a_1^{(2)}} - 1}{L^{a_1^{(2)}} - 1} \pmod{\theta_2} \right)$$

Note that A_i is always given $\bmod \theta_1$ and B_i is always given $\bmod \theta_2$. While the computation of A_{i+1} and B_{i+1} involves a "coupling" between these values, the final values seen are always reduced, since they are the exponents in a reduced form word expression. Further, while the first component A_{i+1} of the tuple (A_{i+1}, B_{i+1}) is linear in A_i and B_i, it is no longer linear in the previous terms of the sequence, since B_i's relationship to A_{i-1} and B_{i-1} is non-linear. The general complexity of the $\langle s \rangle$-restricted DLP, i.e. recovering i modulo θ from (A_i, B_i) is not clear. However, note that when $a_1^{(2)} = 0 = a_2^{(1)}$, the problem reduces to a DLP in $(\mathbb{Z}/\theta_1\mathbb{Z})^*$.

$\langle x_1 \rangle$-restricted CSP in n generators when $\langle x_2, \ldots, x_n \rangle$ is abelian

Here it will be convenient to write $s = x_1$, $t_1 = x_2, \ldots, t_{n-1} = x_n$ in the group presentation (3.1). Also write $S = \langle s \rangle$ and $T = \langle t_1, \ldots, t_{n-1} \rangle$, $\theta = \mathrm{ord}(s)$, $\theta_i =$

$\mathrm{ord}(t_i)$. Write $t_i^s = t_1^{a_1^{(i)}} \dots t_{n-1}^{a_{n-1}^{(i)}}$ for $1 \leq i \leq n$. Representing the elements of T as column vectors $(r_1 \dots, r_{n-1})$ of the exponents r_i of the t_i, we can describe the action of s on T by the endomorphism

$$\mathbb{Z}_{\theta_1} \times \mathbb{Z}_{\theta_2} \times \ldots \times \mathbb{Z}_{\theta_{n-1}} \to \mathbb{Z}_{\theta_1} \times \mathbb{Z}_{\theta_2} \times \ldots \times \mathbb{Z}_{\theta_{n-1}}$$

$$(r_1, \ldots, r_{n-1}) \to \begin{bmatrix} a_1^{(1)} & \ldots & a_1^{(n-1)} \\ a_2^{(1)} & \ldots & a_2^{(n-1)} \\ \vdots & \cdots & \vdots \\ a_{n-1}^{(1)} & \ldots & a_{n-1}^{(n-1)} \end{bmatrix} \cdot \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_{n-1} \end{bmatrix}.$$

Note that since $a_i^{(j)}$ is always given modulo θ_i the entries of each column in the above matrix (call it M) actually lie in separate groups $\mathbb{Z}/\theta_i\mathbb{Z}$. However, we may obtain a well-defined matrix power M^N by first computing the power over the integers, and then reducing the i^{th} column modulo θ_i. Then, the action of s^N on T is given by the endomorphism described by the matrix M^N and the $\langle x_1 \rangle$-CSP is simply the problem of recovering N given M^N. Note that this is in general not the same as a matrix DLP because the entries of each column actually lie in separate groups. However, if $\theta_1 = \ldots = \theta_{n-1}$ and M is invertible over $\mathbb{Z}/\theta_1\mathbb{Z}$, we can obtain N by solving the matrix DLP in the subgroup $\langle M \rangle$ of the ring $\mathrm{Mat}(\mathbb{Z}/\theta_1\mathbb{Z})$.

Examples

Holomorphs of Squarefree Cyclic Groups The holomorph of a group H is defined as the natural semidirect product of H with its automorphism group $\mathrm{Aut}(H)$. Let $\mathrm{Hol}(C_p) = C_p \rtimes \mathrm{Aut}(C_p)$ of a cyclic group C_p of prime order p, generated by g. $\mathrm{Aut}(C_p) \cong \mathbb{Z}_p^\times$ is cyclic so $\mathrm{Hol}(C_p)$ is a polycyclic group with two generators. The action of $\mathbb{Z}_p^\times$ on C_p is written as a conjugation: $k^{-1}h^i k = h^{ik},\ k \in \mathbb{Z}_p^\times,\ i \in \mathbb{Z}$.

Given conjugate elements $g_1 = h^l k_1$, $g_2 = h^n k_2$ in G, suppose we want to find $g = h^m k$ such that $g^{-1} g_1 g = g_2$. It is easy to check that $(h^m k)^{-1}(h^l k_1)(h^m k) = h^W (k^{-1} k_1 k)$ with $W = k((-m + l) + m k_1^{-1})$. Subsequently we get $h^W = h^n$ and $(k^{-1} k_1 k) = k_2$. The latter equation is trivial, so one only needs to solve $n = k((k_1{}^{-1} - 1)m + l) \pmod p$ for m and k.

If $k_1 \neq 1 \pmod p$ then $k_1^{-1} - 1 \in \mathbb{Z}_p^\times$ and for any k we find $m = \frac{1}{k_1{}^{-1}-1}(\frac{n}{k} - l)$. If $k_1 = 1 \pmod p$ then $lk = n \pmod p$. If $l = 0 \pmod p$ then $n = 0 \pmod p$ and both m and k take any value (here $g_1 = g_2 = 1$). If $l \neq 0 \pmod p$ then $k = l^{-1} n$ and m takes any value.

We remark here that $\mathrm{Hol}(C_p)$ embeds into the ring $\mathrm{Mat}_2(\mathbb{F}_p)$ of 2×2 matrices over $\mathbb{F}_p$, so the above solution has an equivalent matrix formulation. This solution may

also easily be generalized to abelian groups of squarefree order, for which there is a direct decomposition into cyclic factors, computable in polynomial time [18].

Generalized Quaternions A generalized quaternion group is a finite polycyclic group given by the presentation

$$Q_{2^n} = \langle x, y \mid x^N = 1, y^2 = x^{N/2}, yx = x^{-1}y, N = 2^{n-1} \rangle. \tag{3.3}$$

Clearly, any element in this group has a normal form $x^i y^j$, where $0 \leq i \leq N$, $0 \leq j \leq 1$.

One easily derives the relation $y^j x^i = x^{i(-1)^j} y^j$ for all $i, j \in \mathbb{Z}$. Suppose we have a CSP instance $(x^i y)^{-1}(x^a y)(x^i y) = x^A y$ and want to solve for i. We have, $(y^{-1}x^{-i})(x^a y)(x^i y) = y^{-1}x^{a-2i}y^2 = x^{2i-a}y$. Thus, the exponent i is found by solving $2i - a = A \pmod N$. Note that $(x^i y)^{-1}(x^a)(x^i y) = y^{-1}x^a y = x^{-a}$, so in this case any value of i is a valid solution. Similarly, since x lies in the center of Q_{2^n}, the $\langle x \rangle$-restricted CSP is trivial.

Decomposition in Q_{2^n} It is easy to check that in Q_{2^n} this collection method can be used to solve other problems related to the CSP, like the decomposition problem of [92]. In [93], the following protocol was proposed for key exchange in the platform Q_{2^n}. The underlying problem is a variation of the decomposition problem of [92]. As shown below, this protocol is easily broken by collection in Q_{2^n} and solving linear equations $\mod N$.

Protocol 3.3. The public parameters are $G = Q_{2^n}$ given by (3.3) and subgroups $A_1, A_2 \subseteq \langle x \rangle$.

1. (a) Alice picks secret elements $a \in G$, $b_1, b_2 \in A_1$ and sends $x_1 = b_1 a b_2$ to Bob.
   (b) Bob picks secret elements $d_i \in A_2$ and sends $x_2 = d_1 x_1 d_2$ to Alice.
   (c) Alice sends $x_3 = b_1^{-1} x_2 b_2^{-1} (= d_1 a d_2)$ to Bob.
2. (a) Bob picks a secret element $c \in G$ and sends $y_1 = d_1 c d_2$ to Alice.
   (b) Alice sends $y_2 = b_1 y_1 b_2$ to Bob.
   (c) Bob sends $y_3 = b_1 c b_2 (= d_1^{-1} y_2 d_2^{-1})$ to Alice.

The shared secret is $b = ac = a(b_1^{-1} y_3 b_2^{-1}) = (d_1^{-1} x_3 d_2^{-1})c$.

Proposition 3.2. Protocol 3.3 can be broken in polynomial time by retrieving a and c, which reduces to a system of linear equations over $\mathbb{Z}_N$.

Proof. We assume that $a \notin \langle x \rangle$, since otherwise, finding a is trivial. Similarly, assume $c \notin \langle x \rangle$. Write $a = x^A y$, $c = x^C y$, $d_i = x^{D_i}$, $b_i = x^{B_i}, 1 \leq i \leq 2$. We have, by collection,

$$x_1 = x^{n_1} y = x^{B_1 + A - B_2} y, \quad x_2 = x^{n_2} y = x^{D_1 + n_1 - D_2} y, \quad x_3 = x^{n_3} y = x^{D_1 + A - D_2} y,$$
$$y_1 = x^{m_1} y = x^{D_1 + C - D_2} y, \quad y_2 = x^{m_2} y = x^{B_1 + m_1 - B_2} y, \quad y_3 = x^{m_3} y = x^{B_1 + C - B_2} y.$$

The adversary sees the x_i's and y_i's, and thus also the m_i's and n_i's, for $1 \leq i \leq 3$, and needs to solve linear equations for A, C, D_i's, and B_i's. The result is now clear. □

Observe that the discussion of this example is applicable in any group of the form $\langle x \rangle \rtimes \langle y \rangle, y^2 \in \langle x \rangle$. Another notable example is the dihedral group D_{2n}.

Using matrix representations of polycyclic groups

Let G be given by the presentation (3.1). It is known that every polycyclic group is linear, and thus embeds faithfully into a matrix group over some field. More precisely, there exists $m > 0$, a field $\mathbb{F}$ and an injective homomorphism $\phi : G \to \mathrm{GL}_m(\mathbb{F})$.

Suppose that this matrix representation is used by the designer of the cryptosystem to hide the structure of G. The public parameters are the generator matrices $M_i = \phi(a_i)$ and a base matrix M_x, and operations take place in $\mathrm{GL}_m(\mathbb{F})$. Let M_y denote one of the conjugated public keys. Note that given the generator matrices, an adversary can compute their orders in polynomial time using binary search. However, to reconstruct the presentation of G and reduce the problem back to the CSP in G, they are faced with the following problem: given a matrix $X \in \langle M_1, \ldots, M_n \rangle$, find integers $(i_1, \ldots, i_n)$ such that $X = M_1^{i_1} \ldots M_n^{i_n}$. Clearly, by solving $\mathcal{O}(n^2)$ instances of this problem they can compute the presentation of G and the words representing M_x and M_y, thereby reducing the problem back to the CSP in G.

This problem has been discussed in [51], and is called the Generalized Discrete Logarithm Problem (GDLP). The thesis [48] discusses some square-root type algorithms for the GDLP in finite matrix groups. The case $n = 2$ has been discussed for general finite groups in [61] and [50], both of which show a square root algorithm to reduce the GDLP to at most two DLPs. Observe that in Q_{2^n}, this process introduces a single matrix DLP into the protocol: an adversary sees matrices $A = M_x^i M_y$ or $A = M_x^i$, and so can recover i by solving one of the matrix DLPs $AM_y^{-1} = M_x^i$, $A = M_x^i$. Therefore, in general using the matrix representation likely does not offer any novel security feature, though it may enhance the overall security.

Now, suppose that the original problem is given as a CSP in G and the adversary is able to efficiently compute a faithful representation $\phi : G \to \mathrm{GL}_m(\mathbb{F}_q)$ as well as its inverse. Denote by $x \in G$ the public base element and $y = g^{-1}xg \in G$ the public key. Then, it suffices for the adversary to find a matrix $M_g \in \phi(G)$ such that $M_g^{-1}\phi(x)M_g = \phi(y)$, so solving a CSP in $\phi(G)$ breaks the system. In fact, if the original CSP in G is an A-restricted CSP for $A \leq G$ cyclic, then it is not even necessary to compute ϕ^{-1} since the secret here is essentially an integer. In Theorem 3.1, we will see that an A-restricted CSP in $\mathrm{GL}_m(\mathbb{F})$ is reducible to a set of $\mathcal{O}(m^2)$ DLPs over $\mathbb{F}$. So, any system with a linear platform must ensure that the subgroup from which conjugators are chosen has at least two generators. We remark here that in the case where such an efficient representation ϕ and its inverse are available, several attacks exist to directly retrieve the shared key, see [12, 54, 65, 103].

3.2.2 Matrix Groups

Throughout this section, q denotes a power of a prime p and $\mathbb{F}_q$ denotes the finite field with q elements, and $\mathrm{ord}(a)$ denotes the multiplicative order of an element $a \in \mathbb{F}_q^*$.

Matrix groups over finite fields have served as platform groups for several proposed protocols. In [60] and [32], the DLP over the matrix group $\mathrm{GL}_n(\mathbb{F}_q)$ was studied and shown to be no more difficult than the DLP over a small extension of $\mathbb{F}_q$, and consequently less efficient in terms of key sizes for the same security level. Most known non-abelian platform groups are linear, i.e. they embed faithfully into a matrix group. If this embedding and its inverse can efficiently be computed by an adversary, the security of the system depends on that of the matrix CSP rather than that in the original platform.

It is then natural and important to study the complexity of the CSP over matrix groups. Several attacks exist to directly retrieve the shared key from CSP-based protocols without computing the secret keys [12, 54, 65, 103]. However, to the best of our knowledge, the CSP and its variants have not been investigated in this platform. In this section, we study the A-restricted matrix CSP for a cyclic subgroup $A \subseteq \mathrm{GL}_n(\mathbb{F}_q)$. Here, for maximum generality we also allow the base element be a non-invertible matrix. In other words, we provide a cryptanalysis of Protocol 3.4 over a ring R, for the case $R = \mathrm{Mat}_n(\mathbb{F}_q)$ of $n \times n$ matrices over $\mathbb{F}_q$.

Protocol 3.4. Let $X \in R$ and $Z \in R^*$ be public elements.

1. Alice picks a secret integer $r \in \mathbb{Z}$ and publishes $Z^{-r}XZ^r$.
2. Bob picks a secret integer $s \in \mathbb{Z}$ and publishes $Z^{-s}XZ^s$.

3. The shared secret is $Z^{-s-r}XZ^{s+r}$.

In subsection 1, we provide a polynomial time reduction of recovering $r \in \mathbb{Z}$ to a set of $\mathcal{O}(n^2)$ DLPs. In subsection 2, we show how this enables a full cryptanalysis of the system proposed in [105].

Remark 3.2. Similarly to [32] (also mentioned in [60]) we will use an easy generalization of the Chinese Remainder Theorem (CRT) for systems of equations of the form $x \equiv x_i \pmod{\theta_i}, 1 \leq i \leq s$, where the θ_i are not necessarily coprime, but a solution is required $\mod \theta := \operatorname{lcm}_i \theta_i$. Write $\theta = p_1^{e_1} \dots p_t^{e_t}$ as the prime factorization of θ. For each $j \in \{1, \dots, t\}$, let $\emptyset \neq \mathcal{I}_j \subseteq \{1, \dots, s\}$ denote the list of indices i such that $p_j^{e_j} \mid \theta_i$. If a solution x to the original system of equations exists we have $x_i - x_{i'} = 0 \pmod{p_j^{e_j}}$ for each $i, i' \in \mathcal{I}_J$ and $x = x_{i_0} \pmod{p_j^{e_j}}$ for any $i_0 \in \mathcal{I}_j$. So if a solution exists, we can translate the original system to one where the moduli are coprime, and so can be solved with the CRT.

$\langle Z \rangle$-restricted CSP in $\mathrm{GL}_n(\mathbb{F}_q)$

Suppose that $Z^{-r}XZ^r = Y$, the adversary sees X, Z, and $Y \in \mathrm{Mat}_n(\mathbb{F}_q)$, the integer r is secret. There exists an extension $\mathbb{F}_{q^k}$ and a unique matrix $P \in \mathrm{GL}_n(\mathbb{F}_{q^k})$ (computable in polynomial time, by Algorithm 1 in [60]) such that $J_Z = PZP^{-1}$, where J_Z is the Jordan Normal form of Z. Here $\mathbb{F}_{q^k}$ is the smallest extension containing all eigenvalues of Z, and k is polynomial in q and n [60]. Let θ_Z be the order of Z in the group $\mathrm{GL}_n(\mathbb{F}_q)$ and θ_J be the order of J_Z in the group $\mathrm{GL}_n(\mathbb{F}_{q^k})$. Then, clearly since $J_Z = PZP^{-1}$ we have $\theta_Z = \theta_J$. Further, we only require a value of r modulo θ_Z to break the system.

Consider $M = PXP^{-1}$ and $N = PYP^{-1}$. Note that these are both computable by the adversary. It is easily verified that $Z^{-r}XZ^r = Y \iff J_Z^{-r}MJ_Z^r = N$. Thus, to recover r we may assume that Z is already in Jordan form. We divide the rest of the analysis into two cases. We first consider the case where Z is diagonalizable.

Case: J_Z is diagonal.

We write $M = (M_{i,j})_{n \times n}$ and $N = (N_{i,j})_{n \times n}$.

Theorem 3.3. If $J_Z = \begin{pmatrix} d_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & d_n \end{pmatrix}$ is diagonal then the retrieval of r in Protocol 3.4 reduces to solving at most n^2 DLPs over $\mathbb{F}_{q^k}$.

Proof. Note that $d_i \neq 0 \ \forall\ i$ since J_Z is invertible. We expand $J_Z^{-r} M J_Z^r = N$:

$$\begin{pmatrix} d_1^r & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & d_n^r \end{pmatrix} \begin{pmatrix} M_{11} & \dots & M_{1n} \\ \vdots & \ddots & \vdots \\ M_{n1} & \dots & M_{nn} \end{pmatrix} \begin{pmatrix} d_1^{-r} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & d_n^{-r} \end{pmatrix} = \begin{pmatrix} N_{11} & \dots & N_{1n} \\ \vdots & \ddots & \vdots \\ N_{n1} & \dots & N_{nn} \end{pmatrix}$$

$$\iff M_{ij}(d_i d_j^{-1})^r = N_{ij},\ 1 \leq i, j \leq n. \tag{3.4}$$

By assumption M and N are nonzero matrices, so there exists at least one pair of indices (i, j) such that $M_{ij} \neq 0$, $N_{ij} \neq 0$. For any such pair we have

$$(d_i d_j^{-1})^r = M_{ij}^{-1} N_{ij},$$

so $r \pmod{\operatorname{ord}(d_i d_j^{-1})}$ is found by solving a DLP in $\mathbb{F}_{q^k}$. Repeating this for all such pairs (i, j), we may compute r modulo

$$\operatorname{lcm}_{\exists (i,j) | M_{ij} \neq 0}(\operatorname{ord}(d_i^{-1} d_j)) = \operatorname{lcm}_{d_i \neq 0}(\operatorname{ord}(d_i)) = \theta_Z$$

using the generalized Chinese Remainder Theorem (see Remark 3.2). □

Case: J_Z is not diagonal

Suppose that $J_Z = \begin{pmatrix} J_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & J_s \end{pmatrix}$ is the Jordan-Normal form of Z, where each $J_i = \begin{pmatrix} \lambda_i & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \\ 0 & 0 & \dots & \lambda_i \end{pmatrix}$, $1 \leq i \leq s$, is a $d_i \times d_i$ Jordan block corresponding to the eigenvalue $\lambda_i \in \mathbb{F}_{q^k}$, and $d_i > 1$ for at least one i. Denoting $\binom{-n}{k} := (-1)^k \binom{n+k-1}{k}$, and with the convention $\binom{r}{m} = 0$ if $m < r$ and $r > 0$, we have by induction, for $r \geq 1$,

$$J_i^r = \begin{pmatrix} \lambda_i^r & \binom{r}{1}\lambda_i^{r-1} & \dots & \binom{r}{d_i-1}\lambda_i^{r-d_i+1} \\ 0 & \lambda_i^r & \dots & \binom{r}{d_i-2}\lambda_i^{r-d_i+2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \binom{r}{1}\lambda_i^{r-1} \\ 0 & 0 & \dots & \lambda_i^r \end{pmatrix},\ J_i^{-r} = \begin{pmatrix} \lambda_i^{-r} & \binom{-r}{1}\lambda_i^{-r-1} & \dots & \binom{-r}{d_i-1}\lambda_i^{-r-d_i+1} \\ 0 & \lambda_i^{-r} & \dots & \binom{-r}{d_i-2}\lambda_i^{-r-d_i+2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \binom{-r}{1}\lambda_i^{-r-1} \\ 0 & 0 & \dots & \lambda_i^{-r} \end{pmatrix}$$

More concisely, for any $1 \leq i \leq s$, $r \in \mathbb{Z}$, $(J_i^r)_{(k,l)} = \binom{r}{l-k}\lambda_i^{r-l+k}$, $0 \leq k, l \leq d_i$. Now, we write M and N as block matrices with s^2 $d_i \times d_j$ blocks $(M_{i,j})_{d_i \times d_j}$, $(N_{i,j})_{d_i \times d_j}$:

$$M = \begin{pmatrix} (M_{1,1})_{d_1 \times d_1} & \dots & (M_{1,s})_{d_1 \times d_s} \\ \vdots & \ddots & \vdots \\ (M_{s,1})_{d_s \times d_1} & \dots & (M_{s,s})_{d_s \times d_s} \end{pmatrix}, \quad N = \begin{pmatrix} (N_{1,1})_{d_1 \times d_1} & \dots & (N_{1,s})_{d_1 \times d_s} \\ \vdots & \ddots & \vdots \\ (N_{s,1})_{d_s \times d_1} & \dots & (N_{s,s})_{d_s \times d_s} \end{pmatrix}.$$

The next result reduces the problem of recovering r from Protocol 3.4 to a set of s^2 matrix equations involving Jordan blocks.

Lemma 3.6. $J_Z^{-r} M J_Z^r = N \iff J_i^{-r} M_{i,j} J_j^r = (N_{i,j})_{d_i \times d_j}\ \forall i,j,\ 1 \le i,j \le s.$

Proof. The result is clear from the observation that

$$J_Z^{-r} M J_Z^r = \begin{pmatrix} J_1^{-r} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & J_s^{-r} \end{pmatrix} \begin{pmatrix} M_{11} & \dots & M_{1s} \\ \vdots & \ddots & \vdots \\ M_{s,1} & \dots & M_{s,s} \end{pmatrix} \begin{pmatrix} J_1^{r} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & J_s^{r} \end{pmatrix}$$
$$= \begin{pmatrix} J_1^{-r} M_{11} J_1^r & \dots & J_1^{-r} M_{1s} J_s^r \\ \vdots & \ddots & \vdots \\ J_s^{-r} M_{s1} J_1^r & \dots & J_s^{-r} M_{ss} J_s^r \end{pmatrix}.$$

□

Lemma 3.7. Writing $M_{i,j} = (m_{l,k}^{(i,j)})$, $N_{i,j} = (n_{l,k}^{(i,j)})$, we have $J_Z^{-r} M J_Z^r = N \iff$

$$n_{f,h}^{(i,j)} = \sum_{k=1}^{h} \sum_{l=f}^{d_i} \binom{-r}{l-f} \binom{r}{h-k} m_{l,k}^{(i,j)} \lambda_i^{-r+f-l} \lambda_j^{r-h+k}$$

$\forall f,h$ with $1 \le f \le d_i,\ 1 \le h \le d_j\ \forall i,j,\ 1 \le i,j \le s.$

Proof. Recall that $(J_i^r)_{(k,l)} = \binom{r}{l-k} \lambda_i^{r-l+k)}$ for any integer r and any $1 \le k,l \le d_i$. We compute the $(f,h)^{\text{th}}$ term of $J_i^{-r} M_{ij} J_j^r$. Note that for $1 \le g \le d_i$ the $(f,g)^{\text{th}}$ term of $J_i^{-r} M_{i,j}$ is given by $\sum_{l=f}^{d_i} \binom{-r}{l-f} \lambda_i^{-r+f-l} m_{l,g}^{(i,j)}$. Thus, the $(f,h)^{\text{th}}$ term of $N_{i,j} = J_i^{-r} M_{i,j} J_j^r$ is

$$n_{f,h}^{(i,j)} = \sum_{g=1}^{h} \sum_{l=f}^{d_i} \binom{-r}{l-f} \binom{r}{h-g} m_{l,g}^{(i,j)} \lambda_i^{-r+f-l} \lambda_j^{r-h+g}. \tag{3.5}$$

For the equality $J_i^{-r} M_{ij} J_j^r = N_{ij}$ we require $n_{f,h}^{(i,j)} = m_{f,h}^{(i,j)}$ for all indices f,h with $1 \le f,h \le d_i$. The result now follows from Lemma 3.6. □

Now, by assumption, $M \neq 0$. So, we may choose the largest index $l_0^{(i,j)}$ such that the $l_0^{(i,j)\text{ th}}$ row of $M_{i,j}$ has at least one nonzero term. Let $g_0^{(i,j)}$ be the smallest column index such that $m_{l_0,g_0}^{(i,j)} \neq 0$. We have, $m_{l,g}^{(i,j)} = 0\ \forall\ l,g$ such that $l > l_0, 1 \le g \le d_j$, and $m_{l_0,g}^{(i,j)} = 0\ \forall\ g$ such that $g < g_0$.

Taking $f = l_0, h = g_0 + 1$, by Equation (3.5) the $(f,h)^{th}$ term of $N_{i,j} = J_Z^{-r} M_{i,j} J_Z^r$ is given by

$$\begin{aligned} n_{l_0,g_0+1}^{(i,j)} &= \sum_{g=1}^{g_0+1} \sum_{l=l_0}^{d_i} \binom{-r}{l-l_0} \binom{r}{g_0+1-g} m_{l,g}^{(i,j)} \lambda_i^{-r+l_0-l} \lambda_j^{r-g_0-1+g} \\ &= \sum_{g=g_0}^{g_0+1} \binom{r}{g_0+1-g} m_{l_0,g}^{(i,j)} \lambda_i^{-r} \lambda_j^{r-g_0-1-g} \\ &= r m_{l_0,g_0}^{(i,j)} \lambda_j (\lambda_i^{-1}\lambda_j)^r + m_{l_0,g_0+1}^{(i,j)} (\lambda_i^{-1}\lambda_j)^r \end{aligned} \tag{3.6}$$

In particular, for $i = j$, we have the equation $r m_{l_0,g_0}^{(i,j)} \lambda_j^{-1} + m_{l_0,g_0+1}^{(i,j)} = n_{l_0,g_0+1}^{(i,j)}$, where by construction, $m_{l_0 g_0}^{(i,j)} \neq 0$. This can be solved to get $r' := r \pmod p$. We thus have the following results.

Proposition 3.3. The value of $r' := r \mod p$ can be computed in polynomial time.

Proposition 3.4. Computing $r \mod \operatorname{lcm}_{1 \le i \le s} \operatorname{ord}(\lambda_i)$ reduces in polynomial time to solving at most s^2 DLPs in $\mathbb{F}_{q^k}$.

Proof. We write $r = r' + pw$ for $w \in \mathbb{Z}$. From equation (3.6), since $m_{l_0,g_0}^{(i,j)} \in \mathbb{F}_{q^k}$, we have

$$m_{l_0,g_0+1}^{(i,j)} = r m_{l_0,g_0}^{(i,j)} \lambda_j (\lambda_i^{-1}\lambda_j)^r + m_{l_0,g_0+1}^{(i,j)} (\lambda_i^{-1}\lambda_j)^r = (\lambda_i^{-1}\lambda_j)^r (r' m_{l_0,g_0}^{(i,j)} \lambda_j + m_{l_0,g_0+1}^{(i,j)}).$$

Equating $(\lambda_i^{-1}\lambda_j)^r (r' m_{l_0,g_0}^{(i,j)} \lambda_j + m_{l_0,g_0+1}^{(i,j)}) = n_{l_0,g_0+1}^{(i,j)}$ for $i \neq j$, we see that r can now be recovered $\pmod{\operatorname{ord}(\lambda_i^{-1}\lambda_j)}$ by solving a DLP. Repeating this process for all pairs (i,j), and using the generalized CRT (see Remark 3.2) we get r modulo $\operatorname{lcm}_{1 \le i < j \le s} \operatorname{ord}(\lambda_i^{-1}\lambda_j) = \operatorname{lcm}_{1 \le i \le s} \operatorname{ord}(\lambda_i)$. □

Computing $r \mod \theta_Z$ As before, θ_Z denotes the order of J_Z in the group $GL(\mathbb{F}_{q^k})$. We will now show how Propositions 3.3 and 3.4 allow us to compute $r \pmod{\theta_Z}$. We have the following result on the value of θ_Z from [60].

Lemma 3.8 ([60]). The order θ_Z of J_Z is $\operatorname{lcm}_{1 \le i \le s} (\lambda_i) \cdot p\{t\}$, where t is the largest Jordan block in J_Z and $p\{t\}$ denotes the smallest power of p greater than or equal to t.

Below, we show that if we can compute $r \pmod p$, then by extension we can find $r \pmod{p\{t\}}$.

Lemma 3.9. Suppose that an Algorithm A returns $r \pmod p$ for an equation of the form $Z^{-r}XZ^r = Y$. Then, for any $v \geq 1$, one may find $r \pmod{p^v}$ with v applications of Algorithm A.

Proof. We write $r = r_0 + r_1 p + \ldots + r_{v-1}p^{v-1} \pmod{p^v}$. With one application of Algorithm A, one finds $r \pmod p = r_0$, and then compute $Z_1 = Z^p$, $Y_1 = Z^{r_0} Y Z^{-r_0}$, $r' = r_1 + r_2 p + \ldots + r_{v-1}p^{v-2}$. Then, we have an equation $Z_1^{-r'} X Z_1^{r'} = Y_1$. One now uses Algorithm A again to find $r' \pmod p = r_1$. In v applications of Algorithm A, we recover $r \pmod{p^v}$. □

Finally, we can prove the final result of this section.

Theorem 3.4. Let J_Z be non-diagonal, and composed of s Jordan blocks. Then, the computation of r from Protocol 3.4 is polynomial time reducible to a set of s^2 DLPs over $\mathbb{F}_{q^k}$.

Proof. From Propositions 3.3 and 3.4, we may compute $r \pmod{\operatorname{lcm}_{1 \leq i \leq s} \lambda_i}$ and $r' = r \pmod p$ with the same time complexity as solving s^2 DLPs in $\mathbb{F}_q$. Note that since the multiplicative order of any element of $\mathbb{F}_{q^k}^{\times}$ divides $q^k - 1$, the values of $r' = r \pmod p$ and $r \pmod{\operatorname{lcm}_{1 \leq i \leq s} \operatorname{ord}(\lambda_i)}$ are obtained independently. Now, write $p\{t\} = p^v \leq q^k$, by Lemma 3.9, we can obtain $r \pmod{p^v}$ from r' in polynomial time $\mathcal{O}(k \log q)$. Combining $r \pmod{p^v}$ and $r \pmod{\operatorname{lcm}_{1 \leq i \leq s} \lambda_i}$ using the Chinese Remainder Theorem, we get $r \pmod{\theta_Z}$, as required. It is also clear that every step apart from the DLPs has polynomial time complexity. □

We may thus summarize the results of Theorems 3.3 and 3.4 as follows.

Corollary 3.1. The $\langle Z \rangle$-restricted CSP in $\mathrm{GL}_n(\mathbb{F}_q)$ reduces in polynomial time to $\mathcal{O}(n^2)$ DLPs over a small extension $\mathbb{F}_{q^k}$ of $\mathbb{F}_q$.

Remark 3.3. While we have only discussed the search variant of the conjugacy problem, it is not hard to show that the arguments of this section also reduce the decisional variant of the CSP to a set of corresponding decisional versions of the DLP and generalized CRT. Further, this may be used, along with a collision-type square-root algorithm, to solve any A-restricted CSP in $\mathrm{GL}_n(\mathbb{F}_q)$, where A is abelian.

Applications to Cryptanalysis

Protocol based on Quaternions $\pmod p$ In [105], a protocol similar to Protocol 3.4 was proposed for the case of another ring $R = H_p$, which we will refer to as the quaternions mod p. Recall that the set of all Lipchitz quaternions is defined as

$L = \{a = a_1 + a_2 i + a_3 j + a_4 k \mid a_1, a_2, a_3, a_4 \in \mathbb{Z}\}$. Similarly, the Hurwitz quaternions are given by $H = \{a = a_1 + a_2 i + a_3 j + a_4 k \mid a_1, a_2, a_3, a_4 \in \mathbb{Z} + \frac{1}{2}\mathbb{Z}\}$.

For a prime p, the authors define H_p as the set $\{a = a_1 + a_2 i + a_3 j + a_4 k \mid a_i \in \mathbb{Z}_p\}$. Addition, multiplication, the norm $||\cdot||$, and conjugates are defined in the usual way as in quaternion algebras (for an exposition on arithmetic in quaternion sets, see [104]), but over the base ring $\mathbb{Z}_p$. The quaternion a is invertible in H_p if and only if $||a|| \neq 0 \pmod{p}$.

Denote by H_p^* the set of all invertible quaternions in H_p. The protocol in [105] is described as follows:

Protocol 3.5.

1. Alice and Bob agree to choose randomly public elements $x \in H_p$, $z \in H_p^*$.
2. Alice picks two secret integers $r, s \in \mathbb{Z}$ such that $1 \leq r \leq p-1$ and $2 \leq s \leq p-1$ and then computes $y_A = z^r x^s z^{-r}$, and sends $y_A \in H_p$ to Bob.
3. Bob picks two secret integers $u, v \in \mathbb{Z}$ such that $1 \leq u \leq p-1$ and $2 \leq v \leq p-1$ and then computes $y_B = z^u x^v z^{-u}$, and sends $y_B \in H_p$ to Alice.
4. Alice computes $K_A = z^r {y_B}^s z^{-r}$ as the shared session key.
5. Bob computes $K_B = z^u {y_A}^v z^{-u}$ as the shared session key.

We now show how the cryptanalysis for this protocol reduces to breaking Protocol 3.4 for $R = \mathrm{Mat}_2(\mathbb{F}_p)$. First, by the arguments in [104], we have $H/pH = L/pL = H_p$. Now, by Proposition 3.3 of [104], for any integers a and b satisfying $a^2 + b^2 \equiv -1 \pmod{p}$, the map

$$\phi_{a,b} : H/pH \to \mathrm{Mat}_2(\mathbb{Z}/p\mathbb{Z})$$
$$\gamma_1 + \gamma_2 i + \gamma_3 j + \gamma_4 k \mapsto \begin{pmatrix} \gamma_1 + \gamma_2 a + \gamma_4 b & \gamma_3 + \gamma_4 a - \gamma_2 b \\ -\gamma_3 + \gamma_4 a - \gamma_2 b & \gamma_1 - \gamma_2 a - \gamma_4 b. \end{pmatrix}$$

is an isomorphism of rings. Clearly, the inverse of $\phi_{a,b}$ is also easily computed as

$$\begin{aligned} &\phi_{a,b}^{-1}\left(\begin{pmatrix} A & B \\ C & D \end{pmatrix}\right) \\ =&\frac{1}{2}(A+D) + \frac{1}{2}\left(b(B+C) - a(A-D)\right) i + \frac{1}{2}(B-C) j - \frac{1}{2}\left(a(B+C) + b(A-D)\right) k \end{aligned}$$

Thus, Protocol 3.5 may be treated as if it is over $\mathrm{Mat}_2(\mathbb{F}_p)$. Now, note that the public keys are of the form $y = z^{-r} x^s z^r$ where r and s are private integers, and $x \in \mathrm{Mat}_2(\mathbb{F}_p)$, $z \in \mathrm{GL}_2(\mathbb{F}_p)$ are public matrices. The only difference now with Protocol 3.4 is that this scheme has two secret integers instead of one. However,

we observe that the presence of two secret integers weakens the scheme, because an adversary can break the system if they find any pair of solutions (r, s) such that $K_A = z^r x^s x^{-r}$. So, it makes sense to fix s, without loss of generality, to $s = 1$. Thus, Corollary 3.1 for the case $n = 2$ now shows that finding r and breaking Protocol 3.5 reduces to at most four DLPs.

Subgroup Conjugacy Search in Matrix Groups In [41], the authors introduce what they call the Subgroup Conjugacy Search Problem (SCSP) in a non-abelian group G, and propose a protocol based on it, suggesting as potential platforms the matrix group $\mathrm{GL}_n(\mathbb{F}_q)$ and a subgroup of it. We first note that the SCSP corresponds exactly to the A-restricted CSP defined in this chapter, for a cyclic subgroup A of G. Therefore, Corollary 3.1 gives a direct cryptanalysis of this protocol, reducing its security to that of a set of $\mathcal{O}(n^2)$ DLPs over a small extension of $\mathbb{F}_q$.

While the authors state that the SCSP is at least as hard as the CSP, we remark that this is likely not true in general. In Section 3.2.1 we showed that in finite polycyclic groups with two generators, a well-chosen SCSP is harder than the CSP, whereas by Remark 3.1 the SCSP in certain infinite polycyclic groups with two generators groups is seemingly easier than the CSP. Similarly, Section 3.2.2 gives a complete reduction of the SCSP in $\mathrm{GL}_n(\mathbb{F}_q)$ to a set of DLPs, but a solution to the general CSP is unclear. This may be intuitively realized, since while there is an added constraint in the SCSP, the adversary also has more information on where to search for the conjugator.

3.3 CSP in Central Products

In this section, we study the complexity of the CSP in so-called central products of groups. We define a property called efficient C-decomposability of a group and show that an instance of the CSP in a central product $G = HK$ reduces to two separate instances of the CSP in H and K, if G satisfies this property. We also provide examples of groups which possess this property, and apply this concept to demonstrate the solution of the CSP in all extraspecial p-groups.

The results on central products also demonstrate that while considering any platform group for a CSP-based system, care must be taken to ensure that an efficient decomposition into a central product is not possible. This is practically significant for future work in group-based cryptography since several non-abelian groups, and in particular, p-groups, are constructed by combining smaller p-groups by taking direct, central, and semidirect products (see, for example, [15, 25]).

Throughout, we denote by p a prime number, C_p the cyclic group of order p, and

by $H \rtimes K$ a semidirect product of groups H and K (with K acting on H by automorphisms). The center of a group G is denoted by $Z(G)$. For a subset S and an element x of a group G we use the notations $xS := \{xz \mid z \in S\}$ and $Sx := \{zx \mid z \in S\}$. For any $x \in G$, denote by $C_x := \{g^{-1}xg \mid g \in G\}$ the conjugacy class of x. For subsets S_1 and S_2, the product S_1S_2 denotes the set $\{s_1s_2 \mid s_1 \in S_1,\ s_2 \in S_2\}$. An algorithm on a finite group G will be said to be polynomial time if its time complexity is $\mathcal{O}(\log|G|)$.

Definition 3.6. A group G is said to be a central product of its subgroups H and K if every element $g \in G$ can be written as hk, with $h \in H,\ k \in K$ (i.e. $G = HK$), and we have $hk = kh\ \forall\ h \in H,\ k \in K$ (thus, $H \cap K \subseteq Z(G)$).

Below, we introduce the concept of efficient C-decomposability.

Definition 3.7. A finite group G is said to be efficiently C-decomposable if for any elements $h, k, x, y \in G$ with $hC_x \cap kC_y \neq \emptyset$, an element of $hC_x \cap kC_y$ can be found in polynomial time.

The result below shows that in an efficiently decomposable group G which is also a central product of its subgroups H and K, the CSP in G can be reduced to the CSP in the individual component subgroups H and K.

Theorem 3.5. Let G be an efficiently C-decomposable group and H and K be subgroups of G such that $G = HK$ is a central product. Then, solving the CSP in G is polynomial time reducible to solving two separate CSP's in H and K.

Proof. Let $\tilde{g} = \tilde{h}\tilde{k}$, $g' = h'k'$ be elements in G. Suppose that we want to solve the CSP for $\tilde{g}$ and g', i.e. to find an element $g = hk$ such that $g^{-1}\tilde{g}g = g'$. We have,

$$\begin{aligned} & (hk)^{-1}(\tilde{h}\tilde{k})(hk) = k^{-1}h^{-1}(\tilde{h}\tilde{k})hk = (h^{-1}\tilde{h}h)(k^{-1}\tilde{k}k) \\ \implies & (h^{-1}\tilde{h}h)(k^{-1}\tilde{k}k) = h'k' \\ \implies & h'^{-1}(h^{-1}\tilde{h}h) = k'(k^{-1}\tilde{k}k)^{-1} \in h'^{-1}C_{\tilde{h}} \cap k'C_{\tilde{k}^{-1}} \subseteq H \cap K \end{aligned} \tag{3.7}$$

Note that $h'^{-1}(h^{-1}\tilde{h}h) = k'(k^{-1}\tilde{k}k)^{-1} \in h'^{-1}C_{\tilde{h}} \cap k'C_{\tilde{k}} \subseteq H \cap K$. By hypothesis, we can find, in polynomial time, an element $t \in h'^{-1}C_{\tilde{h}} \cap k'C_{\tilde{k}^{-1}}$.

Now consider the following two separate instances of the CSP in H and K.

$$h^{-1}\tilde{h}h = h't \in H \text{ and } (k^{-1}\tilde{k}k) = k't^{-1} \in K \tag{3.8}$$

Suppose that we have solutions h and k of (3.8). Then, for $g = hk$, we have $g^{-1}\tilde{g}g = (hk)^{-1}(\tilde{h}\tilde{k})(hk) = (h^{-1}\tilde{h}h)(k^{-1}\tilde{k}k) = (h't)(k't^{-1}) = h'k'$. Thus, $g = hk$ is a solution to $g^{-1}\tilde{g}g = g'$. We conclude that $\tilde{g}$ and g' are conjugate if and only if $h'^{-1}C_{\tilde{h}} \cap k'C_{\tilde{k}} \neq \emptyset$, and that in this case, a conjugator can be found by solving (3.8). □

Examples

We present two examples of efficiently decomposable groups below. Before this, we recall the following well-known number theoretic fact. The modular linear equation $ax \equiv b \pmod{n}$ in x has a solution if and only if $\gcd(a, n) \mid b$. When a solution exists, it may be found in time $\mathcal{O}(\log n)$.

Example 3.1. The dihedral groups D_n are all efficiently C-decomposable. Consider D_n, which has the presentation $\langle x, y \mid x^n = 1, y^2 = 1, yx = x^{-1}y\rangle$. Then we have $y^j x^i = x^{i(-1)^j} y^j$ for all i, j,

$$C_{x^A y^B} = \{x^{i+A(-1)^j - i(-1)^B} y^B \mid 0 \le i \le n-1,\ 0 \le j \le 1\}$$

$$(x^k y^l) C_{x^A y^B} = \{x^{k+(-1)^l[i+A(-1)^j - i(-1)^B]} y^{l+B} \mid 0 \le i \le n-1,\ 0 \le j \le 1\}$$

Write $h = x^{k_1} y^{l_1}, u = x^{A_1} y^{B_1}, k = x^{k_2} y^{l_2}, v = x^{A_2} y^{B_2}$. It is then easily verified that finding a common element of hC_u and kC_v amounts to finding (i_1, i_2, j_1, j_2) from an equation of the form

$$k_1 + (-1)^{l_1}[i_1 + A_1(-1)^{j_1} - i_1(-1)^{B_1}] = k_2 + (-1)^{l_2}[i_2 + A_2(-1)^{j_2} - i_2(-1)^{B_2}] \pmod{n}$$

Clearly, one may choose values of j_1 and j_2 and then find a linear relation between i_1 and i_2 by solving a linear modular equation. Thus, a solution can be found in time $\mathcal{O}(\log n)$.

Example 3.2. The generalized quaternion groups Q_{2^n} are all efficiently C-decomposable. A generalized quaternion group is given by the presentation

$$Q_{2^n} = \langle x, y \mid x^N = 1, y^2 = x^{N/2}, yx = x^{-1}y, N = 2^{n-1}\rangle. \tag{3.9}$$

We derive the relations $x^i y = y x^{N-i}$ and $y x^i = x^{N-i} y$. Using these one easily derives the relation $y^j x^i = x^{i(-1)^j} y^j$ for all $i, j \in \mathbb{Z}$.

$$(x^k y^l) C_{x^A y^B} = \{x^{k+(-1)^{j+l}[A - i + i(-1)^B]} y^{B+l} \mid 0 \le i \le N,\ 0 \le j \le 1\}$$

Write $h = x^{k_1} y^{l_1}, u = x^{A_1} y^{B_1}, k = x^{k_2} y^{l_2}, v = x^{A_2} y^{B_2}$. Then finding a common element of hC_u and kC_v amounts to solving an equation of one of the following forms for (i_1, i_2, j_1, j_2):

$$k_1 + (-1)^{j_1+l_1}[A_1 - i_1 + i_1(-1)^{B_1}] = k_2 + (-1)^{j_2+l_2}[A_2 - i_2 + i_2(-1)^{B_2}] \pmod{N}$$

Again, one may choose values of j_1 and j_2 and then find a linear relation between i_1 and i_2 by solving a linear modular equation. Thus, a solution can be found in time $\mathcal{O}(\log N)$.

Remark 3.4. Consider the group S_n of permutations on n elements. If x and $y \in S_n$ are cycles, then finding an element of $hC_x \cap kC_y$ for any $h, k \in S_n$ can be done in polynomial time. This follows from Theorem 5 in [44], by taking $\sigma = h^{-1}k$ and noting that the result used at each step is in fact constructive and achievable in polynomial time. However, for x and y general permutations, a procedure to find an element in $hC_x \cap kC_y$ is not, in the author's knowledge, so far clear.

Extraspecial p-groups are efficiently C-decomposable

A finite group with order a power of a prime p is called a p-group. Some p-groups have been sporadically suggested as platforms for cryptography. For example, in [42], authentication and signature schemes using the CSP were proposed, and general p-Miller groups were suggested as platforms.

Definition 3.8 (Extraspecial p-group). A p-group G is called extraspecial if its center $Z(G)$ is cyclic of order p, and the quotient $G/Z(G)$ is a non-trivial elementary (i.e. every element has order p) abelian p-group.

The following results are standard, and can be found, for instance, in [38], [21].

Theorem 3.6. There are two isomorphism classes for the extra special group of order p^3: $M(p) = C_{p^2} \rtimes C_p$ and $N(p) = (C_p \times C_p) \rtimes C_p$ with presentations

$$M(p) = \langle x, y \mid x^{p^2} = 1, y^p = 1, yxy^{-1} = x^{1+p} \rangle \tag{3.10}$$

$$N(p) = \langle x, y, z \mid x^p = y^p = z^p = 1, xy = yx, yz = zy, zxz^{-1} = xy^{-1} \rangle \tag{3.11}$$

Remark 3.5. We observe that given the generators of an extraspecial group of order p^3, one can find, in polynomial time, which of the two isomorphism classes this group belongs to, simply by raising each generator to the power p.

Theorem 3.7. Any central product G of finitely many copies of $N(p)$ and $M(p)$ is efficiently C-decomposable.

Proof. We first deal with the case $G = M(p)$. By induction, one can easily show that $y^j x^i = x^{i(1+p)^j} y^j = x^{i(1+jp)} y^j \ \forall\ i, j \in \mathbb{Z}$. By recursively using this formula, we have

$$(x^i y^j)^{-1}(x^a y^b)(x^i y^j) = x^{(a-i)(1-jp)} y^{b-j} x^i y^j = x^{a-p(aj-bi)} y^b. \tag{3.12}$$

We have for $g' = x^a y^b$ and $g = x^c y^d$

$$gC_{g'} = \{(x^c y^d)(x^{a-p(aj-bi)} y^b) \mid i, j \in \mathbb{Z}\} = \{x^{c+a-p(aj-bi-ad)} y^{d+b}\} \mid i, j \in \mathbb{Z}\}$$

Writing $g_i = x^{c_i}y^{d_i}$, $g_i' = x^{a_1}y^{b_1}$, $i = 1,\ 2$. It is clear that if an element common to the sets $g_1C_{g_1'}$ and $g_2C_{g_2'}$ exists, then we must have $b_1 + d_1 = b_2 + d_2 \pmod p$, $c_1 + a_1 = c_2 + a_2 \pmod p$ and such an element may be found by solving

$$\frac{1}{p}[(c_1 + a_1) - (c_2 + a_2)] = (a_1j_1 - b_1i_1 - a_1d_1) - (a_2j_2 - b_2i_2 - a_2d_2) \pmod p$$

for i_1, i_2, j_1, j_2. Clearly, this is a linear equation, so a solution can be found in time $\mathcal{O}(\log p)$.

Now we consider $G = N(p)$. By induction, one can show that $z^kx^az^{-k} = x^ay^{-ka}$ for all $a,\ k \in \mathbb{Z}$. Thus, $z^kx^ay^b = x^ay^{-kab}z^k$ for all $a,\ k \in \mathbb{Z}$. Write $g = x^ay^bz^c$, $g' = x^Ay^Bz^C$, and $h = x^iy^jz^k$. We have,

$$\begin{aligned}(x^iy^jz^k)^{-1}(x^ay^bz^c)x^iy^jz^k =& z^{-k}x^{a-i}z^cx^iz^ky^b = x^{a-i}y^{k(a-i)}x^iy^{ki}z^{-k}y^{b-ci}z^{k+c}\\ =& x^ay^{ka-ic+b}z^c \end{aligned} \tag{3.13}$$

Thus, we have

$$gC_{g'} = \{(x^dy^ez^f)(x^ay^{ka-ic+b}z^c) \mid i,k \in \mathbb{Z}\} = \{(x^{d+a}y^{e-fa+ka-ic+b}z^{f+c}) \mid i,k \in \mathbb{Z}\}$$

Writing $g_i = x^{d_i}y^{e_i}z^{f_i}$, $g_i' = x^{a_i}y^{b_i}z^{c_i}$, $i = 1,\ 2$, It is clear that if an element common to the sets $g_1C_{g_1'}$ and $g_2C_{g_2'}$ exists, then we must have $a_1 + d_1 = a_2 + d_2 \pmod p$, $c_1 + f_1 = c_2 + f_2 \pmod p$ and such an element may be found by finding (i_1, j_1, i_2, j_2) satisfying

$$\frac{1}{p}[(c_1 + a_1) - (c_2 + a_2)] = (a_1j_1 - b_1i_1 - a_1d_1) - (a_2j_2 - b_2i_2 - a_2d_2) \pmod p$$

Thus, $M(p)$ and $N(p)$ are both efficiently C-decomposable.

For the statement on central products, let $G = H_1 \dots H_{r+s}$ be a central product, where for each i, $H_i \in \{M(p), N(p)\}$. First observe that the conjugacy class C_g^G of $g = h_1 \dots h_r$ in G can be written as the product $C_g^G = C_{h_1} \dots C_{h_{r+s}}$ of conjugacy classes C_{h_i} of h_i in H_i. Further, for any $g' = h_1' \dots h_r' \in G$, $g'C_g = h_1'C_{h_1} \dots h_r'C_{h_{r+s}}$. Without loss of generality $H_i = M(p),\ 1 \le i \le r$, $H_i = N(p),\ r+1 \le i \le s+r$ so $x_1^p = x_2^p = \dots = x_r^p = y_{r+1} = \dots = y_{r+s}$.

Write $g = h_1 \dots h_{s+r}$, $g' = h_1' \dots h_{s+r}'$, $h_i, h_i' \in H_i$. By the discussion above, there exist linear polynomials $A_i(s,t)$ and constants B_i $1 \le i \le r$ and linear polynomials $B_i(u)$ and constants D_i, C_i, $r+1 \le i \le r+s$ each determined by h_i, h_i', such that

$$\begin{aligned} g'C_g =& h_1'C_{h_1} \dots h_r'C_{h_r}\\ =& (x_1^{A_1(s_1,t_1)}y_1^{B_1}) \dots (x_r^{A_r(s_r,t_r)}y_r^{B_r}) \dots (x_{r+s}^{D_{r+s}}y_{r+s}^{B_{r+s}(u)}z_{r+s}^{C_{r+s}})\\ =& (x_1^{A_1(s_1,t_1)} \dots x_r^{A_r(s_r,t_r)} \dots x_{r+s}^{D_{r+s}})(y_1^{B_1} \dots y_r^{B_r} \dots y_{r+s}^{B_{r+s}(u)})(z_{r+1}^{C_{r+1}} \dots z_{r+s}^{C_{r+s}}) \end{aligned} \tag{3.14}$$

Note that this expression is not unique, as the intersections $H_i \cap H_j,\ i \neq j$ are not empty. One may further reduce the $A_i(s_i, t_i)$'s by substituting $x_1^p = x_2^p = \ldots = x_r^p = y_{r+1} = \ldots = y_{r+s}$, so without loss of generality we may assume that in the above expression, all exponents are less than p. Under this restriction, such an expression is unique. Now in order to find an element common to $g_1 C_{g_1'}$ and $g_2 C_{g_2'}$ one may equate the polynomials in the exponents of the x_i's and y_i's (note that all the exponents of the z_i's are constants so they must be equal if the intersection is nonempty) and solving for the integers s_i, t_i, u_i as done above, individually in each group. This requires the solution of at most $r + s$ linear equations $\pmod p$, and thus has complexity $\mathcal{O}((r+s)\log p) = \mathcal{O}(\log |G|)$. Thus, a polynomial time solution exists, and G is efficiently C-decomposable. □

It turns out that any extraspecial p-group can be constructed from $M(p)$ and $N(p)$ by taking central products, as formulated in the below well-known theorem which can be found, for instance, in [38].

Theorem 3.8. Every extraspecial p-group has order p^{1+2n} for some positive integer n, and conversely for each such number there are exactly two extraspecial groups up to isomorphism. Every extraspecial group of order p^{1+2n} can be written as a central product of either n copies of $M(p)$ or $n-1$ copies of $M(p)$ and 1 copy of $N(p)$.

A polynomial time algorithm for the computation of such a central product decomposition was given in [107]. As a consequence of this fact, along with Theorems 3.7 and 3.8, one concludes that all extraspecial p-groups are efficiently C-decomposable.

CSP in extraspecial p-groups It is known that the CSP in any finitely presented nilpotent group has a polynomial time solution (for instance, see [57]). Since the extraspecial p-groups are finitely presented nilpotent, the CSP in them can efficiently be solved in polynomial time. Nevertheless, we demonstrate below an alternate solution, as an application of the theory of efficient C-decomposability, by lifting the individual solutions of the CSP in $M(p)$ and $N(p)$ to all extraspecial p-groups.

3.4 An Overview of Braid-Based Cryptography

In this section, we provide a brief overview of some of the construction and cryptanalytic techniques that have been employed in protocols based on Braid groups. In particular, we will focus on the Algebraic Eraser and WalnutDSA (a signature scheme) We begin by giving a description of two of the common employed groups used as platforms for the protocols, the Braid groups and the Coloured Burau group.

Algorithm 3.1: Solving the CSP in an extraspecial p-group

Input Generators of an extraspecial p-group G and conjugate elements $\tilde{g}, g' \in G$

Output A conjugator $g \in G$ such that $g^{-1}\tilde{g}g = g'$

1: Use the algorithm of [107] to get a central product decomposition of G finitely many extraspecial groups of order p^3. Without loss of generality, $G = H_1 \dots H_{r+s}$, for $H_i = M(p),\ 1 \leq i \leq r$, $H_i = N(p),\ r+1 \leq i \leq s+r$.

2: Write $g = h_1 \dots h_{r+s}$, set $j \leftarrow 1$.

3: **while** $j \leq r+s$

(a) $\bar{K} \leftarrow H_{j+1} \dots H_{r+s}$, $\bar{H} \leftarrow H_j$. Write $g' = h'k',\ \tilde{g} = \tilde{h}\tilde{k}',\ h', h \in \bar{H},\ k', k \in \bar{K}$.

(b) Find an element $t_j \in h'^{-1}C_{\tilde{h}} \cap k'C_{\tilde{k}}$ by equating expressions of the form (3.14).

(c) Solve for h_j by solving the CSP $h_j^{-1}h'_j h_j = h'_j t_j$ in H_j, using Equations (3.12) and (3.13).

(d) $j \leftarrow j+1$.

3.4.1 Platform Groups and Algorithmic Problems

Braid groups

The Braid group on n stands is defined as the infinite group with the following presentation (called classical Artin's presentation)

$$B_n = \langle \sigma_1, \dots, \sigma_{n-1} \,|\sigma_i\sigma_j\sigma_i = \sigma_j\sigma_i\sigma_j \text{ if } |i-j| = 1, \\ \sigma_i\sigma_j = \sigma_j\sigma_i \text{ if } |i-j| > 1\rangle$$

A word over the alphabet $\{\sigma_1, \sigma_2, \dots, \sigma_{n-1}\}$, also called Artin's set of generators, is called a Braid word or Braid. The length of a shortest braid word representing an element $g \in B_n$ is called the geodesic length of g relative to Artin's set of generators and is denoted by $|g|$.

This group has an elegant geometric description. One may view a braid on n strings (i.e. an element of G) as an object consisting of $2n$ points and n strings such that

1. The beginning/ending points of the strings are (all) the top and bottom points,
2. The strings do not intersect,
3. No string intersects any horizontal line more than once.

Two braids on the same number of strings are said to be equivalent if the strings of one can be transformed into the strings of the other in the space strictly between

the peripheral points and without crossing ("pulling the strands"), keeping the ends fixed. Two braids can be "added" to yield a new braid by concatenation, i.e. joining the bottom points of the first braid to the top points of the second.

Note that B_n has a cyclic center generated by an element Δ^2, where $\Delta = (\sigma_1 \dots \sigma_{n-1})(\sigma_1 \dots \sigma_{n-2}) \dots (\sigma_1)$ is called the half-twist. The normal form typically used for the Braid groups is called the Garside normal form.

Coloured Burau group

Let $t = (t_1, \dots, t_n)$ be variables and S_n denote the group of permutations on n elements. Then, S_n acts naturally on $GL_n(\mathbb{F}_q(t))$ by permuting the t_i's. Denote this action by ${}^s m$ for $m \in GL_n(\mathbb{F}_q(t))$, $s \in S_n$. Consider the semidirect product $GL_n(\mathbb{F}_q(t)) \rtimes S_n = \{(m, s) \mid m \in GL_n(\mathbb{F}_q(t)),\ s \in S_n\}$ with multiplication given by

$$(m_1, s_1) \cdot (m_2, s_2) := (m_1 \, {}^{s_1}m_2,\ s_1 s_2)$$

Each generator σ_i of B_n is associated homomorphically to a specific matrix $CB(\sigma_i)$ in $GL_n(\mathbb{F}_q(t))$.

$$CB(\sigma_1) = \left(\begin{array}{cc|c} -t_1 & 1 & 0 \\ 0 & 1 & 0 \\ \hline 0 & 0 & I_{n-2} \end{array}\right), \quad CB(\sigma_1^{-1}) = \left(\begin{array}{cc|c} -\frac{1}{t_2} & \frac{1}{t_2} & 0 \\ 0 & 1 & 0 \\ \hline 0 & 0 & I_{n-2} \end{array}\right)$$

$$CB(\sigma_i) = \left(\begin{array}{c|ccc|c} I_{i-2} & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 & 0 \\ 0 & t_i & -t_i & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ \hline I_{i-2} & 0 & 0 & 0 & I_{n-i-1} \end{array}\right)$$

$$CB(\sigma_i^{-1}) = \left(\begin{array}{c|ccc|c} I_{i-2} & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 & 0 \\ 0 & -\frac{1}{t_{i+1}} & \frac{1}{t_{i+1}} & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ \hline I_{i-2} & 0 & 0 & 0 & I_{n-i-1} \end{array}\right)$$

Let $s_i = (i \ i+1) \in S_n$ be the transposition switching i and $i+1$ and $g_i = (CB(\sigma_i),\ s_i) \in GL_n(\mathbb{F}_q(t)) \rtimes S_n$. The subgroup $G := \langle g_1, \dots, g_{n-1} \rangle$ of $GL_n(\mathbb{F}_q(t)) \rtimes S_n$ is called the **Coloured Burau group**. The following algorithmic problem has been proposed to be a one-way function.

Definition 3.9 (E-multiplication). Let τ_i be nonzero elements of $\mathbb{F}_q$. Define a map ϕ from a subgroup of (infinite matrices) $GL_n(\mathbb{F}_q(t_1, \dots, t_n))$ to $GL_n(\mathbb{F}_q)$ by replacing each t_i with a nonzero element $\tau_i \in \mathbb{F}_q$. For

$(S,\pi),(M,\sigma) \in GL_n(\mathbb{F}_q(t_1,\ldots,t_n)) \rtimes S_n$, define **e-multiplication** by $(S,\pi) \star (M,\sigma) = (S\phi({}^{\pi}M), \pi\sigma)$.

Thus, reversing e-multiplication comprises the following problem. Given a pair $(M,\sigma) \in GL_N(\mathbb{F}_q) \times S_n$ such that $(M,\sigma) = \mathcal{P}(s) := (Id,1) \star s$ for some braid s, find a braid s' such that $\mathcal{P}(s') = (M,\sigma)$. Some protocols also employ in addition what is called the cloaking problem, which involves obfuscation of braids by multiplication with suitable group elements. Denote by the $C(b_1,\ldots,b_r)$ the commutator of the elements $b_1,\ldots,b_r$, i.e. $C(b_1,\ldots,b_r) := \langle b_i^{-1}b_j^{-1}b_ib_j \mid 1 \leq i,j \leq r\rangle$.

Definition 3.10 (Cloaking Problem). Given braids $b_1,\ldots,b_r$ commuting with unknown braid A, a permutation σ_B associated with an unknown braid $B \in \langle b_1,\ldots,b_r\rangle$, and a braid $P_A \in C_{\sigma_B}A$, find an element in $C_{\sigma_B}P_A \cap C(b_1,\ldots,b_r)$.

We now go on to describe some of the well-known public-key protocols based on the platforms and algorithmic problems described above.

3.4.2 Some Braid-Based Protocols

Algebraic Eraser

The Anshel-Anshel-Goldfeld-Lemieux key agreement protocol [8] uses the action of the semidirect product $GL(n,\mathbb{F}_p(t)) \rtimes S_n$ on the set $GL(n,\mathbb{F}_p) \times S_n$ to obscure algebraic structures. This protocol lies at the core of the Algebraic Eraser, which was proposed for key exchange use within lightweight cryptography, low-cost devices which constraint the use of computational resources, specifically in Internet of Things (IoT) devices, such as RFID tags.

It combines the commutator key exchange protocol with an additional method for obscuring elements. In [66], it was shown that the security of this system is broken if the following problem can be solved.

Definition 3.11 (Simultaneous Conjugacy Separation Search Problem). For tuples $\{v_1,\ldots,v'_\gamma\}$, $\{w_1,\ldots,w'_\gamma\}$ find any braid z' and any numbers $p_1,\ldots,p_\gamma,r_1,\ldots,r_\gamma \in \mathbb{Z}$ such that the words $\{\Delta^{2p_1}z'^{-1}v'_1z',\ldots,\Delta^{2p_\gamma}z'^{-1}v'_\gamma z'\}$ and $\{\Delta^{2r_1}z'^{-1}w'_1z',\ldots,\Delta^{2r_\gamma}z'^{-1}w'_\gamma z'\}$ can be expressed as words over 2 disjoint commuting subsets of generators of B_n

WalnutDSA

WalnutDSA [9] was one of 20 entries for public key signature schemes at NIST call for post-quantum cryptography. It is based on the colored Burau group platform and e-multiplication as a one-way function. Under this scheme, the signer's private key consists of two braids and the public keys are matrices. Signatures are generated by

performing E-multiplication, but the verification step requires only operations with the matrix components. The security model is chosen to be Existential unforgeability under chosen message attacks (EU-CMA) under which the adversary can ask for polynomially many signatures of messages to a signing oracle.

The Kayawood protocol uses similar techniques to WalnutDSA and is used for key exchange. Similarly, the Ironwood protocol [6] designed using the same principles for message encryption.

3.4.3 Methods for Attacks

Length-based attacks

Under a length-based attack, the adversary uses a **length function** $|\cdot|$ in B_n to approximate the geodesic length, and formulates the underlying group theoretic problem as a minimization problem. This can then be solved heuristically with a greedy descent algorithm. The underlying idea behind the greedy algorithm is that for random x, $|x^{-1}ux| > |u|$ is true for almost all tuples u, and for a random x_1, x_2 with $|x_1| > |x_2|$, $|x_1^{-1}ux_1| > |x_2^{-1}ux_2|$ is true for almost all tuples u. To find the unknown conjugator, the algorithm repeatedly conjugates the tuple of elements by generators of B_n and their inverses. If a decrease in total length of the tuple is observed for a generator it is assumed that this generator is involved in the required conjugator. The attack of [66] on WalnutDSA is an important example.

Linear algebra attacks

There exists a broad range of attacks that apply when the underlying platform group G has an efficient, faithful matrix representation $\rho : G \to GL_n(\mathbb{F}_q)$ with an efficiently computable inverse. In this case, the algorithmic problem of G can be reduced to the corresponding problem in a matrix algebra over a finite field. In order to solve this problem, the attacker must develop a method for efficiently computing the bases of cleverly chosen subspaces or subalgebras of the matrix algebra. Once such a basis is available, the underlying problem is solved in polynomial time. There are several variations of such an attack, for example [12, 65, 103].

In [12] and [11] the attacks on the Algebraic Eraser proceed by generating lots of elements from the involved subgroup of $GL_n(\mathbb{F}_q)$ and using them to find linear relationships between the secret braids. Then, an algorithm from permutation group theory is used to compute braids with the specific involved permutations. From this, the shared key is derivd. In [11] an attack is devised that subverts the problem of sovling the simulataneous CSP by building surrogate private keys.

Other attacks

In [43], a universal forgery attack on WalnutDSA was provided, which broke the system in a matter of two minutes assuming access to a small set of random message-signature pairs. It reduces the problem of forging signatures to a factorization problem in the group $\mathrm{GL}_n(\mathbb{F}_q)$. There is also a generic birthday/collision-based attack against WalnutDSA in [13], under which, if the hashes of two messages are "equivalent" in the coloured Burau group, then a valid signature for one is a valid signature for the other. The security of the system is broken by finding two such messages. In [53], an "uncloaking" attack is presented against the Kayawoof protocol, which shows that the method of generating cloaking elements produces a distribution that enables an attack using only the public keys.

Chapter 4

Cryptanalysis of a System based on Twisted Dihedral Group Algebras

4.1 Introduction

The design of efficient public key cryptographic systems that resist quantum attacks presently constitutes the important area of research called post-quantum cryptography. Non-commutative structures such as non-abelian groups, group rings, semigroups, etc., along with pertinent algorithmic problems, have been used for the construction of cryptosystems in a plethora of works in this field. Two algorithmic problems that find great mention in this realm are the so-called conjugacy search problem and the decomposition problem (see [16, 41, 92]).

In Chapter 3, we discussed the various aspects of non-abelian group-based cryptography and the solution of the conjugacy search problem in various groups. In this chapter, we shift our focus to the decomposition problem, and look at a different kind of platform, viz. twisted group algebras, which are similar to group algebras but have a more complicated multiplicative structure. In [24], the authors construct a key exchange system based on so-called twisted group algebras over a finite field $\mathbb{F}_q$ and a double-sided multiplication problem. In this chapter, we give a full cryptanalysis of this system.

The underlying platform of the system in [24] is a twisted group algebra of the dihedral group D_{2n} over a finite field $\mathbb{F}_q$ with twisted multiplication defined with the help of a function, called a 2-cocycle. The 2-cocycle α is chosen by the authors such that $\mathbb{F}_q^{\alpha} D_{2n}$ and $\mathbb{F}_q D_{2n}$ (these notations are introduced in the next section) are not isomorphic, so that one is no longer working over a group algebra. Some recent relevant works on twisted group algebras are [37] and [26]. In [26], the authors study right ideals of twisted group algebras, endowing them with a natural distance and thus studying them as codes; they show that all perfect linear codes are twisted group codes. In [37], the authors use twisted dihedral group rings as a platform for a public key protocol as a non-commutative variation of the Diffie-Hellman protocol. This protocol has a similar platform to the one in [24], but with the twisted multiplication and 2-cocycle defined differently. The authors show in [24] that the twisted group

algebra platforms are structurally different.

Group algebras have found mention in some other proposed public key cryptographic schemes. In [49], the authors construct a key exchange protocol based on the discrete logarithm problem in the semigroup $\mathrm{Mat}_3(\mathbb{F}_7[S_5])$ of 3×3 matrices over the group ring $\mathbb{F}_7[S_5]$, where S_5 is the group of permutations on five symbols. In [68], an attack was devised by showing that $\mathrm{Mat}_3(\mathbb{F}_7[S_5])$ embeds into $\mathrm{Mat}_{360}(\mathbb{F}_7)$, for which the discrete logarithm problem can then be solved using the method in [60] adapted to singular matrices. Another attack [28] on the same system uses the fact that the algebra $\mathbb{F}_7[S_5]$ is semisimple, and so by Maschke's theorem it is isomorphic to a direct sum of matrix algebras over $\mathbb{F}_7$. Some other protocols using related methods have been proposed in [56] and [58].

The authors of [24] assert that since Maschke's Theorem is valid also for twisted group algebras, a similar attack might break the underlying problem of their system. To resolve this they choose q such that the twisted group algebra is not semisimple. Further, they assert that the general methods of cryptanalysis in [85] and [86], which require the construction of bases over some vector spaces, do not apply to their system. This is attributed to the facts that the twisted group algebra is not a group under the twisted multiplication and that there is an added dimension of non-commutativity with the twisted multiplication.

The security of the protocol in [24] relies on a newly introduced algorithmic assumption, which the authors call Dihedral Product Decomposition (DPD) Assumption. Under this assumption, the authors prove that their protocol is session-key secure in the authenticated-links adversarial model of Canetti and Krawczyk [14]. The underlying algorithmic problem can be seen as a special form of the decomposition problem over the multiplicative monoid of an algebra A: given $(x, y) \in A$ and $S \subseteq G$, the problem is to find $z_1, z_2 \in S$ such that $y = z_1 x z_2$.

The Dihedral Product Decomposition Problem constitutes finding (z_1, z_2) given $z_1 x z_2$ and x in the platform, where z_1 and z_2 lie in specific predefined subalgebras of $\mathbb{F}_q^{\alpha} D_{2n}$. It is therefore a more restricted version of the general decomposition problem in the platform. The authors claim that the protocol proposed is quantum-safe, with justification based on the fact that the decomposition problem is a generalization of the conjugacy search problem, which is believed to be difficult even for quantum computers, in certain platform groups.

Summary of Contributions

In this chapter, we provide a full cryptanalysis of the key exchange system in [24] by providing a classical polynomial time algebraic solution to the underlying DPD

Problem. We provide a heuristic argument to show that our attack algorithm has a high success rate, and verify this experimentally for the parameters proposed by the authors. We also show that the underlying DPD problem may be formulated as a semigroup action problem [58], with multiplication in the multiplicative monoid of a twisted dihedral group algebra.

This chapter is structured as follows. In Section 4.2 we describe the structure and some properties of the underlying platform, viz. the twisted group algebra $\mathbb{F}_q^{\alpha} D_{2n}$, closely following the results of [24]. In Section 4.3, we describe the key exchange protocol proposed in [24] and state the DPD problem, which forms the basis of its security assumption. We show that despite the use of a non-commutative structure, this algorithmic problem is equivalent to a commutative semigroup action problem. In Section 4.4, we present some background definitions and results on circulant matrices, which are needed for our reduction and cryptanalysis. In Section 4.5, we describe an algebraic reduction of the DPD problem to a set of simultaneous equations over $\mathbb{F}_q$ and show that in a majority of cases, they can be solved by linear algebra in polynomial time. Using these results, we provide a polynomial time algorithm which performs the cryptanalysis of the system of [24].

The material in this chapter has been adapted from the preprint [99] authored by me.

Notation Throughout, we let $\mathbb{F}$ denote a field, G denote a finite group and $\mathbb{F}_q$ denote the finite field with q elements, where q is a power of a prime. Also let $\mathbb{F}_q^* = \mathbb{F}_q \setminus \{0\}$. We denote by D_{2n} the dihedral group of size $2n$.

4.2 Structure of the Platform

Definition 4.1 (Group Algebra). Let G be a group. The group algebra $\mathbb{F}[G]$ is the set of the formal sums $\sum_{g \in G} a_g g$, with $a_g \in \mathbb{F}$, $g \in G$. Addition is defined componentwise: $\sum_{g \in G} a_g g + \sum_{g \in G} b_g g := \sum_{g \in G} (a_g + b_g) g$. Multiplication is defined as $\sum_{g \in G} a_g g \cdot \sum_{g \in G} b_g g := \sum_{g \in G} \sum_{h \in G} (a_g b_h) gh = \sum_{k \in G} \sum_{g \in G,\ h \in G : gh = k} a_g b_h k$.

Clearly, $\mathbb{F}[G]$ is an algebra over $\mathbb{F}$ with dimension $|G|$. If G is non-commutative, so is $\mathbb{F}[G]$. In [37], the authors apply the structure of twisted group algebra to construct a public-key exchange system. The multiplication operation in these twisted group algebras is defined using the concept of 2-cocycles.

Definition 4.2 (2-Cocycle). A map $\alpha : G \times G \to \mathbb{F}^*$ is called a 2-cocycle of G if $\alpha(1,1) = 1$ and for all $g, h, k \in G$ we have $\alpha(g, hk)\alpha(h, k) = \alpha(gh, k)\alpha(g, h)$.

Definition 4.3 (Twisted Group Algebra). Let α be a 2-cocycle of G. The twisted group algebra $\mathbb{F}^\alpha G$ is the set of all formal sums $\sum_{g\in G} a_g g$, where $a_g \in \mathbb{F}$, with the following twisted multiplication: $g \cdot h = \alpha(g,h)gh$, for $g, h \in G$. The multiplication rule extends linearly to all elements of the algebra:

$$(\sum_{g\in G} a_g g) \cdot (\sum_{h\in G} b_h h) = \sum_{g\in G}\sum_{h\in G} a_g b_h \alpha(g,h)gh.$$

Addition is given componentwise as in Definition 4.1.

We note that the associativity of a twisted group algebra follows from the condition on α being a 2-cocycle. In fact, it is an if and only if condition.

Remark 4.1. Throughout the rest of the paper, we will be concerned with twisted group algebras, and so it is understood that the product $(\sum_{g\in G} a_g g)\cdot(\sum_{h\in G} a_h h)$ denotes twisted multiplication. Further, we will usually omit the $\cdot$ symbol, so that multiplication in the group G and in the twisted group algebra are not differentiated by operation notation. To avoid confusion we ensure that the symbols used for elements of the group and group algebra do not intersect.

Denote the set of all 2-cocycles of G into $\mathbb{F}_q$ by $Z^2(G,\mathbb{F}_q^*)$. For $\alpha, \beta \in Z^2(G,\mathbb{F}_q^*)$, one may define the cocycle $\alpha\beta \in Z^2(G,\mathbb{F}_q^*)$ by $\alpha\beta(g,h) = \alpha(g,h)\beta(g,h)$ for all $g, h \in G$. With this operation, $Z^2(G,\mathbb{F}_q^*)$ becomes a multiplicative abelian group.

Definition 4.4 (Adjunct). For an element $a = \sum_{g\in G} a_g g \in \mathbb{F}^\alpha G$ we define its adjunct as $\hat{a} := \sum_{g\in G} a_g \alpha(g, g^{-1})g^{-1}$.

4.2.1 A twisted dihedral group algebra

For the rest of this paper, we set $\mathbb{F} = \mathbb{F}_q$ and $G = D_{2n}$, where

$$D_{2n} = \langle x, y : x^n = y^2 = 1, yxy^{-1} = x^{-1}\rangle$$

is the dihedral group of order $2n$. Further, we let $C_n = \langle x^i\rangle$ be the cyclic subgroup of D_{2n} generated by x and α be a 2-cocycle of D_{2n}.

The following lemma from [24] can be verified in a straightforward manner.

Lemma 4.1 ([**24**])**.** We have

1. $\mathbb{F}_q^\alpha D_{2n}$ is a free $\mathbb{F}_q^\alpha C_n$-module with basis $\{1, y\}$. Therefore $\mathbb{F}_q^\alpha D_{2n} = \mathbb{F}_q^\alpha C_n \oplus \mathbb{F}_q^\alpha C_n y$ as a direct sum of $\mathbb{F}_q$-vector spaces.

2. $\mathbb{F}_q^\alpha C_n y \cong \mathbb{F}_q^\alpha C_n$ as $\mathbb{F}_q^\alpha C_n$-modules.
3. For $a \in \mathbb{F}_q^\alpha C_n y$, $ab \in \mathbb{F}_q^\alpha C_n$ if $b \in \mathbb{F}_q^\alpha C_n y$ and $ab \in \mathbb{F}_q^\alpha C_n y$ if $b \in \mathbb{F}_q^\alpha C_n$.
4. If $a \in \mathbb{F}_q^\alpha C_n$, then $\hat{a} \in \mathbb{F}_q^\alpha C_n$. Similarly, if $a \in \mathbb{F}_q^\alpha C_n y$, then $\hat{a} \in \mathbb{F}_q^\alpha C_n y$.

Definition 4.5. 1. For a 2-cocycle α of D_{2n} we define the reversible subspace of $\mathbb{F}_q^\alpha C_n y$ as the vector subspace

$$\Gamma_\alpha = \{a = \sum_{i=0}^{n-1} a_i x^i y \in \mathbb{F}_q^\alpha C_n y \mid a_i = a_{n-i} \text{ for } i = 1, \ldots, n-1\}.$$

2. Define a map $\psi : \mathbb{F}_q^\alpha C_n y \to \mathbb{F}_q^\alpha C_n$ as follows. Given $a = \sum_{i=0}^{n-1} a_i x^i y \in \mathbb{F}_q^\alpha C_n y$ we define $\psi(a) = \sum_{i=0}^{n-1} a_i x^i \in \mathbb{F}_q^\alpha C_n$. Clearly, ψ is an $\mathbb{F}_q$-linear isomorphism.

In this paper, we will refer to an element $\sum_{i=1}^{n-1} a_i x^i y$ of the reversible subspace Γ_α as a reversible element of $\mathbb{F}_q^\alpha C_n y$ and to the corresponding vector $(a_0, \ldots, a_{n-1}) \in \mathbb{F}_{q^n}$ as a reversible vector.

Lemma 4.2 ([24]). Let α be a 2-cocycle of D_{2n}. Then we have

1. If
$$\alpha(x^i, x^{j-i}) = \alpha(x^{j-i}, x^i) \tag{4.1}$$
for all $i, j \in \{0, \ldots, n-1\}$, then $ab = ba$ for $a, b \in \mathbb{F}_q^\alpha C_n$.
2. If
$$\alpha(x^{i-j}y, x^{i-j}y)\alpha(x^i y, x^{i-j}y) = \alpha(x^{n-i}y, x^{n-i}y)\alpha(x^{j-i}y, x^{n-i}y) \tag{4.2}$$
for all $i, j \in \{0, \ldots, n-1\}$, then $a\hat{b} = b\hat{a}$ for $a, b \in \Gamma_\alpha$.

The following lemma provides an explicit construction of the 2-cocycle that will be used throughout in the cryptographic construction of [24].

Lemma 4.3 ([24]). Let $\lambda \in \mathbb{F}_q^* = \mathbb{F}_q \setminus \{0\}$. The map $\alpha_\lambda : D_{2n} \times D_{2n} \to \mathbb{F}_q^*$ defined by

$$\begin{aligned} &\alpha_\lambda(g, h) = \lambda \text{ for } g = x^i y,\ h = x^j y \text{ with } i, j \in \{0, \ldots, n-1\} \text{ and} \\ &\alpha_\lambda(g, h) = 1 \text{ otherwise} \end{aligned} \tag{4.3}$$

is a 2-cocycle. Further, α_λ satisfies the two conditions (4.1) and (4.2).

Proof. By definition, $\alpha_\lambda(1,1) = 1$. Thus one only needs to verify that $\alpha_\lambda(g,h)\alpha_\lambda(gh,k) = \alpha_\lambda(g,hk)\alpha_\lambda(h,k)$ for all $g,h,k \in D_{2n}$. Write $h = x^{j_1}y^{k_1}$ and $k = x^{j_2}y^{k_2}$ with $i, j_1, j_2 \in \{0,\ldots,n-1\}$. The condition may then be directly verified separately in a straightforward way for the two possible cases $g = x^i$ and $g = x^iy$. The fact that α_λ satisfies conditions (4.1) and (4.2) follows from the definition. □

Lemma 4.4 ([24]). $\mathbb{F}_qD_{2n}$ and ${\mathbb{F}_q}^{\alpha_\lambda}D_{2n}$ are isomorphic if and only if λ is a square in $\mathbb{F}_q^*$, i.e. if and only if $\lambda^{(q-1)/2} = 1$.

Lemma 4.5 ([24]). If λ_1, λ_2 are not squares in $\mathbb{F}_q^*$, then ${\mathbb{F}_q}^{\alpha_{\lambda_1}}D_{2n}$ and ${\mathbb{F}_q}^{\alpha_{\lambda_2}}D_{2n}$ are isomorphic.

From Lemma 4.2 we thus have that for the choice $\alpha = \alpha_\lambda$ of 2-cocycle, the multiplicative ring of $\mathbb{F}_q^\alpha C_n$ is commutative, and that $a\hat{b} = b\hat{a}$ for all $a,b \in \Gamma_\alpha$. The form (4.3) of $\alpha = \alpha_\lambda$ is adopted throughout for the cryptosystem in [24] and thus we restrict our study to this cocycle. Thus, henceforth we take $\alpha = \alpha_\lambda$.

4.3 The key exchange protocol

Having described the relevant structural properties of the underlying platform, we now describe the key exchange protocol in [24]. This uses two-sided multiplications in $\mathbb{F}_q^\alpha D_{2n}$.

4.3.1 Public parameters

1. A number $m \in \mathbb{N}$ and a prime $p > 2$ with $p \mid 2n$ and set $q = p^m$.
2. A 2-cocycle $\alpha = \alpha_\lambda$ for a non-square λ in $\mathbb{F}_q^*$. This ensures that the platform $\mathbb{F}_q^\alpha D_{2n}$ is not isomorphic to $\mathbb{F}_qD_{2n}$.
3. An element $h = h_1 + h_2$ for a random $0 \neq h_1 \in \mathbb{F}_q^\alpha C_n$ and a random $0 \neq h_2 \in \mathbb{F}_q^\alpha C_n y$. (Clearly, since h is public, so are h_1 and h_2.)

Protocol 4.1 describes the key exchange protocol of [24].

Protocol 4.1.

1. *Alice chooses a secret pair* $(s_1,t_1) \in \mathbb{F}_q^\alpha C_n \times \Gamma_\alpha$, *and sends* $\mathrm{pk}_A = s_1ht_1$ *to Bob.*
2. *Bob chooses a secret pair* $(s_2,t_2) \in \mathbb{F}_q^\alpha C_n \times \Gamma_\alpha$ *and sends* $\mathrm{pk}_B = s_2ht_2$ *to Alice.*
3. *Alice computes* $K_A = s_1\,\mathrm{pk}_B\,\hat{t_1}$,
4. *Bob computes* $K_B = s_2\,\mathrm{pk}_A\,\hat{t_2}$

5. *The shared key is* $K = K_A = K_B$

The authors' proposed values for parameters q and n are $q = n = 19$, $q = n = 23$, $q = n = 31$, $q = n = 41$.

4.3.2 Correctness

It is easy to show that within an uncorrupted session, both Alice and Bob establish the same key. Indeed, because of the choice of $\alpha = \alpha_\lambda$, we have $s_i s_j = s_j s_i$ in $\mathbb{F}_q^\alpha C_n$ and $t_i \hat{t_j} = t_j \hat{t_i}$ in $\mathbb{F}_q^\alpha C_n y$ for $i, j \in \{1, 2\}$, so

$$K_A = s_1 \, \mathrm{pk}_B \, \hat{t_1} = s_1 s_2 h t_2 \hat{t_1} = s_2 s_1 h t_1 \hat{t_2} = s_2 \, \mathrm{pk}_A \, \hat{t_2} = K_B.$$

4.3.3 Security Assumption

The security of the protocol depends on the assumption of the difficulty of the following algorithmic problem.

Definition 4.6 (Dihedral Product Decomposition (DPD) Problem). Let $(s, t) \in \mathbb{F}_q^\alpha C_n \times \Gamma_{\alpha_\lambda}$ be a secret key. Given a public element $h = h_1 + h_2 \in \mathbb{F}_q^\alpha D_{2n}$, $h_1 \in \mathbb{F}_q^\alpha C_n$, $h_2 \in \mathbb{F}_q^\alpha C_n y$, and a public key $\mathrm{pk} = sht$, the DPD problem requires an adversary to compute $(\tilde{s}, \tilde{t}) \in \mathbb{F}_q^\alpha C_n \times \Gamma_\alpha$ such that $\mathrm{pk} = \tilde{s} h \tilde{t}$.

Let $(\tilde{s}, \tilde{t})$ be the output of an adversary $\mathcal{A}$ attempting to solve the DPD problem for $\mathbb{F}_q^\alpha D_{2n}$. The authors define $\mathcal{A}$'s advantage $DPD_{adv}[\mathcal{A}, \mathbb{F}_q^\alpha D_{2n}]$ in solving the DPD problem as the probability that $\tilde{s} h \tilde{t} = sht$.

Definition 4.7 (DPD Assumption). The DPD assumption is said to hold for $\mathbb{F}_q^\alpha D_{2n}$ if for all efficient adversaries $\mathcal{A}$ the quantity $DPD_{adv}[\mathcal{A}, \mathbb{F}_q^\alpha D_{2n}]$ is negligible.

In Section 4.5, we provide a cryptanalysis of Protocol 4.1 by solving the DPD problem. We show that in most cases, a polynomial time solution is possible, and so the DPD assumption does not hold. For our method of cryptanalysis, we need some prerequisites on circulant matrices, which we provide in the next section. However, we first show below how the DPD problem can be formulated as a special case of a commutative semigroup action problem, in the framework introduced in [58].

DPD problem as a commutative semigroup action

The authors of [24] assert that given a fixed $h \in \mathbb{F}_q^\alpha D_{2n}$, the set of keys $\{sht \mid (s, t) \in \mathbb{F}_q^\alpha C_n \times \Gamma_\alpha\}$ is not even a semigroup under the twisted algebra multiplication. From this observation, they claim that their system is immune to the quantum cycle-finding algorithm of Shor [90] which is known to solve the hidden subgroup problem in abelian groups.

Further, the security of the system of [24] is based on the presence of a non-commutative multiplication in the twisted group algebra. However, we now show that the DPD problem can be formulated as a commutative semigroup action problem, and so any classical or quantum solution to the latter also applies to the former. In [64], a Pollard-rho type square root algorithm was provided to solve an abelian group action problem, whereas the possibility for a modification to the commutative semigroup case was left open.

As observed before, the cocycle $\alpha = \alpha_\lambda$ satisfies conditions (4.1) and (4.2). Thus, $ab = ba$ for $a, b \in \mathbb{F}_q^\alpha C_n$ and $a\hat{b} = b\hat{a}$ for $a, b \in \Gamma_\alpha$. In particular, $\mathbb{F}_q^\alpha C_n$ is a commutative subalgebra of $\mathbb{F}_q^\alpha D_{2n}$. Recall the $\mathbb{F}_q$-linear isomorphism $\psi : \mathbb{F}_q^\alpha C_n y \to \mathbb{F}_q^\alpha C_n$ given by $\psi(a) = \sum_{i=0}^{n-1} a_i x^i \in \mathbb{F}_q^\alpha C_n$ for $a = \sum_{i=0}^{n-1} a_i x^i y \in \mathbb{F}_q^\alpha C_n y$. Below, we show that $\psi(\Gamma_\alpha)$ is a commutative semigroup under the multiplication defined by $\psi(t) \star \psi(t') := tt' \in \psi(\Gamma_\alpha)$.

Lemma 4.6. Let α be a cocycle on $\mathbb{F}_q^\alpha D_{2n}$ such that for all i and j, $\alpha(x^i y, x^j y) = \alpha(x^j y, x^i y)$ and $\alpha(x^k y, x^{m-k} y) = \alpha(x^{n-k} y, x^{k-m} y)$ where all exponents are taken modulo n. Then, $\psi(\Gamma_\alpha)$ is a commutative semigroup under $\star$.

Proof. The associativity of the operation follows from the definition of a cocycle. Let $a = \sum_{i=0}^{n-1} a_i x^i y, b = \sum_{i=0}^{n-1} b_i x^i y \in \Gamma_\alpha$, We have

$$\begin{aligned}\psi(a) \star \psi(b) = ab &= \left(\sum_{i=0}^{n-1} a_i x^i y\right)\left(\sum_{j=0}^{n-1} b_j x^j y\right) \\ &= \sum_{m=0}^{n-1}\left(\sum_{k=0}^{n-1} a_k b_{m-k}\alpha(x^k y, x^{m-k} y)\right) x^m\end{aligned}$$

Since $a, b \in \Gamma_\alpha$, $a_{n-k} = a_k$ and $b_{k-m} = b_{m-k}$ for each m, k, so, the applying the assumption on α, $\sum_{k=0}^{n-1} a_k b_{m-k}\alpha(x^k y, x^{m-k} y) = \sum_{k=0}^{n-1} a_{n-k} b_{k-m}\alpha(x^{n-k} y, x^{k-m} y)$, i.e. the coefficients of x^m and x^{n-m} are equal, so $\psi(a) \star \phi(b) = ab \in \psi(\Gamma_\alpha)$. Further, since for all i, j, $\alpha(x^i y, x^j y) = \alpha(x^j y, x^i y)$, we have $\psi(a) \star \psi(b) = ab = ba = \psi(b) \star \phi(a)$, so the semigroup is commutative. □

Clearly, the choice of $\alpha = \alpha_\lambda$ in the protocol satisfies the properties in Lemma 4.6. We can now look at the key exchange in Protocol 4.1 as an instance of a semigroup action problem, introduced in [58].

Definition 4.8 (Semigroup Action Problem). Let S be any semigroup acting on a set X

$$S \times X \to X$$
$$(s, x) \mapsto s \cdot x$$

Given an element $y = s \cdot x \in X$, where $x \in X$ is known and $s \in S$ is a secret, the semigroup action problem is to find some $\tilde{s} \in S$ such that $\tilde{s} \cdot x = y$.

Proposition 4.1. The commutative semigroup $\mathbb{F}_q^\alpha C_n \times \psi(\Gamma_\alpha)$ acts on $\mathbb{F}_q^\alpha D_{2n}$ as follows

$$\begin{aligned}(\mathbb{F}_q^\alpha C_n \times \psi(\Gamma_\alpha)) \times \mathbb{F}_q^\alpha D_{2n} &\to \mathbb{F}_q^\alpha D_{2n} \\ (s, \psi(t)) \cdot h &= sht \end{aligned} \tag{4.4}$$

Proof. Clearly, $(1, 1) \cdot h = h$ for all $h \in \mathbb{F}_q^\alpha D_{2n}$. Further,

$$(s, \psi(t))((s', \psi(t')) \cdot h) = ss'ht't = ss'ht = (ss', tt') \cdot h = (ss', \psi(t) \star \psi(t')) \cdot h.$$

□

Lemma 4.7. The DPD problem is equivalent to the semigroup action problem for the commutative semigroup action (4.4).

Proof. Clearly, t and $\psi(t)$ can easily be read from each other without any significant computational cost. Suppose that given public element h and public key pk, the adversary can find s, t such that sht = pk. Then, $(s, \psi(t))$ is a solution to the SAP (4.4). Conversely, any solution $(s, \psi(t))$ of the SAP (4.4) gives the solution (s, t) of the DPD problem. □

The next section highlights some prerequisites on circulant matrices which will be used in the cryptanalysis of the system in Section 4.5.

4.4 Circulant Matrices

Definition 4.9. A matrix over $\mathbb{F}_q$ of the form $\begin{pmatrix} c_0 & c_{n-1} & \dots & c_1 \\ c_1 & c_0 & \dots & c_2 \\ \vdots & \vdots & \ddots & \vdots \\ c_{n-1} & c_{n-2} & \dots & c_0 \end{pmatrix}$ with $c_i \in \mathbb{F}_q$, is called circulant. Given a vector $\mathbf{c} = (c_0, c_1, \dots, c_{n-1})^T \in \mathbb{F}_{q^n}$, we use the notation

$M_{\mathbf{c}}$ to denote the circulant matrix $M_{\mathbf{c}} := \begin{pmatrix} c_0 & c_{n-1} & \dots & c_1 \\ c_1 & c_0 & \dots & c_2 \\ \vdots & \vdots & \ddots & \vdots \\ c_{n-1} & c_{n-2} & \dots & c_0 \end{pmatrix}$.

Definition 4.10. Given vectors $\mathbf{b} = (b_0, b_1, \dots, b_{n-1})^T \in \mathbb{F}_{q^n}$, $\mathbf{c} = (c_0, c_1, \dots, c_{n-1})^T \in \mathbb{F}_{q^n}$, define, for $0 \leq \ell \leq n-1$ the constants

$$z_\ell(\mathbf{b}, \mathbf{c}) = \sum_{i+j=\ell \pmod n} b_i c_j = \begin{pmatrix} c_\ell, & c_{\ell-1}, & \dots, & c_{\ell+1} \end{pmatrix} \cdot \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{n-1} \end{pmatrix}, 0 \leq \ell \leq n-1.$$

Also define the vector $\mathbf{z}_{\mathbf{b},\mathbf{c}} = (z_0(\mathbf{b}, \mathbf{c}), \dots, z_\ell(\mathbf{b}, \mathbf{c}), \dots, z_{n-1}(\mathbf{b}, \mathbf{c}))^T$. In other words,

$$\mathbf{z}_{\mathbf{b},\mathbf{c}} = \begin{pmatrix} c_0 & \dots & c_1 \\ c_1 & \dots & c_2 \\ \vdots & \ddots & \vdots \\ c_{n-1} & \dots & c_0 \end{pmatrix} \cdot \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{n-1} \end{pmatrix} = M_{\mathbf{c}} \cdot \mathbf{b}.$$

As in Definition 4.9, denote by $M_{\mathbf{z}}(\mathbf{b}, \mathbf{c})$ the circulant matrix $M_{\mathbf{z}}(\mathbf{b}, \mathbf{c}) = \begin{pmatrix} z_0(\mathbf{b}, \mathbf{c}) & \dots & z_1(\mathbf{b}, \mathbf{c}) \\ z_1(\mathbf{b}, \mathbf{c}) & \dots & z_2(\mathbf{b}, \mathbf{c}) \\ \vdots & \ddots & \vdots \\ z_{n-1}(\mathbf{b}, \mathbf{c}) & \dots & z_0(\mathbf{b}, \mathbf{c}) \end{pmatrix}$. The following result is easy to verify by direct computation.

Lemma 4.8. $M_{\mathbf{z}}(\mathbf{b}, \mathbf{c}) = M_{\mathbf{c}} \cdot M_{\mathbf{b}}$.

4.4.1 Probability of a circulant matrix being invertible

We will require the invertibility of some random circulant matrices over $\mathbb{F}_q$ for our reduction of the system. For this reason, we discuss the criteria for a random circulant matrix being invertible, and study this probability. We have the following result from [88].

Proposition 4.2 ([88])**.** Let $x^n - 1 = f_1^{\alpha_1}(x) \dots f_\tau^{\alpha_\tau}(x)$ be the factorization of $x^n - 1$ over $\mathbb{F}_{q^m}$ into powers of irreducible factors. The number of invertible circulant matrices in $Mat_n(\mathbb{F}_{q^m})$ is equal to $\prod_{i=1}^{\tau} (q^{md_i\alpha_i} - q^{md_i(\alpha_i - 1)})$, where d_i is the degree of $f_i(x)$ in the factorization of $x^n - 1$.

Note that the number of circulant matrices over $\mathbb{F}_{q^m}$ is q^{mn}. As a direct consequence, the probability of a randomly chosen circulant matrix over $\mathbb{F}_{q^m}$ being invertible is

$$\prod_{i=1}^{\tau} \frac{q^{md_i\alpha_i} - q^{md_i(\alpha_i-1)}}{q^{nm}} = \prod_{i=1}^{\tau} \left(1 - \frac{1}{q^{md_i}}\right)$$

It is now easy to see that a lower bound for this quantity is $(1 - \frac{1}{q^m})^n$, which is achieved if $x^n - 1$ splits into distinct linear factors, i.e. $\tau = n$, $d_i = 1$, $\alpha_i = 1$. Similarly, an upper bound is achieved when there is a single factor in the factorization, i.e. $\tau = 1$ and $\alpha_1 = n$, in which case the quantity is $(1 - \frac{1}{q^m})$. Note that this upper bound is achieved when n is a power of the characteristic p of $\mathbb{F}_{q^m}$. ($x^n - 1 = (x-1)^n \pmod p$). In general, we have the following corollary.

Corollary 4.1. Let e be the largest number such that $p^e \mid n$. Then the probability that a randomly chosen $n \times n$ circulant matrix over $\mathbb{F}_q$ is invertible is at least $(1 - \frac{1}{q})^{\frac{n}{p^e}}$. If n is a power of p, then the probability is exactly equal to $1 - \frac{1}{q}$.

In [24], the authors deliberately choose the case $p \mid n$, so as to avoid having $\mathbb{F}_{q^m} D_{2n}$ semisimple. In fact, in all of their proposed parameters, one has $n = p = q$, and so, the probability $1 - \frac{1}{q}$ applies for a random circulant matrix being invertible.

4.5 Cryptanalysis

Note that the adversary is given an equation of the form $sht = \gamma$ over $\mathbb{F}_q^{\alpha} D_{2n}$, where

$$s = \sum_{i=0}^{n-1} a_i x^i \in \mathbb{F}_q^{\alpha} C_n, \; t = \sum_{i=0}^{n-1} b_i x^i y \in \Gamma_{\alpha} \subseteq {\mathbb{F}_q}^{\alpha_\lambda} D_{2n} \tag{4.5}$$

are unknown, and $h = \sum_{i=0}^{n-1} c_i x^i + \sum_{i=0}^{n-1} d_i x^i y$ is known. Since $t \in \Gamma_{\alpha}$, the coefficients in t satisfy $b_k = b_{n-k}$ for $k = 1, \ldots, n-1$. We write

$$\gamma = \sum_{i=0}^{n-1} v_i x^i + \sum_{i=0}^{n-1} w_i x^i y$$

for known constants v_i, w_i. Substituting the above expansions into the equation $sht = \gamma$, we have

$$\begin{aligned}
&\left(\sum_{i=0}^{n-1} a_i x^i\right)\left(\sum_{i=0}^{n-1} c_i x^i + \sum_{i=0}^{n-1} d_i x^i y\right)\left(\sum_{i=0}^{n-1} b_i x^i y\right) = \sum_{i=0}^{n-1} v_i x^i + \sum_{i=0}^{n-1} w_i x^i y \\
\implies &\left(\sum_{i,j=0}^{n-1} a_i c_j x^{i+j} + \sum_{i,j=0}^{n-1} a_i d_j x^{i+j} y\right)\left(\sum_{k=0}^{n-1} b_k x^k y\right) = \sum_{i=0}^{n-1} v_i x^i + \sum_{i=0}^{n-1} w_i x^i y \\
\implies &\sum_{i,j,k=0}^{n-1} a_i c_j b_k x^{i+j+k} y + \sum_{i,j,k=0}^{n-1} a_i d_j b_k \lambda x^{i+j+k} = \sum_{i=0}^{n-1} v_i x^i + \sum_{i=0}^{n-1} w_i x^i y
\end{aligned}$$

Comparing coefficients, we have the following two equations

$$\sum_{i,j,k=0}^{n-1} a_i c_j b_k x^{i+j+k} y = \sum_{i=0}^{n-1} w_i x^i y, \tag{4.6}$$

$$\lambda \sum_{i,j,k=0}^{n-1} a_i d_j b_k x^{i+j+k} = \sum_{i=0}^{n-1} v_i x^i \tag{4.7}$$

Define vectors $\mathbf{a} = (a_0, \ldots, a_{n-1})^T$, $\mathbf{b} = (b_0, \ldots, b_{n-1})^T$, $\mathbf{c} = (c_0, \ldots, c_{n-1})^T$, $\mathbf{d} = (d_0, \ldots, d_{n-1})^T$, $\mathbf{w} = (w_0, \ldots, w_{n-1})^T$, $\mathbf{v} = (v_0, \ldots, v_{n-1})^T$ in $\mathbb{F}_{q^n}$. Here, $\mathbf{b}$ is a reversible vector, i.e. $b_i = b_{n-i}$ for each $i = 1, \ldots, n-1$. The vectors $\mathbf{a}$ and $\mathbf{b}$ are unknown to the adversary, while $\mathbf{c}$, $\mathbf{d}$, $\mathbf{v}$, and $\mathbf{w}$ are publicly known.

4.5.1 Reduction to matrix equations

In the below discussion, all subscripts are taken modulo n. The following lemma shows that Equation (4.6) can be reduced to a matrix equation over $\mathbb{F}_q$.

Lemma 4.9. Equation (4.6) is equivalent to the matrix equation $M_{\mathbf{z}}(\mathbf{b}, \mathbf{c}) \cdot \mathbf{a} = \mathbf{w}$ over $\mathbb{F}_q$.

Proof. Equating the coefficients of the basis vectors $x^i y$ in Equation (4.6), we have

$$\begin{aligned}
w_i &= \sum_{\ell=0}^{n-1} \sum_{(j,k)|j+k=\ell \pmod n} c_j b_k a_{i-\ell} \\
&= \sum_{\ell=0} \sum_{(j,k)|j+k=i-\ell \pmod n} c_j b_k a_\ell \\
&= \begin{pmatrix} z_i(\mathbf{b},\mathbf{c}) & z_{i-1}(\mathbf{b},\mathbf{c}) & \ldots & z_0(\mathbf{b},\mathbf{c}) & z_{n-1}(\mathbf{b},\mathbf{c}) & \ldots & z_{i+1}(\mathbf{b},\mathbf{c}) \end{pmatrix} \cdot \mathbf{a}
\end{aligned}$$

Thus, we can rewrite Equation (4.6) equivalently as the system

$$\begin{aligned} w_0 &= \big(z_0(\mathbf{b},\mathbf{c}) \quad z_{n-1}(\mathbf{b},\mathbf{c}) \quad \dots \quad z_1(\mathbf{b},\mathbf{c})\big)\cdot\mathbf{a} \\ w_1 &= \big(z_1(\mathbf{b},\mathbf{c}) \quad z_0(\mathbf{b},\mathbf{c}) \quad \dots \quad z_2(\mathbf{b},\mathbf{c})\big)\cdot\mathbf{a} \\ &\vdots \\ w_{n-1} &= \big(z_{n-1}(\mathbf{b},\mathbf{c}) \quad z_{n-2}(\mathbf{b},\mathbf{c}) \quad \dots \quad z_0(\mathbf{b},\mathbf{c})\big)\cdot\mathbf{a} \end{aligned}$$

In other words, $M_{\mathbf{z}}(\mathbf{b},\mathbf{c})\cdot\mathbf{a} = \mathbf{w}$. □

One may similarly rewrite Equation (4.7) as above, so that we have the following lemma.

Lemma 4.10. Equation (4.7) is equivalent to the matrix equation $\lambda M_{\mathbf{z}}(\mathbf{b},\mathbf{d})\cdot\mathbf{a} = \mathbf{v}$ over $\mathbb{F}_q$.

Combining the results of Lemmas 4.9 and 4.10, if the vectors $\mathbf{b}$,$\mathbf{c}$ and $\mathbf{d}$ are given, then $\mathbf{a}$ is a simultaneous solution to the matrix equations $M_{\mathbf{z}}(\mathbf{b},\mathbf{c})\cdot\mathbf{a} = \mathbf{w}$ and $\lambda M_{\mathbf{z}}(\mathbf{b},\mathbf{d})\cdot\mathbf{a} = \mathbf{v}$. However, a priori the vector $\mathbf{b}$ is unknown to the adversary. If we can find $\mathbf{b}$ such that this system of equations has a simultaneous solution, then we are done with reducing the DPD problem to a solving a single system of linear equations, which can be done in polynomial time. Summarizing this discussion, we have the following result.

Proposition 4.3. Suppose that a vector $\mathbf{b} = (b_0,\dots,b_{n-1})$ is such that the system of simultaneous equations $\lambda M_{\mathbf{z}}(\mathbf{b},\mathbf{d})\mathbf{a} = \mathbf{v}$ and $M_{\mathbf{z}}(\mathbf{b},\mathbf{c})\mathbf{a} = \mathbf{w}$ has a simultaneous solution $\mathbf{a} = (a_0,\dots,a_{n-1})$. Then, $s = \sum_{i=0}^{n-1} a_i x^i$, $t = \sum_{i=0}^{n-1} b_i x^i y$ is a solution of the equation $sht = \gamma$.

Now, for an adversary, the vectors $\mathbf{a}$ and $\mathbf{b}$ are both unknown. We will show below that in most cases, it suffices for the adversary to fix a suitable value for $\mathbf{b}$ and then proceed to solve any one of the linear equations in Lemmas 4.9 and 4.10 for $\mathbf{a}$. More precisely, we show that if $M_{\mathbf{c}}$ and $M_{\mathbf{d}}$ are invertible, then a solution is possible for any randomly chosen $\mathbf{b} \in \Gamma_\alpha$ for which the correponding circulant matrix $M_{\mathbf{b}}$ is invertible. Since the values arise from a legitimate public key, we know that there exists a vector $\mathbf{b} \in \Gamma_\alpha$ such that the equations $\lambda M_{\mathbf{z}}(\mathbf{b},\mathbf{d})\mathbf{a} = \mathbf{v}$ and $M_{\mathbf{z}}(\mathbf{b},\mathbf{c})\mathbf{a} = \mathbf{w}$ have a simultaneous solution $\mathbf{a}$.

Proposition 4.4. Let the vectors $\mathbf{c}$ and $\mathbf{d}$ be such that $M_{\mathbf{c}}$ and $M_{\mathbf{d}}$ are invertible. Assume that at least one simultaneous solution $(\mathbf{a},\mathbf{b})$ exists to the matrix equations $\lambda M_{\mathbf{z}}(\mathbf{b},\mathbf{d})\mathbf{a} = \mathbf{v}$ and $M_{\mathbf{z}}(\mathbf{b},\mathbf{c})\mathbf{a} = \mathbf{w}$. Then, for any randomly chosen $\mathbf{b} \in \Gamma_\alpha$ such that $M_{\mathbf{b}}$ is invertible, the equations $\lambda M_{\mathbf{z}}(\mathbf{b},\mathbf{d})\mathbf{a} = \mathbf{v}$ and $M_{\mathbf{z}}(\mathbf{b},\mathbf{c})\mathbf{a} = \mathbf{w}$ have a simultaneous solution $\mathbf{a}$ computable in polynomial time.

Proof. Here, $\mathbf{b}$, $\mathbf{c}$, and $\mathbf{d}$ are invertible, and thus so are $M_{\mathbf{z}}(\mathbf{b},\mathbf{d}) = M_{\mathbf{d}} \cdot M_{\mathbf{b}}$ and $M_{\mathbf{z}}(\mathbf{b},\mathbf{c}) = M_{\mathbf{c}} \cdot M_{\mathbf{b}}$. Now, we know that a solution $(\mathbf{a},\mathbf{b})$ exists, and so for some vectors $\mathbf{a}$ and $\mathbf{b}$ we have

$$\lambda M_{\mathbf{d}} M_{\mathbf{b}} \mathbf{a} = \mathbf{v}, \quad M_{\mathbf{c}} M_{\mathbf{b}} \mathbf{a} = \mathbf{w}, \quad \text{i.e. } \lambda^{-1} M_{\mathbf{d}}^{-1} \mathbf{v} = M_{\mathbf{b}} \mathbf{a}, \; M_{\mathbf{c}}^{-1} \mathbf{w} = M_{\mathbf{b}} \mathbf{a}$$

So, independently of $\mathbf{a}$ and $\mathbf{b}$ we necessarily have

$$\lambda^{-1} M_{\mathbf{d}}^{-1} \mathbf{v} = M_{\mathbf{c}}^{-1} \mathbf{w} \tag{4.8}$$

Now let $\mathbf{b}$ be any random vector such that $M_{\mathbf{b}}$ is invertible. Multiplying equation (4.8) by $M_{\mathbf{b}}^{-1}$, we get

$$\begin{aligned} & \lambda^{-1} M_{\mathbf{b}}^{-1} M_{\mathbf{d}}^{-1} \mathbf{v} = M_{\mathbf{b}}^{-1} M_{\mathbf{c}}^{-1} \mathbf{w} \\ \implies & \lambda^{-1} M_{\mathbf{z}}(\mathbf{b},\mathbf{d})^{-1} \mathbf{v} = M_{\mathbf{z}}(\mathbf{b},\mathbf{c})^{-1} \mathbf{w} \end{aligned}$$

Setting $\mathbf{a} := \lambda^{-1} M_{\mathbf{z}}(\mathbf{b},\mathbf{d})^{-1} M_{\mathbf{v}} = M_{\mathbf{z}}(\mathbf{b},\mathbf{c})^{-1}\mathbf{w}$, we get $\mathbf{a}$ as the simultaneous solution $\lambda M_{\mathbf{z}}(\mathbf{b},\mathbf{d})\mathbf{a} = \mathbf{v}$ and $M_{\mathbf{z}}(\mathbf{b},\mathbf{c})\mathbf{a} = \mathbf{w}$. □

4.5.2 The algorithm for cryptanalysis

Before describing the cryptanalysis algorithm, we state the following assumption, which we make for the sake of our complexity argument. We do not have a proof of it, but experimental evidence strongly suggests that it holds to a good approximation.

Assumption 1. Let P_1 denote the probability of a uniformly sampled $n \times n$ circulant matrix over $\mathbb{F}_q$ being invertible and P_2 denote the probability that the circulant matrix corresponding to a uniformly sampled reversible vector is invertible. Then $P_1 = P_2$. In other words, the probability distribution corresponding to invertibility remains the same when restricted to matrices corresponding to reversible vectors.

We have the following result.

Corollary 4.2. Let $M_{\mathbf{c}}$ and $M_{\mathbf{d}}$ be invertible and γ be a legitimate public key. Let e denote the largest power of p dividing n. Further, assume that Assumption 1 holds. Then, the equation $sht = \gamma$ in the unknowns $s \in \mathbb{F}_q^{\alpha} C_n, \; t \in \Gamma_{\alpha}$ can be solved for a legitimate secret key (s,t) in an expected $\mathcal{O}\left(\left(1 - \frac{1}{q}\right)^{-n/p^e}\right)$ steps. If n is a power of p then we have a constant time solution.

Proof. Since γ is a legitimate public key, a least one simultaneous solution $(\mathbf{a},\mathbf{b})$ exists (the one corresponding to the initial secret key) to the matrix equations

$\lambda M_{\mathbf{z}}(\mathbf{b},\mathbf{d})\mathbf{a} = \mathbf{v}$ and $M_{\mathbf{z}}(\mathbf{b},\mathbf{c})\mathbf{a} = \mathbf{w}$. Now, from Corollary 4.1, a vector $\mathbf{b} \in \mathbb{F}_{q^n}$ such that $\mathbf{b}$ is invertible can be found in an expected $\left\lceil \frac{1}{\left(1-\frac{1}{q}\right)^{n/p^e}} \right\rceil$ number of steps. For the solution of the DPD problem, one further requires that the vector $\mathbf{b}$ satisfies $b_i = b_{n-1}$ for $1 \leq i \leq n-1$, i.e. that $\mathbf{b} \in \Gamma_\alpha$. From Assumption 1, we may assume the same number of expected steps. Thus, by Proposition 4.4, we can set $\mathbf{b}$ to be any vector in Γ_α such that $M_{\mathbf{b}}$ is invertible. If n is a power of p, this quantity is $\left\lceil \frac{1}{1-\frac{1}{q}} \right\rceil$, which is decreasing in q, with the smallest value being 2, for $q = 2$. Thus, in this case the time complexity is $\mathcal{O}(1)$. This is also confirmed by experimental results, where randomly chosen symmetric vectors $\mathbf{b} \in \Gamma_\alpha$ were invertible in almost all trials. Once such a vector $\mathbf{b}$ is found, one can compute $\mathbf{a} = \lambda^{-1} M_{\mathbf{z}}(\mathbf{b},\mathbf{d})^{-1} M_{\mathbf{v}} = M_{\mathbf{z}}(\mathbf{b},\mathbf{c})^{-1}\mathbf{w}$. By Proposition 4.3, this gives a solution to the DPD problem $sht = \gamma$. □

We now state an algorithm to cryptanalyse the key exchange. Its correctness follows from the above discussion. All the parameters suggested for Protocol 4.1 by the authors of [24] use $n = p$, so this algorithm provides a constant-time cryptanalysis.

Algorithm 4.1: Cryptanalysis of Key Exchange over $\mathbb{F}_q^\alpha D_{2n}$

Input Parameter λ and the cocycle $\alpha = \alpha_\lambda$, public element $h = \sum_{i=0}^{n-1} c_i x^i + \sum_{i=0}^{n-1} d_i x^i y$, public key $\gamma = \sum_{i=0}^{n-1} v_i x^i + \sum_{i=0}^{n-1} w_i x^i y$.

Output A solution $(s,t) \in \mathbb{F}_q^\alpha C_n \times \Gamma_\alpha$ satisfying $sht = \gamma$. This tuple is a solution to the DPD problem.

1: Define vectors in $\mathbb{F}_{q^n}$: $\mathbf{c} := (c_0,\ldots,c_{n-1})$, $\mathbf{d} := (d_0,\ldots,d_{n-1})$, $\mathbf{v} := (v_0,\ldots,v_{n-1})$, $\mathbf{w} := (w_0,\ldots,w_{n-1})$.
2: If $M_{\mathbf{c}}$ or $M_{\mathbf{d}}$ is not invertible
  Return Fail
3: Pick a vector $\mathbf{b} = (b_0,\ldots,b_{n-1}) \leftarrow \Gamma_\alpha$ at random.
4: If $M_{\mathbf{b}}$ is not invertible, repeat Step (3). If it is invertible, go to Step (5).
5: Compute $\mathbf{a} = \lambda^{-1} M_{\mathbf{z}}(\mathbf{b},\mathbf{c})^{-1}\mathbf{w}$ $(= M_{\mathbf{b}}^{-1} M_{\mathbf{d}}^{-1}\mathbf{v})$.
6: With $\mathbf{a} = (a_0,\ldots,a_{n-1})$, set $s = \sum_{i=0}^{n-1} a_i x^i$ and $t = \sum_{i=0}^{n-1} b_i x^i y$.
7: Return (s,t).

Remark 4.2. The solution (s,t) to the DPD returned by Algorithm 4.1 and referenced in Corollary 4.2 is a legitimate secret key, but not necessarily the same as the originally chosen secret key. In fact, as is clear from the discussion above, $t = \sum_{i=0}^{n-1} b_i x^i y \in \Gamma_\alpha$ can be selected at random, and a solution for $s \in \mathbb{F}_q^\alpha C_n$ is found long as $M_{\mathbf{b}}$ is invertible.

Now, since $\mathbf{c}$ and $\mathbf{d}$ are randomly chosen in $\mathbb{F}_{q^n}$, the circulant matrices $M_{\mathbf{c}}$ and $M_{\mathbf{d}}$ are invertible with high probability. The probability that the algorithm fails is the probability that at least one of them is not invertible, which is given by $1-(1-\frac{1}{q})^2$. Clearly this quantity shrinks with increasing values of q and n. In [24] the smallest values of these parameters are $q = n = 19$, for which this probability is ≈ 0.1. Thus, Algorithm 4.1 succeeds in cryptanalyzing the system with a probability of at least 90 percent.

An immediate corollary of the above argument is that the two-sided multiplication action

$$(\mathbb{F}_q^\alpha C_n \times \Gamma_\alpha) \times \mathbb{F}_q^\alpha D_{2n} \to \mathbb{F}_q^\alpha D_{2n}$$

$$(s,t)\cdot h \mapsto sht,\ \ s \in \mathbb{F}_q^\alpha C_n,\ \ t \in \Gamma_\alpha$$

is far from being injective, contrary to the assumption of the authors. In fact, for most values of t and $\gamma \in \mathbb{F}_q^\alpha D_{2n}$, there is a unique pre-image $s \in \mathbb{F}_q^\alpha C_n$ such that $sht = \gamma$. Thus, the probability that random choosing yields the right solution is not $1/|\mathbb{F}_q^\alpha C_n \times \Gamma_\alpha|$, as claimed by the authors. The real probability is greater than or equal to probability that the matrices $M_{\mathbf{c}}$ and $M_{\mathbf{d}}$ are invertible and that the correct value of s corresponding to t is chosen, which is $\approx 1/|\mathbb{F}_q^\alpha C_n|$ (we already saw that the probability of the matrices being invertible is very close to 1). From this, one also sees that the run time of an exhaustive search would be linear in $|\mathbb{F}_q^\alpha C_n| = p^{nm}$, rather than in $|\mathbb{F}_q^\alpha C_n \times \Gamma_\alpha| = p^{nm}p^{m\lfloor\frac{n+1}{2}\rfloor}$, as claimed by the authors of [24].

4.6 Examples

In this section, we present some examples generated by computer search, using the algebra software package SageMath [87]. For the structure of the twisted group algebra and the generation of the keys, we made use of the original source code of the authors. Our entire working code including the cryptanalysis can be found at: https://github.com/simran-tinani/Cryptanalysis-of-twisted-group-algebra-system

In the following examples, an element $\sum_{i=0}^{n-1} a_i x^i + \sum_{i=0}^{n-1} b_i x^i y$ of $\mathbb{F}_q^\alpha D_{2n}$ is denoted by the $2n$-tuple $(a_0, \ldots, a_{n-1}, b_0, \ldots, b_{n-1})$. We use the notations above.

Example 4.1. For parameters $n = 23,\ q = 23,\ \lambda = 11$, consider a randomly gener-

ated public element h, and secret key $(s,t) \in \mathbb{F}_q^{\alpha} C_n \times \Gamma_{\alpha}$.

$h =(19,9,4,14,6,13,21,18,18,10,9,2,5,15,13,22,18,13,16,20,11,2,11,6,18,7,17,8,20,20,17,7,$
$15,1,11,9,17,4,11,16,5,17,19,18,19,20),$
$s =(20,17,20,22,18,18,11,12,2,3,18,11,2,18,3,14,10,2,13,14,3,9,17,0,0,0,0,0,0,0,0,0,0,0,$
$0,0,0,0,0,0,0,0,0,0,0)$
$t =(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,22,0,14,3,2,19,4,15,1,21,3,6,6,3,21,1,15,$
$4,19,2,3,14,0)$

Using the method in Section 4.5, the program computed the solution $(\tilde{s},\tilde{t})$ to the DPD, where

$\tilde{s} =(13,16,5,1,21,1,2,8,17,2,12,11,4,0,20,7,19,16,3,14,22,6,2,0,0,0,0,0,0,0,0,0,0,0,0,0,$
$0,0,0,0,0,0,0,0,0)$
$\tilde{t} =(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,7,2,17,16,12,16,12,10,6,8,3,0,0,3,8,6,10,$
$12,16,12,16,17,2)$

It was verified that $sht = \tilde{s}h\tilde{t}$, so a legitimate private key was recovered.

Example 4.2. For parameters $n = 19,\ q = 19,\ \lambda = 18$, and using the notations above, consider the randomly generated public element h, and secret key $(s,t) \in \mathbb{F}_q^{\alpha} C_n \times \Gamma_{\alpha}$.

$h =(14,5,13,4,10,12,8,6,17,18,15,1,14,14,15,15,13,4,6,7,7,11,13,4,11,12,3,11,18,8,3,3,6,$
$11,17,1,7,10),$
$s =(18,14,1,0,15,5,7,0,1,7,10,5,9,18,2,12,17,12,14,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)$
$t =(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,17,4,10,18,5,5,9,15,18,18,15,9,5,5,18,10,4,17)$

Using the method in Section 4.5, the program computed the solution $(\tilde{s},\tilde{t})$ to the DPD, where

$\tilde{s} =(12,6,4,10,12,4,5,7,0,15,8,7,1,0,2,15,6,7,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)$
$\tilde{t} =(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,14,13,11,10,3,3,1,1,3,16,16,3,1,1,3,3,10,11,13)$

It was verified that $sht = \tilde{s}h\tilde{t}$, so a legitimate private key was recovered.

Example 4.3. For parameters $n = 41,\ q = 41,\ \lambda = 29$, and using the notations above, consider the randomly generated public element h, and secret key $(s,t) \in \mathbb{F}_q^{\alpha} C_n \times \Gamma_{\alpha}$.

$h =(33,2,29,20,9,5,36,13,26,15,38,27,33,4,20,4,14,23,12,0,35,5,38,40,1,6,16,26,9,0,29,6,32,26,$
$14,32,18,29,13,35,7,8,38,26,20,25,24,18,30,28,22,8,21,1,33,29,2,22,25,6,13,24,18,26,30,38,3,$
$1,39,11,15,10,9,16,3,7,36,26,22,6,0,15),$
$s =(24,2,12,32,10,2,27,1,5,7,17,32,7,24,28,26,17,8,32,18,13,8,19,17,0,11,33,17,27,1,36,3,33,9,30,34,$
$22,26,21,5,29,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)$
$t =(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,13,8,18,11,31,$
$9,34,3,16,39,32,0,15,31,3,26,0,31,39,4,40,40,4,39,31,0,26,3,31,15,0,32,39,16,3,34,9,31,11,18,8)$

Using the method in Section 4.5, the program computed the solution $(\tilde{s},\tilde{t})$ to the DPD, where

$\tilde{s} =(39,9,4,23,8,8,10,40,31,27,22,36,11,14,35,28,25,0,0,10,16,33,24,6,33,17,15,13,17,10,18,31,33,16,13,$
$28,2,36,37,13,30,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)$
$\tilde{t} =(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,35,6,8,35,23,22,$
$39,12,22,36,34,1,29,8,16,40,29,16,24,14,31,31,14,24,16,29,40,16,8,29,1,34,36,22,12,39,22,23,35,8,6)$

It was verified that $sht = \tilde{s}h\tilde{t}$, so a legitimate private key was recovered.

Clearly, in each of the above examples, $s \neq \tilde{s}$ and $t \neq \tilde{t}$, but $sht = \tilde{s}h\tilde{t}$. Thus, each of these examples also serves as a counterexample to the injectivity of the two-sided action.

Chapter 5

Methods for Collisions in some Algebraic Hash Functions

5.1 Introduction

Let $\mathcal{A}$ be an alphabet and $\mathcal{A}^*$ denote the set of all finite-length words in $\mathcal{A}$ and $\mathcal{A}^n$ denote the set of all words up to length n in $\mathcal{A}$. A length n hash function, or compression function, is a map $\mathcal{A}^* \to \mathcal{A}^n$ which takes messages of arbitrary length to fixed-length message digests. A hash function $h : \mathcal{A}^* \to \mathcal{A}^n$ is called a cryptographic hash function if it satisfies the following properties:

- Collision-resistance: it is computationally infeasible to find a pair x, x' of distinct messages such that $h(x) = h(x')$.
- Second pre-image resistance: given a message x, it is computationally infeasible to find another message $x' \neq x$ such that $h(x) = h(x')$.
- One-wayness: given a hash value $y \in \mathcal{A}^n$ it is computationally infeasible to find a pre-image $x \in \mathcal{A}$ such that $h(x) = y$.

Cryptographic hash functions are also often required to exhibit the avalanche effect, under which a small modification in the message text causes a big change in the hash. This prevents the hash value from leaking information about the message string, and ensures no visible correlation between the hashes of related strings. A hash function that does not exhibit this property is called mallaeble.

Several widely used hash functions, including the NIST-standardized SHA (Secure Hash Algorithms) functions [73], are built from block ciphers. While the state of the art block cipher-based hash functions have shown considerable resiliency to attacks, their security is nevertheless heuristic; in other words, it does not reduce to a well-known difficult mathematical problem. The search for a provably secure hash function is therefore, at the least, of theoretical interest.

The idea of building hash functions from a group and its corresponding Cayley graph was introduced in [109]. The generic design of Cayley hash functions has several advantages over traditional hash functions. Firstly, their security is equivalent

to some concise mathematical problem. Cayley hash functions are also inherently parallelizable, i.e. allow for simultaneous computation of the hash value of different parts of the message, and recombining these at the end. While the Tillich-Zémor hash functions are slower than SHA-256, they can be designed to have reasonably efficient implementations: in [78] it is stated that they can be made faster than SHA-1. In particular, when fields of characteristic 2 are used, the group law is the most efficient.

There may also be other disadvantages of Cayley hash functions. By themselves, such hash functions are inherently mallaeble: given a hash $h(m)$ of an unknown message m, $h(x_1||m||x_2) = h(x_1)h(m)h(x_2)$ for any texts x_1, x_2. Further, they lack preimage resistance for small messages. However, in [78], the authors describe a heuristic modification which seemingly resolves these issues, as well as enhances the efficiency. Cayley hash functions have a simple, elegant, clear design related to hard mathematical problems. A very generic description is given below.

Definition 5.1 (Cayley hash function). Let G be a finite group with a set of generators S and $\mathcal{A}$ be an alphabet the same size as S. Given an injective map $\pi : \mathcal{A} \to S$, one may define the hash value of the message $x_1x_2 \dots x_k$ to be the group element $\pi(x_1)\pi(x_2) \dots \pi(x_k)$.

Typically, one is concerned only with binary messages, and therefore with two-generator Cayley hash functions. The common design principle behind the Cayley hash functions is the performance of a walk on a regular Cayley graph according to the bits of an input message, the last vertex giving the hash value.

Definition 5.2 (Cayley Graph). Let G be a group and S a subset of elements of G. The Cayley graph of $C_{G,S} = (V, E)$ of G with respect to S is defined to have vertices v_g corresponding to each element $g \in G$, and edges $(v_{g_1}, v_{g_2}) \in E \iff \exists s \in S$ such that $g_2 = g_1 s$. Here, the set S is the set of graph generators.

The security of a Cayley hash function defined on a group G is determined by the properties of the corresponding graph. More precisely, the difficulty of producing collisions for such a hash function depends on the difficulty of solving the *factorization problem* in a certain finite non-abelian group.

Definition 5.3 (Factorization problem). Let G be a group with generators $S = \{s_1, \dots, s_k\}$ and $L > 0$ be a fixed constant. Given $g \in G$, return $m_1, \dots, m_L$ and $\ell \leq L$, with $m_i \in \{1, \dots, k\}$ such that $\prod_{i=1}^{\ell} s_{m_i} = g$. The factorization problem for $g = 1$ fixed is called the representation problem.

Apart from hash functions, the difficulty of factorization in finite matrix groups has been used to build some key exchange, encryption and authentication schemes [6,

7, 9].

Factorization in finite groups is related to the famous conjecture of Babai on the diameter of Cayley graphs, which states that diameter of any undirected Cayley graph of non-abelian simple group is polylogarithmic in group size. In other words, "short" factoriations always exist for finite non-abelian groups, regardless of the choice of generating set. The conjecture is known to be true for certain groups like $SL_2(2, \mathbb{F}_p), SL_2(2, \mathbb{F}_{2^k})$. However, existing generic proofs for any generating set are non-constructive. In [108], the following Cayley hash function was defined in the group $G = SL_2(\mathbb{F}_p)$:

Definition 5.4 (Zémor Hash Function). Choose the generators $A_0 = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$, $A_1 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$ of $SL_2(\mathbb{F}_p)$. For a message $m = m_1 m_2 \dots m_k \in \{0,1\}^*$ define $H(m_1 \dots m_k) = A_{m_1} \dots A_{m_k}$.

This hash function, given in [108] was attacked for both collisions and preimages in [97] using the Euclidean algorithm. However, this attack is specific to the generators A_0 and A_1. In [77], it is claimed that the system is potentially secure by replacing A_0 and A_1 by A_0^2 and A_1^2. We define the *generalized Zémor hash function* as above, but using generators $A_0 = \begin{pmatrix} 1 & \alpha \\ 0 & 1 \end{pmatrix}$ and $A_1 = \begin{pmatrix} 1 & 0 \\ \beta & 1 \end{pmatrix}$ over $\mathbb{F}_{p^k}$. While the generators A_0 and A_1 have multiplicative order p, and so one trivially has collisions of length p with the empty word, we nevertheless also consider the case $k > 1$ and study the difficulty of finding non-trivial collisions (i.e. collisions of non-empty words).

The following variant of the above hash function was later proposed in [98].

Definition 5.5 (Tillich-Zémor Hash function). Let $n > 0$ and $p(x)$ be an irreducible polynomial over $\mathbb{F}_2$. Write $K = \mathbb{F}_2[x]/p(x)$. Consider $A_0 = \begin{pmatrix} x & 1 \\ 1 & 0 \end{pmatrix}$ and $A_1 = \begin{pmatrix} x & x+1 \\ 1 & 1 \end{pmatrix}$, which are generators of $SL_2(K)$. For a message $m = m_1 m_2 \dots m_k \in \{0,1\}^*$ define $H(m_1 \dots m_k) = A_{m_1} \dots A_{m_k} \pmod{p(x)}$. $K = \mathbb{F}_2[x]/\langle q(x) \rangle \cong F_{2^n}$.

Collisions for this hash function for these set of generators were found in [39]. This attack uses the structure in hash values of palindromic messages, and a result of Mesirov-Sweet [62] on the Euclidean algorithm for polynomials x and $x+1$ in characteristic 2. This attack was extended to recover preimages in [76]. Since current attacks do not allow controlling of the forms of collisions, and work specifically for the set of generators A_0 and A_1, the security is an open problem for general

parameters.

In [74], the authors provide a new heuristic algorithm for factoring generic generator sets by reducing them to contain a "trapdoor" matrix. Their algorithm is subexponential in time, memory, and factorization lengths. This algorithm, however, does not yield a practical attack on the Tillich-Zémor design with generic generators for $n = 160$. In fact, to the best of knowledge till date, the collision and preimage resistance are recovered by replacing A_0 and A_1 by $B_0 = \begin{pmatrix} x^2 & 1 \\ 1 & 0 \end{pmatrix}$ and $B_1 = \begin{pmatrix} x+1 & 1 \\ 1 & 0 \end{pmatrix}$ [79]. We define the *generalized Tillich-Zémor hash function* as above, but using generators $A_0 = \begin{pmatrix} \alpha & 1 \\ 1 & 0 \end{pmatrix}$ and $A_1 = \begin{pmatrix} \beta & 1 \\ 1 & 0 \end{pmatrix}$ where $\alpha, \beta \in \mathbb{F}_{p^k}$.

It has also been proposed to use LPS Ramanujan graphs [96] to construct Cayley hash functions. However, these have also been cryptanalyzed [75]. Some other methods of cryptanalysis for Cayley hash functions are discussed in [77].

Collisions from Triangular and Diagonal Matrices In [79], it is shown that if one can produce "sufficiently many" messages whose images in the matrix groups are upper/lower triangular, then one can find collisions of the generalized Zémor and Tillich-Zémor hash functions.

Proposition 5.1. Let n be such that discrete logarithms can be solved in $\mathbb{F}_{2^n}^*$. Let $\mathcal{D}$, $\mathcal{T}^{up}$, $\mathcal{T}^{low}$, $\mathcal{L}^v$, $\mathcal{R}^v \subset SL_2(\mathbb{F}_{2^n})$ be the subgroups of diagonal, upper and lower triangular matrices and the subgroup of matrices with left or right eigenvector v. If an attacker can compute N random elements M_i of one of these subgroups together with bit sequences m_i of length at most L hashing to these matrices, then he can also find a message m such that $H_{ZT}(m) = I$. The message m has expected size smaller than $NL2^{n/N}$ in the diagonal case and smaller than $NL2^{1+n/N}$ in the other cases.

In this same work, the authors use random probabilistic search to find pre-images of upper/lower triangular matrices, and subsequently uses these to produce collisions. This approach works for any Cayley hash function, and for SL_2 groups over any finite field.

Summary of Contributions

This chapter focuses on devising methods for producing collisions in algebraic hash functions that may be seen as generalized forms of the well-known Zémor and Tillich-

Zémor hash functions. In contrast to some of the previous approaches, we attempt to construct collisions in a structured and deterministic manner.

In Section 5.2, we introduce a method for constructing messages with triangular or diagonal hashes messages. For this, we extend existing hash values in $SL_2(\mathbb{F}_{p^k})$ into triangular or diagonal form by multiplying with products of the form $A_0^m A_1^n$, where $A_0 = \begin{pmatrix} 1 & \alpha \\ 0 & 1 \end{pmatrix}$ and $A_1 = \begin{pmatrix} 1 & 0 \\ \beta & 1 \end{pmatrix}$ denote generators. More precisely, we consider the following problem.

Problem 5.1. Given a matrix $C \in SL_2(\mathbb{F}_{p^k})$ formed as product of A_0 and A_1, find the conditions under which there exist integers m and n (of size significantly smaller than p^k) such that $CA_0^m A_1^n$ is upper/lower triangular, or even diagonal. Compute m and n if they exist.

We also discuss the application of this method to produce collisions, and the feasibility and efficiency thereof. Our method thus provides an alternate deterministic approach to the method for finding triangular hashes in [79].

In Section 5.3, we consider the generalized Tillich-Zémor hash functions over $\mathbb{F}_{p^k}$ for $p \neq 2$, relating the generator matrices to a polynomial recurrence relation, and derive a closed form for any arbitrary power of the generators. We then provide conditions for collisions, and a method to maliciously design the system so as to facilitate easy collisions, in terms of this polynomial recurrence relation.

On simplifying the general criteria, and through experiments, our general conclusion is that it is very difficult in practice to achieve the theoretical collision conditions efficiently, in both the generalized Zémor and the generalized Tillich-Zémor cases. Therefore, although the techniques are interesting theoretically, in practice the collision-resistance of the generalized Zémor functions is reinforced.

5.2 Generalized Zémor Hash functions

Definition 5.6 (Generalized Zémor hash functions). Consider the generators $A_0 = \begin{pmatrix} 1 & \alpha \\ 0 & 1 \end{pmatrix}$ and $A_1 = \begin{pmatrix} 1 & 0 \\ \beta & 1 \end{pmatrix}$ in the group $SL_2(\mathbb{F}_{p^k})$. For a message $m = m_1 m_2 \dots m_k \in \{0,1\}^*$ define the hash value $H(m_1 \dots m_k) = A_{m_1} \dots A_{m_k}$.

The Zémor hash function proposed in [108] is a special case of this, with $\alpha = 1 = \beta$. In [77] it was suggested that security is preserved when s_0^2 and s_1^2 are used instead, these cases correspond to the values $\alpha = 2 = \beta$. Thus, the generalized Zémor hash functions are so far secure.

Clearly, $A_0^m = \begin{pmatrix} 1 & m\alpha \\ 0 & 1 \end{pmatrix}$ and $A_1^n = \begin{pmatrix} 1 & 0 \\ n\beta & 1 \end{pmatrix}$ for any integers m and n. A_0 and A_1 have multiplicative orders p, so trivial collisions of length p with the empty word always exist. Nevertheless, we study the general case of non-trivial collisions in the finite field $\mathbb{F}_{p^k}$. In general, one is interested in finding collisions with a significantly smaller length than the orders, at most, say $\mathcal{O}(\sqrt{p})$.

5.2.1 Euclidean Algorithm Attack for $\alpha = \beta = 1$

In this subsection, we describe the attack of [97], which uses the Euclidean algorithm, and explain why it fails if one chooses $\alpha \neq 1$, $\beta \neq 1$ in the hash function design. Consider a matrix $X = \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) \in SL_2(\mathbb{F}_p)$ and suppose that we have found a non-identity matrix $Y = \left(\begin{smallmatrix} A & B \\ C & D \end{smallmatrix}\right) \in SL_2(\mathbb{Z})$ that reduces modulo p to X.Let $A_0 = \left(\begin{smallmatrix} 1 & 1 \\ 0 & 1 \end{smallmatrix}\right)$, $A_1 = \left(\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right)$. Consider the general case with generators $\tilde{A}_0 = A_0^\alpha = \left(\begin{smallmatrix} 1 & \alpha \\ 0 & 1 \end{smallmatrix}\right)$, $\tilde{A}_1 = A_1^\beta = \left(\begin{smallmatrix} 1 & 0 \\ \beta & 1 \end{smallmatrix}\right)$.

We write

$$Y = \begin{pmatrix} A & B \\ C & D \end{pmatrix} = \begin{pmatrix} a_i & b_i \\ c_i & d_i \end{pmatrix} A_0^{\alpha q_i} A_1^{\beta q_{i-1}} \dots A_0^{\alpha q_2} A_1^{\beta q_1}$$

where $\begin{pmatrix} a_0 & b_0 \\ c_0 & d_0 \end{pmatrix} = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$. Since the matrix X is obtained as a product of $\tilde{A}_0$ and $\tilde{A}_1$, there exists an integer n such that $\begin{pmatrix} a_n & b_n \\ c_n & d_n \end{pmatrix} = A_0^{\alpha p_n}$ or $\begin{pmatrix} a_n & b_n \\ c_n & d_n \end{pmatrix} = A_0^{\alpha p_n} = A_1^{\beta q_n}$. We wish to determine the values of the q_i, $1 \leq i \leq n$ for such an n. We have, for $1 \leq i \leq n$,

$$\begin{aligned} a_i &= a_{i-1} - \beta b_{i-1} q_{i-1} \\ b_i &= b_{i-1} - \alpha a_{i-1} q_i \\ c_i &= c_{i-1} - \beta d_{i-1} q_{i-1} \\ d_i &= d_{i-1} - \alpha c_{i-1} q_i \end{aligned} \tag{5.1}$$

where $\begin{pmatrix} a_0 & b_0 \\ c_0 & d_0 \end{pmatrix} = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$ are the only fixed values of the a_i, b_i, c_i, d_i's and α, β are treated as integers smaller than p.

Case $\alpha = \beta = 1$ First assume that $\alpha = \beta = 1$. Since $AD - BC = 1$, $\gcd(A, B) = \gcd(C, D) = 1$, and we have either $A > B$ or $D > C$. We consider the case $A > B$, the argument for other case is analogous. Let $q_1, \dots, q_n$ be the quotients that appear

when Euclidean algorithm is applied to (A, B), i.e.

$$\begin{aligned} A &= Bq_1 + r_1 \\ B &= r_1 q_2 + r_2 \\ &\vdots \\ r_i &= r_{i+1} q_{i+2} + r_{i+2} \\ &\vdots \\ r_{n-2} &= r_{n-1} q_n = \gcd(A, B) q_n = q_n \end{aligned}$$

Note that $YA_1^{-q_1}A_0^{-q_2} = \left(\begin{smallmatrix} r_1 & r_2 \\ C - Dq_1 & -q_2(C - Dq_1) + D \end{smallmatrix}\right)$, and for i odd, $YA_1^{-q_1}A_0^{-q_0} \ldots A_1^{-q_i} = \left(\begin{smallmatrix} r_i & r_{i-1} \\ \star & \star \end{smallmatrix}\right)$ and $YA_1^{-q_1}A_0^{-q_0} \ldots A_1^{-q_i}A_0^{-q_{i+1}} = \left(\begin{smallmatrix} r_i & r_{i+1} \\ \star & \star \end{smallmatrix}\right)$. If n is even then $YA_1^{-q_1}A_0^{-q_2} \ldots A_0^{-q_n} = \left(\begin{smallmatrix} 1 & 0 \\ q & 1 \end{smallmatrix}\right) = A_1^x$, for some $q \in \mathbb{Z}$, $(q < C)$ since $r_{n-1} = 1$. In other words, we have $Y = A_0^{q_n}A_1^{q_{n-1}} \ldots A_0^{q_2}A_1^{q_1+q}$ where $q_1, \ldots, q_n, q$ are obtained when the Euclidean algorithm is applied to (A, B). The practical application of this principle to produce collisions of reasonable length is demonstrated in [97]. This gives a polynomial time solution for collisions.

Case $\alpha \neq 1,\ \beta \neq 1$ In the case with general α, β, there is no clear protocol to solve the set of equations (5.1) for the q_i's. For instance, we start with the assumption that $A > B$, but it may still happen that $A \leq \beta B$, and then there is no clear way to apply the Euclidean algorithm.

5.2.2 Extending messages for diagonal hashes over $\mathbb{F}_p$

In this subsection, we discuss the extension of messages to produce diagonal hashes in $SL_2(\mathbb{F}_p)$. The below lemma demonstrates the production of a diagonal matrix starting with the hash of an arbitrary message.

For a matrix C, we denote by $C[i, j]$ the $(i, j)^{\text{th}}$ entry of C. Further, a message is denoted in its binary string representation $m = 0^{n_1}1^{m_1} \ldots 0^{n_r}1^{m_r}$ where multiplication corresponds to concatenation, and for $b \in \{0, 1\}$, the string b^n denotesf the concatenation of n consecutive b-bits.

Lemma 5.1. Let z be any message and $C := H(z) \in SL_2(\mathbb{F}_p)$ be its corresponding hash value. Assume that $a := C[0, 0] \neq 0$. Then there exist integers $m, n \in \{0, 1, \ldots, p-1\}$ such that $C \cdot A_0^m \cdot A_1^n$ is a diagonal matrix and $(C \cdot A_0^m \cdot A_1^n)[0, 0] = C[0, 0]$.

Proof. Write $C = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$. Set $m = -b/(a\alpha)$ and $n = -ac/\beta$. Then

$$C \cdot A_0^m \cdot A_1^n = \begin{pmatrix} a(1+mn\alpha\beta) + nb\beta & 0 \\ c(1+mn\alpha\beta) + nd\beta & mc\alpha + d \end{pmatrix}$$

Now, $c(1+mn\alpha\beta)+nd\beta = c(1+bc)-acd = c-c(ad-bc) = 0$, since $C \in SL_2(\mathbb{F}_{p^k})$. Further, $A(1+mn\alpha\beta)+nb\beta = a(1+bc)-abc = a$, and $mc\alpha+d = -bc/a+d = 1/a$. Thus, $C \cdot A_0^m \cdot A_1^n = \begin{pmatrix} a & 0 \\ 0 & a^{-1} \end{pmatrix}$. □

Generating Collisions

In [79], the authors describe a method which uses the representation of 1 and pre-images of multiple diagonal matrices to produce a collision with the hash $H()$ of the empty word. We show here that Lemma 5.1 can be used to generate collisions in an alternative fashion. Let z be any message and $C := H(z) \in SL_2(\mathbb{F}_p)$ be its hashed value. Observe that for any integers m and n, the matrix $D := A_1^n \cdot C \cdot A_0^m$ satisfies $D[0,0] = C[0,0]$. Pick arbitrary distinct integers n_1, m_1, n_2, m_2 and set $D_1 := A_1^{n_1} \cdot C \cdot A_0^{m_1}$ and $D_2 := A_1^{n_2} \cdot C \cdot A_0^{m_2}$. Then clearly, $D_1[0,0] = C[0,0] = D_2[0,0]$. Now apply Lemma 1 and find integers $\tilde{m}_1, \tilde{m}_2, \tilde{n}_1, \tilde{n}_2$ such that $\tilde{D_1} := D_1 \cdot A_0^{\tilde{m}_1} \cdot A_1^{\tilde{n}_1}$ and $\tilde{D_2} = D_2 \cdot A_0^{\tilde{m}_2} \cdot A_1^{\tilde{n}_2}$ are diagonal and satisfy $\tilde{D_1}[0,0] = D_1[0,0] = C[0,0]$ and $\tilde{D_2}[0,0] = D_2[0,0] = C[0,0]$. Since $\tilde{D_1}$ and $\tilde{D_2}$ are diagonal, have the same entry, and lie in $SL_2(\mathbb{F}_p)$, they must be equal, i.e. $\tilde{D_1} = \tilde{D_2}$. Defining $z_1 = 1^{n_1} \cdot z \cdot 0^{m_1+\tilde{m}_1} \cdot 1^{\tilde{n}_1}$ and $z_2 = 1^{n_2} \cdot z \cdot 0^{m_2+\tilde{m}_2} \cdot 1^{\tilde{n}_2}$. Clearly, $z_1 \neq z_2$ but $H(z_1) = H(z_2)$.

Example 5.1. For $p = 7919$, $\alpha = 5698$, $\beta = 6497$, consider the message text

$$\begin{aligned} z =& 0^{44}1^{41}0^{17}1^{49}0^{47}1^{17}0^{50}1^{31}0^{15}1^{10}0^{39}1^{12}0^{2}1^{0}0^{24}1^{41}0^{28}1^{23}0^{9}1^{0}0^{47}1^{23}0^{1}1^{30}0^{18} \\ & 1^{32}0^{24}1^{14}0^{0}1^{49}0^{19}1^{28}0^{24}1^{26}0^{26}1^{26}0^{11}1^{1}0^{17}1^{20}0^{38}1^{22}0^{12}1^{38}0^{8}1^{33}0^{39}1^{42}0^{47}1^{29} \\ & 0^{10}1^{41}0^{14}1^{45}0^{13}1^{40}0^{42}1^{13}0^{2}1^{6}0^{40}1^{31}0^{2}1^{27}0^{1}1^{7}0^{36}1^{19}0^{3}1^{25}0^{10}1^{27}0^{21}1^{2}0^{12}1^{23} \\ & 0^{36}1^{8}0^{25}1^{39}0^{36}1^{0}0^{19}1^{39}0^{37}1^{32}0^{14}1^{4}0^{3}1^{12}0^{16}1^{23}0^{49}1^{25}0^{23}1^{19}0^{46}1^{23}0^{36}1^{31} \end{aligned}$$

We have, $H(z) = \begin{pmatrix} 4812 & 5537 \\ 4987 & 1690 \end{pmatrix}$. Choose random numbers $m_1 = 18, n_1 = 30$, $m_2 = 35$, $n_2 = 33$ and compute $\tilde{m_1} = 6208, \tilde{n_1} = 744$, $\tilde{m_2} = 6191$, $\tilde{n_2} = 180$ using Lemma 5.1. Then for $z_1 = 1^{30} \cdot z \cdot 0^{6226}1^{744}$ and $z_2 = 1^{33} \cdot z \cdot 0^{6226}1^{180}$ we have the collision $H(z_1) = H(z_2) = \begin{pmatrix} 4812 & 0 \\ 0 & 1542 \end{pmatrix}$.

It is clear that the above method provides a deterministic technique to produce arbitratrily many distinct non-trivial collisions of size $\mathcal{O}(p)$. For the collisions to

be of practical length, we further require that the values of the exponents m and n are considerably small relative to p so that the resulting messages are of reasonable length.

Experimental results We looked for such m and n by random computer search in the above examples, using randomly generated messages of reasonable length and checking for the condition $m = -b/(a\alpha) < \sqrt{p}$ and $n = -ac/\beta < \sqrt{p}$. Unfortunately, for larger values of p, experiments indicate a very low probability of finding such values. For $30 - 40$ digit primes, brute force could no longer find any such examples.

The below proposition describes a more structured approach for extending messages of the form $T = A_0^r A_1^s$.

Proposition 5.2. Consider a bound δ. If there exist integers $r < \delta$, $y > p - \delta$ such that $s := (y^{-1} - r^{-1})(\alpha\beta)^{-1} < \delta$ (where inverses are taken mod p) and $s(1 + rs\alpha\beta) > p - \delta$, then for the message $T = A_0^r A_1^s$ (of length $\leq 2\delta$), there exist $m, n < \delta$ such that $H(T)A_0^m A_1^n$ is diagonal.

Proof. We have $H(T) = \begin{pmatrix} 1 + rs\alpha\beta & r\alpha \\ s\beta & 1 \end{pmatrix}$. As per Lemma 5.1, the condition for $H(T)A_0^m A_1^n$ diagonal is that $m = -r/(1 + rs(\alpha\beta))$ and $n = -s(1 + rs(\alpha\beta))$. Write $x = (1 + rs(\alpha\beta))$. For $m, n < \delta$ we require $rx^{-1} > p - \delta$, and $sx > p - \delta$. We set $y := rx^{-1}$, from this we derive $s = (y^{-1} - r^{-1})(\alpha\beta)^{-1}$. Clearly, if $y > p - \delta$ and $s(1 + rs\alpha\beta) > p - \delta$ then $m = p - y < \delta$ and $n = p - sx < \delta$, and by assumption, $r, s < \delta$, so the messages $T = A_0^r A_1^s$ and $A_0^m A_1^n$ each have length $\leq 2\delta$ and $H(T)A_0^m A_1^n$ is a diagonal matrix. □

This clearly yields an algorithm with time complexity $\mathcal{O}(\delta^2)$ to test if for given values of α, β, a message of the form can be extended to have a diagonal hash. Indeed, one needs only to test all integers $r < \delta$, $y > p - \delta$ for the two conditions given in Proposition 5.2.

5.2.3 Extending messages for triangular hashes over $\mathbb{F}_{p^k}$

We now turn our attention to the more general case of producing upper or lower triangular hashes. In this subsection, we will consider the general case of $k \geq 1$ for the generalized Zémor hash functions over $\mathbb{F}_{p^k}$, and study the feasibility of efficiently producing messages hashing to upper triangular messages.

Transforming the hash values of a message into an upper or lower triangular matrix leads to producing collisions due to the following observation, which was explained in Propositon 3 of [79]. Suppose that one can produce r distinct messages z_i such

that $H(z_i) = \begin{pmatrix} a_i & b_i \\ 0 & d_i \end{pmatrix}$ is upper triangular. Also let e_i be exponents such that $\prod_{i=1}^{r} a_i^{e_i} = 1$. In other words, $(e_1, \ldots, e_r)$ is a solution to the representation problem in $\mathbb{F}_{p^k}^*$, which in turn reduces to a discrete log problem, since the multiplicative group of a finite field is cyclic. Then, for $z = z_1 \ldots z_r$ we have $H(z) = \begin{pmatrix} 1 & b \\ 0 & 1 \end{pmatrix}$ and thus $H(z||z) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ which collides with the hash value of the empty word.

We begin by proving some preliminary results below.

Lemma 5.2. Suppose that the product $\alpha \cdot \beta \in \mathbb{F}_p$ and that $C = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ is an arbitrary product of finitely many copies of A_0 and A_1. Then, $c/\beta \in \mathbb{F}_p$ and $d \in \mathbb{F}_p$.

Proof. Write $C = A_0^{m_1} A_1^{n_1} \ldots A_0^{m_r} A_1^{n_r} := \begin{pmatrix} a_r & b_r \\ c_r & d_r \end{pmatrix}$. The proof proceeds by induction on r. For the case $r = 1$, we have $C = \begin{pmatrix} 1 + mn\alpha\beta & m\alpha \\ n\beta & 1 \end{pmatrix}$. Clearly, $c_1/\beta = n_1 \in \mathbb{F}_p$ and $d = 1 \in \mathbb{F}_p$, so the result holds.

Now assume that the result holds for $r \geq 1$. Write $m := m_{r+1}$ and $n := n_{r+1}$. We have

$$C_{r+1} = \begin{pmatrix} a_r & b_r \\ c_r & d_r \end{pmatrix} \cdot \begin{pmatrix} 1 + mn\alpha\beta & m\alpha \\ n\beta & 1 \end{pmatrix} = \begin{pmatrix} a_r(1 + mn\alpha\beta) + nb_r\beta & a_r m\alpha + b_r \\ c_r(1 + mn\alpha\beta) + nd_r\beta & c_r m\alpha + d_r \end{pmatrix}$$

Now,

$$\begin{aligned} c_{r+1}^p/\beta^p =& 1/\beta^p[c_r^p(1 + mn\alpha\beta)^p + nd_r^p\beta^p] \\ =& 1/\beta^p[c_r^p(1 + mn\alpha\beta) + nd_r^p\beta^p] \\ =& (c_r/\beta)^p(1 + mn\alpha\beta) + nd_r^p. \end{aligned}$$

By the induction hypothesis we have $d_r^p = d_r \in \mathbb{F}_p$ and $(c_r/\beta)^p = c_r/\beta \in \mathbb{F}_p$, and so $c_{r+1}^p/\beta^p = (c_r/\beta)(1 + mn\alpha\beta) + nd_r = c_{r+1}/\beta \in \mathbb{F}_p$. Further,

$$\begin{aligned} d_{r+1}^p = (c_r m\alpha + d_r)^p =& mc_r^p\alpha^p + d_r^p = mc_r^p\alpha^p + d_r = mc_r \cdot c_r^{p-1}\alpha^p + d_r \\ =& mc_r \cdot \beta^{p-1}\alpha^p + d_r \\ =& mc_r \cdot (\alpha\beta)^{p-1}\alpha + d_r \\ =& mc_r\alpha + d_r = d_{r+1} \end{aligned}$$

Hence, we have $c_{r+1}/\beta \in \mathbb{F}_p$ and $d_{r+1} \in \mathbb{F}_p$. By induction, the statement holds for all $r \geq 1$ and hence the lemma holds. □

Lemma 5.3. Let $k \geq 1$ and $\alpha \cdot \beta \in \mathbb{F}_p$. Let z be any message and $C := H(z)$ be its corresponding hash value. Assume that $a := C[0,0] \neq 0$. Then, there exist integers $m, n \in \{0, 1, \ldots, p-1\}$ such that $C \cdot A_0^m \cdot A_1^n$ is upper triangular.

Proof. Write $C = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$. For integers m and n, we have

$$\begin{aligned} C \cdot A_0^m \cdot A_1^n &= \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} 1 + mn\alpha\beta & m\alpha \\ n\beta & 1 \end{pmatrix} \\ &= \begin{pmatrix} a(1 + mn\alpha\beta) + nb\beta & ma\alpha + b \\ c(1 + mn\alpha\beta) + nd\beta & mc\alpha + d \end{pmatrix} \end{aligned}$$

Choose m to be any integer such that $mc\alpha + d \neq 0$. Since $\alpha\beta \in \mathbb{F}_p$, $c/\beta \in \mathbb{F}_p$ and $d \in \mathbb{F}_p$, we also have $c\alpha = (c/\beta) \cdot \alpha \cdot \beta \in \mathbb{F}_p$, and we can define

$$n := -\frac{c}{\beta(mc\alpha + d)} \in \mathbb{F}_p \tag{5.2}$$

This gives $C \cdot A_0^m \cdot A_1^n = \begin{pmatrix} a(1 + mn\alpha\beta) + nb\beta & ma\alpha + b \\ 0 & mc\alpha + d \end{pmatrix}$. □

Corollary 5.1. Let $k \geq 1$ and $\alpha \cdot \beta \in \mathbb{F}_p$. Let δ be a bound. Let $C = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ be the hash of an arbitrary message and $m \in \{0, 1, \ldots, p-1\}$ be an integer such that both m and $n = -c/(\beta(mc\alpha + d)) \in \mathbb{F}_p$ are smaller than δ. Then $CA_0^m A_1^n$ has length at most 2δ more than C and is upper triangular.

Experimental results Again, we looked for such m and n by random computer search in the above examples, using randomly generated messages of reasonable length and checking for the condition above. Once again, for larger values of p, experiments indicate a very low probability of finding such values. For $30-40$ digit primes, brute force could no longer find any such examples.

Below, we explore a more structured approach for messages of the form $T = A_0^r A_1^s$.

Lemma 5.4. Let $k \geq 1$ and $\alpha \cdot \beta \in \mathbb{F}_p$. Consider a bound δ. If there exist integers $m, s < \delta$ such that $s(ms\alpha\beta+1)^{-1} > p-\delta$, then for the message $T = A_0^r A_1^s$ (of length $\leq 2\delta$, here r can be chosen freely), for $n = p - s(ms\alpha\beta + 1)^{-1} < \delta$, $H(T)A_0^m A_1^n$ is upper triangular.

Proof. We have $H(T) = \begin{pmatrix} 1 + rs\alpha\beta & r\alpha \\ s\beta & 1 \end{pmatrix}$. Let $m, s < \delta$ be integers such that $s(ms\alpha\beta + 1)^{-1} > p - \delta$. The condition for $H(T)A_0^m A_1^n =$

$$\begin{pmatrix} 1+rs\alpha\beta & r\alpha \\ s\beta & 1 \end{pmatrix}\begin{pmatrix} 1+mn\alpha\beta & m\alpha \\ n\beta & 1 \end{pmatrix} = \begin{pmatrix} (1+rs\alpha\beta)(1+mn\alpha\beta)+rn\alpha\beta & r\alpha \\ s\beta(1+mn\alpha\beta)+n\beta & 1+ms\alpha\beta \end{pmatrix}$$

being upper triangular is that $n = -s(1+ms(\alpha\beta))^{-1}$. The result is now clear. □

This clearly yields an algorithm with time complexity $\mathcal{O}(\delta^2)$ to test if for given values of α, β, a message of the form $T = A_0^r A_1^s$ can be extended to one with an upper triangular hash, for a fixed bound δ. Indeed, one needs only to test all integers $m, s < \delta$ for the two conditions given above.

Clearly, for $C \cdot A_0^m \cdot A_1^n$ to be upper triangular, n needs to assume the value in equation (5.2). Below, we derive the general condition (without the assumption $\alpha \cdot \beta \in \mathbb{F}_p$) on m so that both the m and n lie in $\mathbb{F}_p$.

Proposition 5.3. If $\alpha \cdot \beta \notin \mathbb{F}_p$, then $C \cdot A_0^m \cdot A_1^n$ is upper triangular for $m, n \in \mathbb{F}_p$ if and only if for

$$\gamma = \left(\frac{d((d\beta)^{p-1} - c^{p-1})}{\alpha c^p (1-(\alpha\beta)^{p-1})} \right), \tag{5.3}$$

we have $\gamma^p = \gamma$, and $m = \gamma$; $n = \dfrac{-c}{\beta(mc\alpha + d)}$.

Proof. From the proof of Lemma 5.3, it is clear that for $CA_0^m A_1^n$ to be upper triangular, Equation (5.2) must hold, along with $m^p = m$ and $n^p = n$. We have,

$$\begin{aligned} & -n = c/(\beta(m\alpha c + d)) \in \mathbb{F}_p \\ \implies & c^p/(\beta^p(m\alpha^p c^p + d^p)) = c/(\beta(m\alpha c + d)) \\ \implies & (c/\beta)^{p-1} \cdot (m\alpha c + d) = m\alpha^p + d^p \\ \implies & c^{p-1} \cdot (m\alpha c + d) = \beta^{p-1} \cdot (m\alpha^p c^p + d^p) \\ \implies & m\alpha c^p \cdot (1-(\alpha\beta)^{p-1}) = \beta^{p-1} d^p - c^{p-1} d \\ \implies & m = \frac{d((d\beta)^{p-1} - c^{p-1})}{\alpha c^p (1-(\alpha\beta)^{p-1})} \end{aligned}$$

Thus, for $m = \gamma$, one necessarily has $n := \dfrac{-c}{\beta(mc\alpha + d)} \in \mathbb{F}_p$. So, in order to obtain an upper triangular matrix, the only condition that needs to be satisfied is $\gamma^p = \gamma$. □

Lemma 5.5 (Case $k = 2$). Let $k = 2$ and $\alpha \cdot \beta \notin \mathbb{F}_p$. As before, let $C = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in SL_2(\mathbb{F}_{p^2})$ be an arbitrary product of finitely many copies of A_0 and A_1. Then with γ defined as in (5.3), $\gamma^p = \gamma$ always holds.

Proof. In this case, for any nonzero field element y we have $y^{p^2-p} = y^{1-p}$, so

$$\begin{aligned}\gamma^p = \left(\frac{d((d\beta)^{p-1} - c^{p-1})}{\alpha c^p(1-(\alpha\beta)^{p-1})}\right)^p &= (d^p c/\alpha^p)\left(\frac{((d\beta)^{1-p} - c^{1-p})}{(1-(\alpha\beta)^{1-p})}\right) \\ &= (d^p/c\alpha)1/(d\beta c)^{p-1}(\alpha\beta)^{p-1}\left(\frac{((d\beta)^{p-1} - c^{p-1})}{(1-(\alpha\beta)^{p-1})}\right) \\ &= d/(\alpha c^p)\left(\frac{((d\beta)^{p-1} - c^{p-1})}{(1-(\alpha\beta)^{p-1})}\right) = \gamma.\end{aligned}$$

□

Thus, in $SL_2(\mathbb{F}_{p^2})$ one can always find collisions by right-multiplying an arbitrary matrix by a product $A_0^m A_1^n$. In other words, a message in $SL_2(\mathbb{F}_{p^2})$ is always extendable to have a triangular hash.

Example 5.2. Consider $SL_2(\mathbb{F}_{p^2})$ for $p = 239$, with generator denoted by z. Consider the following message text

$$\begin{aligned}T =& 0^{17}1^{25}0^{11}1^{8}0^{38}1^{33}0^{29}1^{27}0^{40}1^{24}0^{4}1^{2}0^{38}1^{16}0^{16}1^{39}0^{50}1^{42}0^{36}1^{22}0^{2}1^{41}0^{27}1^{29}0^{11} \\ & 1^{15}0^{29}1^{47}0^{29}1^{33}0^{47}1^{3}0^{35}1^{32}0^{49}1^{27}0^{24}1^{0}0^{21}1^{49}0^{33}1^{4}0^{50}1^{44}0^{42}1^{43}0^{4}1^{29}0^{14}1^{39} \\ & 0^{14}1^{15}0^{41}1^{0}0^{41}1^{1}0^{20}1^{34}0^{4}1^{6}0^{43}1^{5}0^{11}1^{7}0^{37}1^{29}0^{40}1^{20}0^{1}1^{13}0^{2}1^{14}0^{31}1^{41}0^{19}1^{24} \\ & 0^{50}1^{50}0^{25}1^{23}0^{30}1^{39}0^{6}1^{46}0^{39}1^{27}0^{8}1^{9}0^{25}1^{38}0^{0}1^{46}0^{15}1^{33}0^{47}1^{40}0^{40}1^{26}0^{48}1^{45}\end{aligned}$$

For a random choice of α and β, we may obtain hash $H(T) = \begin{pmatrix} 134z+110 & 131z+185 \\ 74z+17 & 58z+41 \end{pmatrix}$. Calculating m and n as per Lemma 5.3, we see that $H(T)0^1 1^{10} = \begin{pmatrix} 106z+192 & 25z+30 \\ 0 & 218z+62 \end{pmatrix}$.

A natural question arises here: can we generalize this method to make

$$C \cdot A_0^{m_1} A_1^{n_1} \dots A_0^{m_r} A_1^{n_r}$$

upper/lower triangular and thereby extend the result to all $SL_2(\mathbb{F}_{p^k})$? Since the major constraint is the condition $\gamma^p = \gamma$ (and the size of γ does not seem easy to control), we first explore how the value of γ changes when C is multiplied on the right by a random product $A_0^m A_1^n$.

Theorem 5.1. Let $C = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ and $C' := C \cdot A_0^m \cdot A_1^n = \begin{pmatrix} a' & b' \\ c' & d' \end{pmatrix}$. Let γ and γ' be defined as in equation (5.3). Then, we have

$$\gamma' = (c/c')^{p+1}(\gamma - m)$$

Proof. Let

$$X := \frac{1}{\alpha(1-(\alpha\beta)^{p-1})},\ \gamma := \frac{Xd}{c^p}[(d\beta)^{p-1} - c^{p-1}]$$

We have,

$$\begin{aligned}
\gamma' =& X\frac{d'((d'\beta)^{p-1} - (n\beta d' + c)^{p-1}))}{(c')^p} = Xd'\frac{(d'\beta)^{p-1} - \frac{n\beta(d')^p + c^p}{n\beta d' + c}}{(c')^p} \\
=& Xd'\frac{(d'\beta)^{p-1}(n\beta d + c)) - n\beta(d')^p - c^p}{(c')^{p+1}} \\
=& Xd'c\frac{(d'\beta)^{p-1} - c^{p-1}}{(c')^{p+1}} = Xd'c\frac{(d'\beta)^{p-1} - c^{p-1}}{(c')^{p+1}} \\
=& \frac{Xcd'}{c'^{p+1}d'\beta}[(mc^p\alpha^p)\beta^p - c^{p-a1}(cm\alpha + d)\beta] \\
=& \frac{Xc}{c'^{p+1}\beta}[mc^p\alpha\beta((\alpha\beta)^{p-1} - 1) + d\beta((d\beta)^{p-1} - c^{p-1})]
\end{aligned}$$

Plugging the values of X and γ from above into the equation, we get

$$\begin{aligned}
\gamma' =& \frac{Xc}{(c')^{p+1}\beta}mc^p\alpha\beta((\alpha\beta)^{p-1} - 1) + \frac{Xcd}{(c')^{p+1}}\left(\frac{c^p\gamma}{Xd}\right) \\
=& \left(\frac{c}{c'}\right)^{p+1}[-m + \gamma]
\end{aligned}$$

□

For an extension where multiplication by a product $A_0^m A_1^n$ is allowed twice, we have the following result.

Lemma 5.6. For $C := \begin{pmatrix} a & b \\ c & d \end{pmatrix}$, there exits integers m_1, m_2, n_1, n_2 such that $CA^{m_1}B^{n_1}A^{m_2}B^{n_2}$ is upper triangular if and only if the equation

$$q_3x^2y + q_2xy + q_1y + q_0 = 0 \tag{5.4}$$

has a solution $(x, y) \in \mathbb{F}_p \times \mathbb{F}_p$, where q_0, q_1, q_2, q_3 are given by

$$\begin{aligned}
q_3 &= c^{p^2}\alpha\beta((\alpha\beta)^{p^2-1} - 1), \\
q_2 &= c^{p^2}\gamma\alpha\beta(\gamma^{p-1} - (\alpha\beta)^{p^2-1}) + d\beta((d\beta)^{p^2-1} - 1), \\
q_1 &= d\beta\gamma(c^{p^2}\gamma^{p-1} - (d\beta)^{p^2-1}), \\
q_0 &= c^{p^2}\gamma(\gamma^{p-1} - 1).
\end{aligned} \tag{5.5}$$

Proof. Let $C' := C \cdot A^m \cdot B^n = \begin{pmatrix} a' & b' \\ c' & d' \end{pmatrix}$ and let γ' be defined as in equation (5.3). By Theorem 5.1 we have $\gamma' := (c/c')^{p+1}(\gamma - m) \in \mathbb{F}_p$. We treat $x := m$ and $y := n$ as variables in $\mathbb{F}_p$. For C' to be upper triangular for some values of x and y, we require

$$
\begin{aligned}
& (\gamma')^p = \gamma' \\
\implies & c^{p^2+p}(c')^{p+1}(\gamma^p - x) = c^{p+1}(c')^{p^2+p}(\gamma - x) \\
\implies & c^{p^2} c'(\gamma^p - x) = c(c')^{p^2}(\gamma - x) \\
\implies & c^{p^2}(\gamma^p - x)(c + (c\alpha x + d)y\beta) = c(c^{p^2}) + (xc^{p^2}\alpha^{p^2} + d^{p^2})y\beta^{p^2})(\gamma - x) \\
\implies & c^{p^2-1}[\gamma^p c + \gamma^p c\alpha\beta xy + \gamma^p d\beta y - cx - c\alpha\beta x^{2y - d\beta xy} = c^{p^2}\gamma + (c\alpha\beta)^{p^2}\gamma xy + \\
& \qquad (d\beta)^{p^2}\gamma y - c^{p^2}x - (c\alpha\beta)^{p^2}x^2 y - (d\beta)^{p^2} xy \\
\implies & ((c\alpha\beta)^{p^2} - c^{p^2}\alpha\beta)x^2 y + (-(c\alpha\beta)^{p^2}\gamma + \gamma^p c^{p^2}\alpha\beta + (d\beta)^{p^2} - d\beta)xy + \\
& \qquad (c^{p^2}\gamma^p(d\beta) - \gamma(d\beta)^{p^2})y + (\gamma^p c^{p^2} - \gamma c^{p^2}) = 0 \\
\implies & q_3 x^2 y + q_2 xy + q_1 y + q_0 = 0
\end{aligned}
$$

where q_0, q_1, q_2, q_3 are given as in Equation (5.5). The proof is now complete. □

Note that for q_0, q_1, q_2, q_3 as in Equation (5.5), we can rephrase the condition in Lemma 5.6 as follows. $C := \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ is upper triangularizable by right multiplication if and only if the system of equations

$$
\begin{aligned}
& q_3 x^2 y + q_2 xy + q_1 y + q_0 = 0 \\
& x^p = x \\
& y^p = y
\end{aligned}
\tag{5.6}
$$

has a simultaneous solution (x, y) in $\mathbb{F}_{p^k} \times \mathbb{F}_{p^k}$. One may then use Gröbner bases to solve this system of equations. A corresponding conditions for C to be lower triangularizable by left/right multiplication and to be upper triangularizable by left multiplication can be derived similarly.

Example 5.3. For simplicity, consider the field $\mathbb{F}_{2^5}$ with generator z_5 and $\alpha = z_5^3 + 1$, $\beta = z_5^3 + z_5^2 + 1$. Consider the hash matrix

$$
C = \begin{pmatrix} z_5^4 + z_5^3 + z_5^2 + z_5 & z_5^4 + z_5^3 + z_5^2 + z_5 \\ z_5^3 & z_5^4 + z_5^3 + z_5^2 \end{pmatrix}.
$$

Here, we have $\gamma = z_5^4 + z_5 + 1$ and the polynomial in Equation (5.4) is $(z_5^2 + z_5)x^2 y + (z_5^3 + z_5^2 + 1)xy + (z_5^3)y + (z_5^4 + z_5^2 + z_5)$. The $\langle (z_5^2 + z_5)x^2 y + (z_5^3 + z_5^2 + 1)xy + z_5^3 y + (z_5^4 + z_5^2 + z_5), x^p - x, y^p - y \rangle$ is trivial, so its Gröbner basis is $\{1\}$.

We attempted the above experiment with various different values of p and k and random matrices C which are products of A and B. In each case, without exception, we found no solution (i.e. the resulting Gröbner base of the ideal $\langle q_3x^2y + q_2xy + q_1y + q_0, x^p - x, y^p - y\rangle$ was $\langle 1\rangle$). Therefore, this method of extension to produce a triangular matrix is not practically feasible.

5.3 Generalized Tillich-Zémor Hash Functions

In this section, we consider the generalized Tillich-Zémor hash function ϕ with the generators $A_0 = \begin{pmatrix}\alpha & 1\\ 1 & 0\end{pmatrix}$ and $A_1 = \begin{pmatrix}\beta & 1\\ 1 & 0\end{pmatrix}$ where $\alpha, \beta \in \mathbb{F}_{p^k}$.

More generally, treating x as a variable, we consider the matrix $Y = \begin{pmatrix}x & 1\\ 1 & 0\end{pmatrix}$ and first compute its powers. Clearly, we have

$$Y^2 = \begin{pmatrix}x^2+1 & x\\ x & 1\end{pmatrix},\ Y^3 = \begin{pmatrix}x^3+x^2+x+1 & x^2+1\\ x^2+1 & x\end{pmatrix}$$

More generally, we can write

$$Y^n = \begin{pmatrix}f_n(x) & f_{n-1}(x)\\ f_{n-1}(x) & f_{n-2}(x)\end{pmatrix},\ n \geq 2 \tag{5.7}$$

where $f_0(x) = 0$, $f_1(x) = 1$, and

$$f_n(x) = xf_{n-1}(x) + f_{n-2}(x) \tag{5.8}$$

It is clear that the recurrence relation (5.8) fully describes the powers of the matrix Y. The above formulation and recurrence relation were also derived in [1].

5.3.1 Computing $f_n(x)$ for characteristic $p \neq 2$

In [1], an expression for $f_n(x)$ was derived in the case $p = 2$. The authors also calculate probabilities that the orders of A and B lie within certain bounds. In this section, we address the case $p \neq 2$ and derive a closed formula for $f_n(x)$.

We may solve the recurrence Equation (5.8) by finding roots of the auxiliary polynomial $t^2 - xt - 1 = 0$. The quadratic formula then gives us

$$t = \frac{x \pm \sqrt{x^2+4}}{2}.$$

Setting $r = \dfrac{x+\sqrt{x^2+4}}{2}$ and $s = \dfrac{x-\sqrt{x^2+4}}{2}$, a general solution to (5.8) has to be of the form $f_n(x) = ur^n + vs^n$. Note that we also have the initial conditions $f_1(x) = 1$ and $f_0(x) = 1$ (or equivalently $f_2(x) = x^2+1$). Plugging these in, we may solve for u and v to get

$$u = \frac{x+\sqrt{x^2+4}}{2\sqrt{x^2+4}},\ v = 1-u = \frac{-x+\sqrt{x^2+4}}{2\sqrt{x^2+4}}.$$

Write $w = \sqrt{x^2+4}$. Note that for any $n \geq 1$, we have

$$\begin{aligned}
f_n(x) = ur^n + vs^n &= \left(\frac{x+w}{2w}\right)\left(\frac{(x+w)^n}{2}\right) + \left(\frac{-x+w}{2w}\right)\left(\frac{(x-w)^n}{2}\right) \\
&= \left(\frac{(x+w)^{n+1}}{2^{n+1}w}\right) - \left(\frac{(x-w)^{n+1}}{2^{n+1}w}\right) \\
&= \frac{1}{2^{n+1}w}\left[\sum_{i=0}^{n+1}\binom{n+1}{i}x^i w^{n+1-i} - \sum_{i=0}^{n+1}\binom{n+1}{i}x^i w^{n+1-i}\right] \\
&= \frac{1}{2^{n+1}}\left[\sum_{0\leq i\leq n+1,\ n-i \text{ is even}}\binom{n+1}{i}x^i w^{n-i}\right] \\
&= \frac{1}{2^{n+1}}\left[\sum_{0\leq i\leq n+1,\ n-i \text{ is even}}\binom{n+1}{i}\sum_{j=0}^{(n-i)/2}\binom{(n-i)/2}{j}x^{i+2j}4^{(n-i)/2-j}\right] \\
&= \frac{1}{2^{n+1}}\left[\sum_{0\leq i\leq n,\ n-i \text{ is even}}\sum_{j=0}^{(n-i)/2}\binom{n+1}{i}\binom{(n-i)/2}{j}2^{n-2j}x^{i+2j}\right]
\end{aligned} \tag{5.9}$$

Thus, $f_n(x)$ is always a polynomial in $\mathbb{F}_p[x]$, and we have a closed formula for calculating it. Powers of A_0 and A_1 may therefore be computed in constant time.

Note that $\mathbb{F}_{p^k}$ is typically viewed through the isomorphism $\mathbb{F}_{p^k} \cong \mathbb{F}_p[x]/\langle p(x)\rangle$ where $p(x)$ is an irreducible polynomial of degree k over $\mathbb{F}_p$. Thus, an element $\gamma \in \mathbb{F}_{p^k}$ is a polynomial of degree smaller than k, say $\gamma = g_\gamma(x)$. $f_n(\gamma)$ can then be calculated as a polynomial modulo $p(x)$ by simply composing f_n and g, i.e. $f_n(\gamma) = f_n(g_\gamma(x)) \pmod{p(x)}$.

We also have the following straightforward observation. A similar statement is found in [1].

Lemma 5.7. Let $q(x)$ be any irreducible polynomial of degree d. Then, the sequence $\{f_n(x) \pmod{p(x)}\}$ is periodic, and its order divides $(p^{2k}-p^k)(p^{2k}-1)/(p^k-1)$.

Proof. Clearly, for $Y = \begin{pmatrix} x & 1 \\ 1 & 0 \end{pmatrix}$, Equation (5.7) gives us a formula for Y^n in terms of $f_n(x)$, $f_{n-1}(x)$ and $f_{n-2}(x)$. Since Y must have a finite multiplicative order n_y in the finite group $SL_2(\mathbb{F}_p[x]/\langle q(x)\rangle)$, we must have $f_{n_y-1}(x) = 0$, $f_n(x) = f_{n-2}(x) = 1$. From this point onwards, the sequence repeats its terms, and therefore its period is given by n_y, which must divide the order of the matrix group. □

Lemma 5.8. Suppose that the adversary can compute integers m and n such that $f_{n-1}(g_\alpha(x)) = f_{m-1}(g_\beta(x)) \pmod{p(x)}$ and $f_{n-2}(g_\alpha(x)) = f_{m-2}(g_\beta(x)) \pmod{p(x)}$. Then, the adversary can compute a collision of size $\mathcal{O}(\max(m, n))$ for the Generalized Tillich-Zëmor hash function ϕ.

Proof. In this case, it is clear from (5.8) that $f_n(g_\alpha(x)) = f_m(g_\beta(x)) \pmod{p(x)}$, and so $A_0^n = A_1^m$, i.e. $\phi(0^n) = \phi(1^m)$. □

Note that the above result includes the cases $m = 0$ and $n = 0$, i.e. finding collisions with the identity matrix. However, by looking at the expression (5.9) for $f_n(x)$, one sees that even for the simplest equation $f_n(x) = 0 \pmod{p(x)}$, finding a solution for the value of n is not straightforward, since in (5.9) n occurs both as a polynomial term (in the binomial coefficients) and in the exponent of 2. This problem seems to be a complex amalgamation of a generalized discrete logarithm with a polynomial rather than a single power. One may extend the above result to products with more terms.

Lemma 5.9. Let $\mathbb{F}_p[x]/\langle q(x)\rangle$ be a finite field. If an adversary can find integers m and n such that the following relations hold

$$\begin{aligned}
&f_m(f_n(x)) + f_{m-1}(f_{n-1}(x)) = 1 \pmod{q(x)} \\
&f_m(f_{n-1}(x)) + f_{m-1}(f_{n-2}(x)) = 0 \pmod{q(x)} \\
&f_{m-1}(f_n(x)) + f_{m-2}(f_{n-1}(x)) = 0 \pmod{q(x)} \\
&f_{m-1}(f_{n-1}(x)) + f_{m-2}(f_{n-2}(x)) = 1 \pmod{q(x)},
\end{aligned}$$

then $H(0^m 1^n) = H()$ gives a collision with the hash $H()$ of the empty word.

Note that in both the above results, the problem potentially becomes even more difficult if one constrains the values of m and n to yield practical-sized collisions.

5.3.2 Malicious paramaters

As seen in the above discussion, the security of the generalized Tillich-Zëmor hash functions relies on the choice of the irreducible polynomial through which the finite field is constructed. Thus, a malicious construction of this polynomial can lead to the designer of the hash function being able to easily compute collisions.

In [17], the authors produce malicious polynomials in finite fields $\mathbb{F}_{2^m}$, reducing the problem of collision-finding in the matrix group to the search for an irreducible polynomial of degree m, such that $m+1$ is a proper divisor of $q-1$ or $q+1$, where $q=2^m$. Over the field so constructed, the matrix A has a small order 263, and leads also to a non-trivial collision, which the authors exemplify for $SL_2(\mathbb{F}_{2^{131}})$. This attack fails if $q-1$ and $q+1$ are both primes or if the factorization of their gcd involves large primes, since the collisions yielded are then too long.

In [1], the authors also deal exclusively with the characteristic 2 case, and propose an algorithm to decide whether the given irreducible polynomial leads to a vulnerable system under the attack of [17], and propose a solution to fix the vulnerability. In the following result, we describe a malicious design for the choice of the polynomial defining the finite field, for odd characteristic p.

Theorem 5.2. If one can find N such that $\gcd(f_N(x)-1, f_{N-1}(x))$ has an irreducible divisor $q(x)$ of degree d, one can find a collision of size $\mathcal{O}(N)$ for the hash function $\phi(x)$ over the finite field $\mathbb{F}_p[x]/\langle q(x)\rangle$. On the other hand, given a fixed finite field $\mathbb{F}_p[x]/\langle q(x)\rangle$, if one can find an integer N such that $q(x)$ divides $\gcd(f_N(x)-1, f_{N-1}(x))$ then one can find collisions of size $\mathcal{O}(N)$ for ϕ.

Proof. If one can find N such that $\gcd(f_N(x), f_{N-1}(x))$ has an irreducible divisor $q(x)$ of degree d, then $f_N(x) = 1 \pmod{q}(x)$, $f_{N+1}(x) = 0 \pmod{q}(x)$, then the sequence $\{f_n(x) \pmod{\mathbb{F}_p[x]}/\langle q(x)\rangle\}$ has a period dividing n_y, and the multiplicative order of Y in $SL_2(\mathbb{F}_p[x]/\langle q(x)\rangle)$ divides N. Thus, one has a collision of $H(0^N)$ with the hash of the empty word. □

Example 5.4. $q(x) = x^{12} + 2x^{10} + x^6 + 2x^4 + 2x^3 + x^2 + x + 1$ is an irreducible polynomial over $\mathbb{F}_3$ such that $\{f_n(x) \pmod{q}(x)\}$ has period $531440 = 3^{12} - 1$. Thus, in $SL_2(\mathbb{F}_{3^{12}})$, we always have collisions $H(0^{531440}) = H(1^{531440}) = H()$.

Bibliography

[1] Kanat S. Abdukhalikov and Chul Kim. "On the Security of the Hashing Scheme Based on SL_2". In: *Fast Software Encryption Workshop*. 1998. DOI: 10.1007/3-540-69710-1_7.

[2] L. M. Adleman and J. DeMarrais. "A subexponential algorithm for discrete logarithms over all finite fields". In: *Math. Comp.* 61.203 (1993), pp. 1–15. DOI: 10.1090/S0025-5718-1993-1190642-2.

[3] M. Agrawal, N. Kayal, and N. Saxena. "PRIMES is in P". In: *Ann. of Math. (2)* 160.2 (2004), pp. 781–793. DOI: 10.4007/annals.2004.160.781.

[4] Navid Alamati et al. "Cryptographic Group Actions and Applications". In: *Advances in Cryptology – ASIACRYPT 2020*. Ed. by Shiho Moriai and Huaxiong Wang. Cham: Springer International Publishing, 2020, pp. 411–439. DOI: 10.1007/978-3-030-16852-6_15.

[5] Iris Anshel, Michael Anshel, and Dorian Goldfeld. "An algebraic method for public-key cryptography". In: *Mathematical Research Letters* 6.3-4 (1999), pp. 287–291. DOI: 10.4310/MRL.1999.v6.n3.a1.

[6] Iris Anshel et al. "Ironwood meta key agreement and authentication protocol". In: *Advances in Mathematics of Communications* 15.3 (2021), pp. 397–413. DOI: 10.3934/amc.2020081.

[7] Iris Anshel et al. "Kayawood, a Key Agreement Protocol". In: *IACR Cryptol. ePrint Arch.* 2017 (2017), p. 1162. DOI: 10.13154/eprint.iacr.org.2017.1162.

[8] Iris Anshel et al. "Key agreement, the Algebraic Eraser™, and lightweight cryptography". In: 418 (2007), pp. 1–34. DOI: 10.1090/conm/418.

[9] Iris Anshel et al. "WalnutDSA™: a group theoretic digital signature algorithm". In: *International Journal of Computer Mathematics: Computer Systems Theory* 6.4 (2017), pp. 260–284. DOI: 10.1080/23799927.2017.1292164.

[10] M. Banin and B. Tsaban. "A Reduction of Semigroup DLP to Classic DLP". In: *Des. Codes Cryptography* 81.1 (Oct. 2016), pp. 75–82. ISSN: 0925-1022. DOI: 10.1007/s10623-015-0130-2.

[11] Adi Ben-Zvi, Simon R. Blackburn, and Boaz Tsaban. "A Practical Cryptanalysis of the Algebraic Eraser". In: *Proceedings, Part I, of the 36th Annual International Cryptology Conference on Advances in Cryptology — CRYPTO 2016 - Volume 9814*. Springer-Verlag, 2016, pp. 179–189. DOI: 10.1007/978-3-662-53018-4_7.

[12] Adi Ben-Zvi, Arkadius Kalka, and Boaz Tsaban. "Cryptanalysis via algebraic spans". In: *Annual International Cryptology Conference*. Springer. 2018, pp. 255–274. DOI: 10.1007/978-3-030-00470-5_10.

[13] Ward Beullens and Simon R. Blackburn. "Practical attacks against the Walnut digital signature scheme". In: *IACR Cryptol. ePrint Arch.* 2018. DOI: 10.13154/tosc.v2021.i1.119-150.

[14] Ran Canetti and Hugo Krawczyk. "Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels". In: *Advances in Cryptology — EUROCRYPT 2001*. Ed. by Birgit Pfitzmann. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 453–474. DOI: 10.1007/3-540-44987-6_28.

[15] A. Caranti. "A module-theoretic approach to abelian automorphism groups". In: *Israel Journal of Mathematics* 205 (2015), pp. 235–246. DOI: https://doi.org/10.1007/s11856-014-1106-z.

[16] Bren Cavallo and Delaram Kahrobaei. "A family of polycyclic groups over which the uniform conjugacy problem is NP-complete". In: *International Journal of Algebra and Computation* 24.04 (2014), pp. 515–530. DOI: 10.1142/S0218196714500234. eprint: https://doi.org/10.1142/S0218196714500234.

[17] Chris Charnes and Josef Pieprzyk. "Attacking the SL_2 hashing scheme". In: *Advances in Cryptology — ASIACRYPT'94*. Ed. by Josef Pieprzyk and Reihanah Safavi-Naini. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 322–330. DOI: 10.1007/3-540-48658-5_29.

[18] Li Chen and Bin Fu. "Linear and Sublinear Time Algorithms for Basis of Abelian Groups". In: *Proceedings of the 20th International Symposium on Algorithms and Computation*. ISAAC '09. Honolulu, Hawaii: Springer-Verlag, 2009, pp. 493–502. DOI: 10.1007/978-3-642-10631-6_51.

[19] A.M. Childs and G. Ivanyos. "Quantum computation of discrete logarithms in semigroups". In: *Journal of Mathematical Cryptology* 8.4 (2014), pp. 405–416. DOI: 10.1515/jmc-2013-0038.

[20] H. Cohen et al., eds. *Handbook of Elliptic and Hyperelliptic Curve Cryptography*. Discrete Mathematics and its Applications (Boca Raton). Chapman & Hall/CRC, Boca Raton, FL, 2006, pp. xxxiv+808. DOI: 10.1201/9781420034984.

[21] Keith Conrad. "Groups of order p^3". In: *Expository papers on group theory* (2014). DOI: 10.1080/00927872.2013.868921.

[22] Thomas H Cormen et al. *Introduction to algorithms*. MIT press, 2009. DOI: 10.5555/58011.

[23] Jean-Marc Couveignes. *Hard Homogeneous Spaces*. Cryptology ePrint Archive, Paper 2006/291. 2006. URL: https://eprint.iacr.org/2006/291.pdf.

[24] Javier de la Cruz and Ricardo Villanueva-Polanco. "Public key cryptography based on twisted dihedral group algebras". In: *Advances in Mathematics of Communications* 16.2 (2022), pp. 195–215. DOI: 10.3934/amc.2021115.

[25] M. J. Curran. "Direct Products with Abelian Automorphism Groups". In: *Communications in Algebra* 35.1 (2006), pp. 389–397. DOI: 10.1080/00927870601042225. eprint: https://doi.org/10.1080/00927870601042225.

[26] Javier De La Cruz and Wolfgang Willems. "Twisted Group Codes". In: *IEEE Transactions on Information Theory* 67.8 (2021), pp. 5178–5184. DOI: 10.1109/TIT.2021.3089003.

[27] Whitfield Diffie and Martin Hellman. "New Directions in Cryptography". In: *IEEE Transactions on Information Theory* 22.6 (1976), pp. 644–654. DOI: 10.1109/TIT.1976.1055638.

[28] Mohammad Eftekhari. "Cryptanalysis of Some Protocols Using Matrices over Group Rings". In: *Progress in Cryptology - AFRICACRYPT 2017*. Cham: Springer International Publishing, 2017, pp. 223–229.

[29] Bettina Eick and Delaram Kahrobaei. *Polycyclic groups: A new platform for cryptology?* 2004. arXiv: math/0411077 [math.GR].

[30] Benjamin Fine et al. "Aspects of nonabelian group based cryptography: a survey and open problems". In: *arXiv preprint. arXiv:1103.4093* (2011).

[31] Robert W Floyd. "An algorithm for finding shortest paths in a graph". In: *Communications of the ACM* 3.11 (1960), pp. 611–616. DOI: 10.1145/367177.367199.

[32] David Freeman. "The Discrete Logarithm Problem in Matrix Groups". In: (2004). URL: https://theory.stanford.edu/~dfreeman/papers/discretelogs.pdf.

[33] Volker Gebhardt. "Efficient Collection in Infinite Polycyclic Groups". In: *Journal of Symbolic Computation* 34.3 (2002), pp. 213–228. ISSN: 0747-7171. DOI: 10.1006/jsco.2002.0559.

[34] Oliver W. Gnilke and Jens Zumbrägel. "Cryptographic Group and Semigroup Actions". In: *arXiv preprint, arXiv: 2301.01657* (2023). DOI: 10.48550/ARXIV.2301.01657.

[35] Oliver Wilhelm Gnilke. "The Semigroup Action Problem in Cryptography". PhD thesis. University College Dublin, 2015.

[36] N. Goel, I. Gupta, and B. K. Dass. "Survey on SAP and its application in public-key cryptography". In: *Journal of Mathematical Cryptology* 14.1 (2020), pp. 144–152. DOI: 10.1515/jmc-2016-0004.

[37] María Dolores Gómez Olvera, Juan Antonio López Ramos, and Blas Torrecillas Jover. "Public Key Protocols over Twisted Dihedral Group Rings". In: *Symmetry* 11.8 (2019). ISSN: 2073-8994. DOI: 10.3390/sym11081019.

[38] D. Gorenstein. *Finite Groups*. AMS Chelsea Publishing Series. American Mathematical Society, 2007.

[39] Markus Grassl et al. "Cryptanalysis of the Tillich–Zémor Hash Function". In: *Journal of Cryptology* 24.1 (Jan. 2011), pp. 148–156. ISSN: 1432-1378. DOI: 10.1007/s00145-010-9063-0.

[40] Jonathan Gryak and Delaram Kahrobaei. "The status of polycyclic group-based cryptography: A survey and open problems". In: *Groups Complexity Cryptology* 8.2 (2016), pp. 171–186. DOI: 10.1515/gcc-2016-0013.

[41] Lize Gu and Shihui Zheng. "Conjugacy systems based on nonabelian factorization problems and their applications in cryptography". In: *Journal of Applied Mathematics* 2014 (2014), 630607:1–630607:10. DOI: 10.1155/2014/630607.

[42] Guangguo Han and Chuangui Ma. "A New Authentication and Signature Scheme Based on the Conjugacy Search Problem". In: *2010 Second International Conference on Networks Security, Wireless Communications and Trusted Computing*. Vol. 2. 2010, pp. 317–320. DOI: 10.1109/NSWCTC.2010.209.

[43] Daniel Hart et al. "A Practical Cryptanalysis of WalnutDSA™". In: *Public-Key Cryptography – PKC 2018*. Ed. by Michel Abdalla and Ricardo Dahab. Cham: Springer International Publishing, 2018, pp. 381–406. DOI: 10.1007/978-3-319-76578-5_13.

[44] Marcel Herzog, Gil Kaplan, and Arieh Lev. "Representation of permutations as products of two cycles". In: *Discrete Mathematics* 285.1 (2004), pp. 323–327. ISSN: 0012-365X. DOI: 10.1016/j.disc.2004.01.019.

[45] Jeffrey Hoffstein, Jill Pipher, and Joseph H Silverman. *An introduction to mathematical cryptography*. Vol. 1. Springer, 2008.

[46] Dennis Hofheinz and Rainer Steinwandt. "A Practical Attack on Some Braid Group Based Cryptographic Primitives". In: *Public Key Cryptography — PKC 2003*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 187–198. DOI: 10.1007/3-540-36288-6_15.

[47] Derek F Holt, Bettina Eick, and Eamonn A O'Brien. *Handbook of computational group theory*. CRC Press, 2005.

[48] Ivana Ilic. "Discrete logs in arbitrary finite groups". PhD thesis. Florida Atlantic University, 2008. DOI: 10.6084/m9.figshare.5779662.v1.

[49] Delaram Kahrobaei, Charalambos Koupparis, and Vladimir Shpilrain. "Public key exchange using matrices over group rings". In: *Groups Complexity Cryptology* 5.1 (2013), pp. 97–115. DOI: 10.1515/gcc-2013-0007.

[50] Sunil Kumar Kashyap, Birendra Sharma, and Amitabh Banerjee. "A Cryptosystem Based on DLP $\gamma = \alpha^a \beta^b \mod p$". In: *International Journal of Network Security* 3 (Jan. 2006), pp. 95–100. DOI: 10.6633/IJNS.200601.3(1).08.

[51] Lee C Klingler et al. "Discrete logarithms for finite groups". In: *Computing* 85.1-2 (2009), p. 3. DOI: 10.1007/s00607-009-0047-3.

[52] Ki Hyoung Ko et al. "New public-key cryptosystem using braid groups". In: *Annual International Cryptology Conference*. Springer. 2000, pp. 166–183. DOI: 10.1007/3-540-44598-6_10.

[53] Matvei Kotov, Anton Menshov, and Alexander Ushakov. "An attack on the Walnut digital signature algorithm". In: *Designs, Codes and Cryptography* 87 (Oct. 2019). DOI: 10.1007/s10623-018-0541-9.

[54] Martin Kreuzer, Alexey D Myasnikov, and Alexander Ushakov. "A linear algebra attack to group-ring-based key exchange protocols". In: *International Conference on Applied Cryptography and Network Security*. Springer. 2014, pp. 37–43. DOI: 10.1007/978-3-319-07500-0_3.

[55] A. K. Lenstra, H. W. Lenstra Jr, and L. Lovász. "Factoring polynomials with rational coefficients". In: *Mathematische Annalen* 261.4 (1982), pp. 515–534. DOI: 10.1007/BF01457454.

[56] Juan Antonio López-Ramos et al. "An application of group theory in confidential network communications". In: *Mathematical Methods in the Applied Sciences* 41 (2016), pp. 2294–2298. DOI: 10.1002/mma.3797.

[57] Jeremy Macdonald et al. *Logspace and compressed-word computations in nilpotent groups*. 2021. arXiv: 1503.03888 [math.GR].

[58] Gérard Maze, Chris Monico, and Joachim Rosenthal. "Public Key Cryptography based on Semigroup Actions". In: *Advances in Mathematics of Communications* 1.4 (2007), pp. 489–507. DOI: 10.3934/amc.2007.1.489.

[59] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of applied cryptography*. CRC Press Series on Discrete Mathematics and its Applications. Boca Raton, FL: CRC Press, 1997, pp. xxviii+780.

[60] Alfred Menezes and Yihong Wu. "The Discrete Logarithm Problem in $GL(n, q)$". In: *Ars Comb.* 47 (1997). DOI: 10.1016/S0022-0000(76)80043-8.

[61] Chandrashekhar Meshram. "A Cryptosystem based on Double Generalized Discrete Logarithm Problem". In: *Int. J. Contemp. Math. Sciences* 6 (Jan. 2011), pp. 285–297. DOI: 10.1007/s11786-013-0166-3.

[62] Jill P. Mesirov and Melvin M. Sweet. "Continued fraction expansions of rational expressions with irreducible denominators in characteristic 2". In: *Journal of Number Theory* 27 (1987), pp. 144–148. DOI: 10.1016/0022-314X(87)90097-2.

[63] G. L. Miller. "Riemann's hypothesis and tests for primality". In: *J. Comput. System Sci.* 13.3 (1976), pp. 300–317. DOI: 10.1016/S0022-0000(76)80043-8.

[64] C. Monico. "Semirings and Semigroup Actions in Public-Key Cryptography". PhD thesis. University of Notre Dame, May 2002. DOI: 10.13140/RG.2.1.2700.0247.

[65] Alexei Myasnikov and Vitaliĭ Roman'kov. "A linear decomposition attack". In: *Groups Complexity Cryptology* 7.1 (2015), pp. 81–94. DOI: 10.1515/gcc-2015-0007.

[66] Alexei Myasnikov, Vladimir Shpilrain, and Alexander Ushakov. *Group-based cryptography*. Springer Science & Business Media, 2008. DOI: 10.1007/978-0-387-77994-4.

[67] Alexei Myasnikov, Vladimir Shpilrain, and Alexander Ushakov. "Random Subgroups of Braid Groups: An Approach to Cryptanalysis of a Braid Group Based Cryptographic Protocol". In: *Public Key Cryptography - PKC 2006*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 302–314. DOI: 10.1007/11693383_20.

[68] Alexey D. Myasnikov and Alexander Ushakov. "Quantum algorithm for discrete logarithm problem for matrices over finite group rings". In: *Groups Complexity Cryptology* 6.1 (2014), pp. 31–36. DOI: 10.1515/gcc-2014-0003.

[69] National Institute of Standards and Technology (NIST). *Announcing Approval of Federal Information Processing Standard (FIPS) 180-2, Secure Hash Standard (SHS)*. Federal Register. Accessed: 02/04/2023. Aug. 2002.

[70] National Institute of Standards and Technology (NIST). *Recommendation for Applications Using Approved Hash Algorithms*. Tech. rep. NIST Special Publication 800-107 Revision 1. Accessed: 02/04/2023. U.S. Department of Commerce, National Institute of Standards and Technology, 2012.

[71] National Institute of Standards and Technology (NIST). *Secure Hash Standard (SHS)*. Tech. rep. FIPS 180-4. Accessed: 02/04/2023. U.S. Department of Commerce, National Institute of Standards and Technology, 2015.

[72] National Institute of Standards and Technology (NIST). *Secure Hashing*. Archived Web Page. Accessed: 02/04/2023. 2011.

[73] Wouter Penard and Tim van Werkhoven. *On the Secure Hash Algorithm family*. 2008. DOI: 10.6028/NIST.IR.7355.

[74] Christophe Petit. "Towards factoring in $SL_2(\mathbb{F}_p)$". In: *Designs, Codes and Cryptography* 71.3 (June 2014), pp. 409–431. ISSN: 1573-7586. DOI: 10.1007/s10623-012-9743-x.

[75] Christophe Petit, Kristin Lauter, and Jean-Jacques Quisquater. "Full Cryptanalysis of LPS and Morgenstern Hash Functions". In: *Security and Cryptography for Networks*. Ed. by Rafail Ostrovsky, Roberto De Prisco, and Ivan Visconti. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 263–277. DOI: 10.1007/978-3-540-85855-1_18.

[76] Christophe Petit and Jean-Jacques Quisquater. "Preimages for the Tillich-Zémor Hash Function". In: *Selected Areas in Cryptography*. Ed. by Alex Biryukov, Guang Gong, and Douglas R. Stinson. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 282–301. DOI: 10.1007/978-3-642-42045-0_16.

[77] Christophe Petit and Jean-Jacques Quisquater. "Rubik's for cryptographers". In: *IACR Cryptology ePrint Archive* 2011 (Jan. 2011), p. 638.

[78] Christophe Petit, Nicolas Veyrat-Charvillon, and Jean-Jacques Quisquater. "Efficiency and pseudo-randomness of a variant of Zémor-Tillich hash function". In: *2008 15th IEEE International Conference on Electronics, Circuits and Systems*. 2008, pp. 906–909. DOI: 10.1109/ICECS.2008.4675001.

[79] Christophe Petit et al. "Hard and Easy Components of Collision Search in the Zémor-Tillich Hash Function: New Attacks and Reduced Variants with Equivalent Security". In: *Topics in Cryptology – CT-RSA 2009*. Ed. by Marc Fischlin. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 182–194. DOI: 10.1007/978-3-642-00862-7_12.

[80] S. Pohlig and M. Hellman. "An improved algorithm for computing logarithms over GF(p) and its cryptographic significance (Corresp.)" In: *IEEE Transactions on information Theory* 24.1 (1978), pp. 106–110. DOI: 10.1109/TIT.1978.1055818.

[81] J. M. Pollard. "Monte Carlo methods for index computation". In: *Mathematics of computation* 32.143 (1978), pp. 918–924. DOI: 10.1090/S0025-5718-1978-0491431-3.

[82] Carl Pomerance. "Analysis and comparison of some integer factoring algorithms". In: *Computational methods in number theory, Part I* 20 (1982), pp. 89–139. DOI: 10.1017/CBO9780511609338.008.

[83] M. O. Rabin. "Probabilistic algorithm for testing primality". In: *J. Number Theory* 12.1 (1980), pp. 128–138. DOI: 10.1016/0022-314X(80)90084-0.

[84] Ron Rivest, Adi Shamir, and Leonard Adleman. "A Method for Obtaining Digital Signatures and public-key Cryptosystems". In: *Comm. ACM* 21.2 (1978), pp. 120–126. DOI: 10.1145/359340.359342.

[85] Vitaly Roman'kov. *A general encryption scheme using two-sided multiplications with its cryptanalysis*. 2017. DOI: 10.1090/conm/418/07954. arXiv: 1709.06282 [math.GR].

[86] Vitaly Roman'kov. "Two general schemes of algebraic cryptography". In: *Groups Complexity Cryptology* 10.2 (2018), pp. 83–98. DOI: 10.1515/gcc-2018-0009.

[87] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 8.6)*. `https://www.sagemath.org`. 2020.

[88] Simona Samardjiska et al. "A Reaction Attack Against Cryptosystems Based on LRPC Codes". In: *Progress in Cryptology – LATINCRYPT 2019*. Ed. by Peter Schwabe and Nicolas Th'eriault. Cham: Springer International Publishing, 2019, pp. 197–216. DOI: 10.1007/978-3-030-25643-6_11.

[89] D. Shanks. "Class number, a theory of factorization, and genera". In: *Proc. of Symp. Math. Soc., 1971*. Vol. 20. 1971, pp. 41–440. DOI: 10.1090/pspum/020/0316385.

[90] Peter W. Shor. "Algorithms for quantum computation: discrete logarithms and factoring". In: *35th Annual Symposium on Foundations of Computer Science (Santa Fe, NM, 1994)*. Los Alamitos, CA: IEEE Comput. Soc. Press, 1994, pp. 124–134.

[91] T. N. Shorey and R. Tijdeman. *Exponential Diophantine Equations*. Cambridge Tracts in Mathematics. Cambridge University Press, 1986. DOI: 10.1017/CBO9780511566042.

[92] Vladimir Shpilrain and Alexander Ushakov. *A new key exchange protocol based on the decomposition problem*. 2005. DOI: 10.1090/conm/418/07954. arXiv: 0512140 [math.GR].

[93] Chang Seng Sin and Huey Voon Chen. "Group-Based Key Exchange Protocol Based on Complete Decomposition Search Problem". In: *Information Security Practice and Experience*. Springer International Publishing, 2019. DOI: 10.1007/978-3-030-34340-8_12.

[94] Benjamin A. Smith. "Pre- and post-quantum Diffie-Hellman from groups, actions, and isogenies". In: *IACR Cryptol. ePrint Arch.* 2018 (2018), p. 882. DOI: 10.13154/tches.v2018.i3.110-145.

[95] Andrew V. Sutherland. "Order computations in generic groups". PhD thesis. Massachusetts Institute of Technolog, 2007. DOI: 10.17863/CAM.12545.

[96] Jean-Pierre Tillich and Gilles Z'emor. "Collisions for the LPS Expander Graph Hash Function". In: *Advances in Cryptology – EUROCRYPT 2008*. Ed. by Nigel Smart. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 254–269. DOI: 10.1007/978-3-540-78967-3_16.

[97] Jean-Pierre Tillich and Gilles Zémor. "Group-theoretic hash functions". In: *Algebraic Coding*. 1993. DOI: 10.1007/3-540-57332-1_20.

[98] Jean-Pierre Tillich and Gilles Zémor. *Hashing with SL_2*. Tech. rep. France: Ecole Nationale Superieure des Telecommunications, 1994, pp. 40–49. DOI: 10.1007/3-540-48658-5_5.

[99] Simran Tinani. *Cryptanalysis of a System based on Twisted Dihedral Group Algebras*. 2022. arXiv: 2207.10979 [cs.CR].

[100] Simran Tinani. *On the Conjugacy Search Problem in Extraspecial p-Groups*. 2022. arXiv: 2203.03526 [cs.CR].

[101] Simran Tinani, Carlo Matteotti, and Joachim Rosenthal. *Complexity of Conjugacy Search in some Polycyclic and Matrix Groups*. 2022. arXiv: 2203.03525 [cs.CR].

[102] Simran Tinani and Joachim Rosenthal. "A deterministic algorithm for the discrete logarithm problem in a semigroup". In: *Journal of Mathematical Cryptology* 16.1 (2022), pp. 141–155. DOI: doi:10.1515/jmc-2021-0022.

[103] Boaz Tsaban. "Polynomial-time solutions of computational problems in noncommutative-algebraic cryptography". In: *Journal of Cryptology* 28.3 (2015), pp. 601–622. DOI: 10.1007/s00145-013-9165-4.

[104] Nikolaos Tsopanidis. "The Hurwitz and Lipschitz Integers and Some Applications". PhD thesis. Universidade do Porto, 2020. DOI: 10.13140/RG.2.2.22348.08326.

[105] Maheswara Rao Valluri and Shailendra Vikash Narayan. "Quaternion public key cryptosystems". In: *2016 World Congress on Industrial Control Systems Security (WCICSS)*. 2016, pp. 1–4. DOI: 10.1109/WCICSS.2016.7882612.

[106] Michael R. Vaughan-Lee. "Collection from the Left". In: *J. Symb. Comput.* 9 (1990), pp. 725–733. DOI: 10.1016/S0747-7171(08)80054-5.

[107] James B. Wilson. "Finding central decompositions of p-groups". In: *Journal of Group Theory* 12.6 (2009), pp. 813–830. DOI: 10.1515/JGT.2009.015.

[108] Gilles Zémor. “Hash Functions And Graphs With Large Girths”. In: *International Conference on the Theory and Application of Cryptographic Techniques*. 1991. DOI: 10.1007/3-540-46416-6_44.

[109] Gilles Zémor. “Hash functions and graphs with large girths”. In: *Workshop on the Theory and Application of Cryptographic Techniques*. Springer. 1991, pp. 508–511. DOI: 10.1007/3-540-46416-6_44.

[110] J. Zumbrägel. “Classification of Finite Congruence-Simple Semirings with Zero”. In: *Journal of Algebra and Its Applications* 07.03 (2008), pp. 363–377. DOI: 10.1142/S0219498808002862.