# ADVANCING AUTONOMY AND SECURITY IN FUTURE COMPUTING SYSTEMS WITH BLOCKCHAIN AND SUPERSINGULAR ISOGENY

by

Miraz Uz Zaman, B.S., M.S.

A Dissertation Presented in Partial Fulfillment
of the Requirements of the Degree
Doctor of Philosophy

COLLEGE OF ENGINEERING AND SCIENCE
LOUISIANA TECH UNIVERSITY

August 2023

LOUISIANA TECH UNIVERSITY

**GRADUATE SCHOOL**

**May 17, 2023**
Date of dissertation defense

We hereby recommend that the dissertation prepared by

**Miraz Uz Zaman, B.S., M.S.**

entitled **Advancing Autonomy and Security in Future Computing Systems with Blockchain and Supersingular Isogeny**

be accepted in partial fulfillment of the requirements for the degree of

**Doctor of Philosophy in Computational Analysis & Modeling**

Manki Min
Supervisor of Dissertation Research

Weizhong Dai
Head of Computational Analysis & Modeling

**Doctoral Committee Members:**
Manki Min
Galen Turner
Pradeep Chowriappa
Andrey Timofeyev
Ibrahim Abdoulahi

**Approved:**

Hisham Hegab
Dean of Engineering & Science

**Approved**:

Ramu Ramachandran
Dean of the Graduate School

# ABSTRACT

The exponential growth of data and increasing connectivity necessitate the evolution of computing systems to handle vast amounts of data and support real-time processing. This evolution is further driven by emergence of technologies like AI, machine learning, Blockchain, and quantum computing which shapes the future of computing systems. Future computing systems are expected to be more intelligent, interconnected, automated, secure, and efficient, enabling advanced applications across various industries. To fully harness the opportunities presented by these future computing systems, it is essential to develop tools and mechanisms that enhance autonomy and security. This dissertation focuses on two such tools: the Blockchain consensus mechanism and the supersingular isogeny-based hash function, which have the potential to significantly enhance autonomy and security in future computing systems.

In the study of Blockchain consensus mechanisms, two distinct methods have been proposed. The first method is called "proof of sincerity," which is designed to be mobile-friendly and fairer in rewarding. It rewards all miners, regardless of their computing power, based on their sincerity level, which is a measure of their contribution to the network. This approach reduces resource waste and enhances security compared to other methods like proof of work.

The second method is designed for Blockchain-based data storage, which offers many advantages over third-party cloud storage, such as data security, integrity, and reliability. This proposed consensus mechanism allows most devices or entities to participate by performing low computational validation work. The proposed consensus mechanism is analyzed using game theory and queuing theory to further validate its effectiveness.

The study on supersingular isogeny hash includes the presentation of different compact implementations of the CGL function, which is based on the traversal in a supersingular isogeny graph and was proposed by Charles, Goren, and Lauter. The compact implementations utilize various forms of elliptic curves, such as Weierstrass, Montgomery, and Legendre. Furthermore, the study compares the running time and the total number of collisions observed in experiments conducted with the implemented algorithms.

Additionally, a novel single compression cryptographic hash function is proposed, which is based on the traversal in the supersingular isogeny graph using the 2-isogeny and point mapping under the isogeny. Unlike existing supersingular isogeny-based hash functions, the proposed function outputs the $X$-abscissa of a point on a supersingular elliptic curve without revealing the $j$-invariant of the traversed curve. The computational complexity of the proposed hash function is analyzed and compared to the CGL and CGL-like hash functions.

The combination of the Blockchain consensus mechanism and Supersingular Isogeny Hash offers a compelling path towards advancing the autonomy, security, and post-quantum capabilities of Blockchain networks. By harnessing their collective

strengths, new frontiers can be unlocked in secure and decentralized computing, revolutionizing industries and shaping the future of trusted digital transactions.

# APPROVAL FOR SCHOLARLY DISSEMINATION

The author grants to the Prescott Memorial Library of Louisiana Tech University the right to reproduce, by appropriate methods, upon request, any or all portions of this Dissertation. It is understood that "proper request" consists of the agreement, on the part of the requesting party, that said reproduction is for his personal use and that subsequent reproduction will not occur without written approval of the author of this Dissertation. Further, any portions of the Dissertation used in books, papers, and other works must be appropriately referenced to this Dissertation.

Finally, the author of this Dissertation reserves the right to publish freely, in the literature, at any time, any or all portions of this Dissertation.

Author _____

Date _____

# DEDICATION

I dedicate this work wholeheartedly to my parents, who have been the cornerstone of my journey with their unwavering love, support, and sacrifices. Their belief in me and constant encouragement have propelled me forward.

I also dedicate this work to my beloved beautiful wife, who has been my guiding light with her understanding, patience, and unwavering faith in my abilities. She has stood by my side throughout this endeavor with endless love and support.

I dedicate this work to my dear son, who is the driving force behind my pursuit of knowledge and my greatest inspiration. I hope that this work sets an example of perseverance and the pursuit of dreams for him.

Lastly, I would like to thank my wonderful sister for her invaluable belief in me and constant encouragement.

Thank you all for being my pillars of strength.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# ACKNOWLEDGMENTS

I extend my deepest gratitude and appreciation to my esteemed advisor, Dr. Manki Min, for his unwavering support and guidance throughout my entire academic journey in Louisiana Tech. Dr. Min's exceptional mentorship has not only helped me refine my thought process but has also provided invaluable insights that aided me in making crucial decisions, both academically and personally. His continuous availability and willingness to assist whenever I needed help have been indispensable in my pursuit of a doctorate. I consider myself extremely fortunate to have had Dr. Min as my advisor, as his constant support has played a pivotal role in the successful completion of my work.

I would also like to express my sincere thanks to Dr. Aaron Hutchinson for generously dedicating his valuable time to address all the inquiries I had regarding supersingular isogeny. His expertise and insights have made a significant impact on the development of my research.

Furthermore, I am deeply grateful to the esteemed members of my dissertation advisory committee, Dr. Galen Turner, Dr. Pradeep Chowriappa, Dr. Andrey Timofeyev, and Dr. Ibrahim Abdoulahi, for their valuable feedback, constructive suggestions, and insightful criticism. Their contributions have played a vital role in shaping the quality and direction of my work.

In addition, I would like to acknowledge and express my heartfelt gratitude to Dr. Collin Wick and Dr. B. Ramu Ramachandran for their exceptional leadership in the graduate school. Their support has been instrumental in overcoming significant financial challenges that I would have otherwise faced in completing my degree. I am truly grateful for their generosity, which has enabled me to reach this significant milestone in my academic journey.

Lastly, I would like to extend my appreciation to the Louisiana Board of Regents for their support of my work through grant GR301278.

To all those mentioned above, as well as to anyone else who has contributed to my academic and personal growth, I offer my deepest appreciation and thanks. Your unwavering support and belief in my abilities have been invaluable, and I am truly grateful for the role each of you has played in my success.

# CHAPTER 1

# INTRODUCTION

## 1.1 Overview

We live in an era of ubiquitous and interconnected computing systems. The system comprises smartphones, laptops, smart watches, smart home appliances, and many more devices. The internet and the development of technologies have facilitated unparalleled connectivity and allowed us to seamlessly exchange information and collaborate in ways that were once impossible. As technology is evolving rapidly, users' expectations and demands are growing. Moreover, with every advancement, new possibilities emerge, expanding the potential for innovation and growth. In this rapidly shifting paradigm of computing systems, the future computing system needs to be distributed, decentralized, and secure to keep it reliable.

Autonomous systems are self-governing networks that lack centralized control, with each participant having a designated role to play. Compared to centralized networks, autonomous systems are more secure, resilient, and democratic since they lack a single point of failure, and all participants can influence network governance. In autonomous systems, data security and transparency are critical aspects. The decentralized and immutable nature of Blockchain, coupled with its smart contract automation capabilities, makes it a powerful tool for enhancing autonomy and security.

By implementing Blockchain in computing systems, they can attain self-governance, self-configuration, self-optimization, and self-protection [1]. For instance, Blockchain enables computing systems to manage resources independently, verify transactions, optimize performance, and defend against attacks [2]. Blockchain also empowers users and devices to interact directly with each other, eliminating the need for intermediaries and third-party services or platforms. This enables peer-to-peer communication, collaboration, coordination, user-centric data ownership, and control [3]. Devices and users can share spectrum, exchange data, provide services, and make decisions leveraging the capabilities of Blockchain [2]. Moreover, the immutability nature of Blockchain ensures that data in the chain cannot be altered or erased, making the system transparent. A mature Blockchain technology that possesses all the qualities of an autonomous system is essential for the future decentralized web (Web 3.0).

In a Blockchain, a consensus mechanism is the foundation of the network. However, many existing consensus algorithms tend to become more centralized as the network grows, giving more power and influence to a few users. This undermines the potential of Blockchain for various applications that require decentralization and trustlessness. Hence, there is a demand for a new consensus mechanism that can maintain its decentralization and scalability as the network expands, enhancing the opportunity for different applications of Blockchain.

However, the significant opportunity in the computing system also comes with significant risks, such as cybersecurity threats, privacy violations, system failures, data breaches, and other threats. These traditional threats are already a significant challenge, but the looming threat of quantum computing attacks compounds them.

These attacks rely on the principle of implementing Shor's algorithm [4], which can efficiently break many of the public-key cryptography algorithms that underpin currently used secure computing systems [5]. The principal idea of the quantum computer is to leverage the quantum mechanical phenomena to find high-quality solutions to problems considered hard or impossible to solve, even with the largest supercomputer available. Thus, there is a growing need for innovative solutions to address these threats and facilitate the creation of more autonomous and secure computing systems.

According to a report by the US National Academy [6], we are roughly a few decades away from a fully-fledged quantum computer, which poses a significant threat to the existing computing system. To counter the looming danger of quantum computing attacks, a field called post-quantum cryptography (PQC) has emerged, which can withstand the quantum attack and seamlessly integrate with the current system. In 2016, the National Institute of Standards and Technology (NIST) initiated a project to develop new standards and protocols for post-quantum public key cryptography. After several rounds of evaluation, four algorithms have been chosen for standardization, while four more algorithms headed to the fourth round as alternative candidates for the analysis. SIKE [7] an isogeny-based key encapsulation scheme is an alternative algorithm that builds upon Supersingular isogeny Diffie-Hellman key exchange or SIDH [8]. SIKE is recently broken by different researchers [9–11] separately exploiting the information exchanged in protocol. Nonetheless, isogeny-based cryptographic schemes that do not rely on the exploited information are still considered safe from such attacks. Isogeny-based cryptographic schemes are gaining

attention from researchers due to their low key size and the computational hardness of isogeny calculations. Given that a cryptographic hash function is a fundamental tool in any cryptographic system, the inclusion of an efficient isogeny-based hash function would be a valuable addition to the suite of post-quantum cryptographic tools.

## 1.2 Motivation

In a Blockchain system, the consensus mechanism is the process by which all stakeholders in the network agree on the state of the network and the validity of transactions. A consensus mechanism should be designed to accumulate agreement from all the stakeholders so that the group interest is prioritized over any individual's interest and ensure equal weighting. It should also allow new stakeholders to join the network efficiently. Different consensus mechanisms are used in the Blockchain network depending on their use case and the requirements of the network. Proof of work (PoW) [12] is one of the well-known and most used consensus mechanisms based on the idea of solving complex mathematical problems to add the transaction to the Blockchain. Bitcoin and most cryptocurrencies are based on the PoW or PoW alike consensus mechanism. However, PoW and other variants are resource intensive and can lead to monopolization or centralization by giving more power to stakeholders with higher computing resources. Hence, there is a demand for consensus mechanisms that can enable smaller stakeholders to participate and contribute equally to the network.

While the initial application of Blockchain technology was primarily related to cryptocurrencies, its evolved features have the potential to impact numerous industries. For example, Blockchain can be used for healthcare, digital voting, decentralized

governance, and data sharing. In today's world, interconnected computing systems produce a vast amount of data, primarily stored, processed, and distributed using cloud-based storage. However, most cloud-based storage is centrally controlled and backed by a handful of technology companies with incredible data storage, which poses significant challenges such as data breaches, server malfunctions, high storage costs, unavailability of storage servers, and lack of data validity. Decentralized Blockchain-based data storage can address this challenge due to its inherent characteristics. Nonetheless, Blockchain-based storage also faces some challenges, such as scalability, network speed, and latency. These challenges can be overcome by carefully assigning different roles to users based on their capabilities and by optimizing the trade-off between the inherent features of Blockchain and its limitations. A Blockchain-based data storage platform that provides an optimized balance between the advantages and limitations of the technology could be a vital component in an autonomous system.

In order to ensure the security of computing systems, a vital component is a cryptographic hash function. A cryptographic hash function has many uses in cryptography, either as a standalone tool or as a component of other schemes. For example, Blockchain storage uses hash functions to generate unique identifiers for each data block and to validate transactions on the network. Some applications of cryptographic hash functions are message authentication code (MAC), password verification, signature generation, and verification. A cryptographic hash function should be efficient and secure. This means it should be difficult to find two inputs with the same output (collision-free) or to find an input that matches a given output (preimage resistant). Additionally, the output of the hash function should be evenly

distributed. However, the security of conventional cryptographic hash functions may not hold against quantum adversaries. Some researchers have argued that quantum attacks can compromise the collision and preimage resistance of some hash functions [13], while others have claimed that quantum attacks are not feasible in practice [14].

To address the quantum threat, Charles, Goren, and Lauter proposed a quantum-resistant hash function based on supersingular isogeny graphs (expander graph) ter in 2009 [15] known as CGL hash. This hash function uses a pseudo-random walk on the graph starting from a fixed supersingular elliptic curve over $\mathbb{F}_{p^2}$ and computes 2-isogenies along the way. The input bit string determines the direction of each step on the graph. The security of this hash function depends on the difficulty of finding an isogeny between two given supersingular curves, which is exponentially hard for quantum algorithms [8, §5]. However, if the endomorphism ring of the starting curve is known [16] or easy to compute [17], then the hash function can be broken. When the endomorphism ring of the initial curve is unknown, CGL security still relies on the difficulty of finding an isogeny between two supersingular elliptic curves. Additionally, the original proposal of the CGL hash function includes some redundant computations that can be minimized by utilizing unique non-trivial characteristics of elliptic curve computations. Supersingular isogeny hash functions are promising candidates for post-quantum cryptography if their efficiency can be improved.

### 1.3 Organization of the Dissertation

The dissertation is structured as follows:

Chapter 2 provides an overview of widely used consensus methods in Blockchain, along with an exploration of the fundamental concepts of supersingular isogeny and the details of the CGL hash function.

Chapter 3 discussed two proposed mobile-friendly Blockchain consensus mechanisms in detail. The first mechanism, Proof of Sincerity, facilitates easier mining and rewards for all sincere miners. The second mechanism focuses on a Blockchain-based storage system.

Chapter 4 demonstrates efficient implementations of the CGL hash function using different elliptic curves. Additionally, it proposes a supersingular isogeny-based single compression cryptographic hash function.

Finally, Chapter 5 concludes the dissertation, offering remarks on the findings and suggesting future research directions based on the study's outcomes.

# CHAPTER 2

# BACKGROUND

This chapter provides a review of some fundamental concepts that are relevant to this thesis. The first section 2.1 presents some common methods for achieving proof of consensus and some Blockchain-based storage architectures. The second section 2.2, covers some essential aspects of supersingular elliptic curves and their isogenies, which are the foundation for a category of post-quantum cryptographic schemes. The section will conclude with a brief overview of the CGL hash function, as well as other hash functions based on Supersingular Isogeny.

## 2.1 Blockchain

### 2.1.1 Proof of Consensus

Blockchain is a distributed and decentralized database. As the name suggests, Blockchain is a chain of blocks where each block stores the data of transactions, timestamps, and a hash of the previous blocks. Here hash links the previous block to the current block and creates a chronological event of data. Now to add a block in the Blockchain, the participants in the network need to agree upon a block. The process for agreement among the participants for a block is called the consensus mechanism. Some of the widely known proof of the consensus mechanism of Blockchain will be discussed here.

### Proof of Work

Bitcoin is the first cryptocurrency that uses the Proof of Work (PoW) consensus mechanism [12], which combines the algorithm proposed by Dwork and Naor [18] with cryptographic signatures, Merkle chains, and P2P networks. The process to add the block in the Bitcoin Blockchain is called *mining*, and the entity responsible for adding the block is called *miner*. To add a new block, miners must solve a time-consuming mathematical puzzle to find an integer or random data point (nonce) that produces an output with a predetermined number of leading zeros through SHA-256 (Secure Hash Algorithm) [19]. The hash output must have a number of leading zeros equal to or lower than a predefined *target*. The first miner who finds a valid nonce gets a fixed amount of bitcoins as a reward, which decreases by half every 210,000 blocks [20], or roughly every four years at the current rate (1800 bitcoins/day). The PoW consensus mechanism is a huge energy consumer, and its electricity usage matches that of the whole country of Ireland [21]. Moreover, the mining technology has evolved from CPU to GPU (Graphics processing unit), GPU to FPGA (Field-programmable gate array), and FPGA to ASIC (Application-specific integrated circuit) [22], making it more expensive and less accessible for ordinary users. The average time for a new block to be added to the Bitcoin Blockchain is ten minutes [23]. A significant drawback of the PoW consensus mechanism is the possibility of a 51% attack [24], where a miner or a group of miners who control more than half of the network's computing power can invalidate any valid transaction.

### Proof of Stake

In 2012, an alternative consensus protocol called Proof of Stake (PoS) was proposed in [25] as a solution to the energy consumption problem of PoW. Instead of energy-intensive competition, PoS selects the next block miner based on a combination of random selection and stake (wealth or age). This means there is no need for miners to compete for the right to mine blocks, which can be less energy-intensive than PoW. A well-known cryptocurrency that uses a hybrid of PoW and PoS is Ethereum, which plans to transition to PoS entirely in the future. However, PoS has some drawbacks, such as requiring a minimum stake to participate as a validator, which may exclude some potential validators and lead to the centralization of power among the wealthy. Additionally, PoS is vulnerable to the double-spending problem due to a nothing-at-stake [26] attack, in which a dishonest miner works on multiple forks simultaneously without risking their stake, enabling attackers to invalidate their spending. Several variations of PoS have been proposed to address some of the limitations of the original protocol. One of the most popular variations is delegated PoS [27], in which token holders elect validators to participate in block validation. Another variation is bonded PoS, in which validators must stake a certain amount of cryptocurrency to participate in block validation.

### Proof of Activity

Proof of activity (PoA) [28] is a hybrid consensus mechanism that combines PoW and PoS, introduced as part of peercoin cryptocurrency. In PoA, miners compete to solve cryptographic puzzles like PoW to mine blocks. Following this, miners are

required to demonstrate ownership of a specified amount of coins to transition into the PoS stage. In PoS, the new validator is chosen based on the coin the user owns. PoA offers better energy efficiency and decentralization than PoW and PoS, respectively. However, it also poses challenges in terms of complexity and maintenance.

**Proof of Space**

Proof of space (PoSpace) was introduced in 2013 as an alternative to PoW and PoS, and later in 2018, it was incorporated into the Chia cryptocurrency. In PoSpace, nodes must demonstrate that they have allocated some storage space to participate in the block validation process. The miner invests in hard drive space, and their likelihood of being the next block miner is proportional to the amount of storage space they have allocated. Other algorithms, such as *Proof of Capacity* and *Proof of Storage*, have been developed based on this approach. While PoSpace offers superior energy efficiency, it also has some drawbacks, such as high initial storage requirements, verification complexity, and the potential for centralization.

**Proof of Time**

A consensus mechanism based on the passage of time is called proof of time (PoT). In PoT, the nodes do not need to use a lot of computing power or storage space, but they need to spend time on the network. To do this, they use a verifiable delay function (VDF), which is a type of cryptographic function that takes a fixed amount of time to calculate. Nodes that spend more time online are more likely to compute the VDF and participate in block validation. PoT is more efficient in terms

of energy and space than other consensus mechanisms, but its security level is still unclear to researchers.

### 2.1.2 Decentralized Storage with Blockchain

#### STORJ

Many Blockchain-based file-sharing systems use BitTorrent [30], a well-known peer-to-peer protocol, as their model. STORJ [31] is another open-source and decentralized Blockchain-based storage platform that resembles BitTorrent. Like BitTorrent, STORJ divides the file into several chunks through file sharding and distributes them across the STORJ network. However, STORJ differs from BitTorrent in using a distributed hash table to locate the file chunks so that only the file owner knows their locations. The core component of the STORJ network is Kademlia [32], a distributed hash table. Additionally, the file owner has the option to choose different levels of redundancy to increase the file's availability on the network.

#### Sia

Sia [33] is a decentralized storage platform that leverages Blockchain technology for data storage. It was proposed in 2013 during HackMIT, a prestigious annual hackathon hosted by MIT. On Sia, storage providers, and clients negotiate contracts for storing and paying for data. The contracts also specify the size and duration of the data, the frequency and reward of proofs, and the penalty for missing proofs. The storage provider must prove that the data is still intact and stored using the consensus mechanism, Proof of Storage [34]. This consensus mechanism verifies the availability and integrity of data. Contracts are successfully terminated when the storage duration

ends or unsuccessfully terminated when the maximum number of missed proofs is exceeded. These contracts are stored on the Blockchain network, ensuring they are publicly auditable, immutable, and decentralized. Sia claims to offer cloud storage that is 90% cheaper than traditional providers like Amazon S3.

**IPFS**

Protocol Labs has developed Interplanetary File System (IPFS) [35], an open-source, decentralized file-sharing technology considered a successor to modern internet architecture. The addressing format used in IPFS is different from the traditional location addressing used in HTTP. IPFS uses content addressing, meaning each file is referred to by a unique address based on the hash of the file's content. This allows for faster retrieval, deduplication, and caching of data. Each IPFS object contains two fields: unstructured binary data less than 256 kB and links to other IPFS objects. The link structure consists of three data fields: the name of the link, the hash of the linked IPFS object, and the cumulative size of the linked IPFS object. The link list remains empty if the data block is smaller than 256 kB. IPFS does not have a built-in mechanism for versioning files, so users must rely on external tools or protocols to track changes in their data. To incentivize users to contribute storage and bandwidth to the network, a cryptocurrency called FileCoin [36] has been developed on top of IPFS. FileCoin rewards users for hosting and retrieving files on IPFS.

## 2.2 Essential Concepts of Supersingular Isogenies

### 2.2.1 Elliptic Curve over Finite Field

A finite field $\mathbb{F}_q$ is a field with $q$ elements, where $q = p^k$ and $p$ is a prime. An elliptic curve $E$ is a non-singular projective curve of genus 1 defined over a finite field $\mathbb{F}_q$. The equation of such a curve in affine coordinates can be written as

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6,$$

where the coefficients $a_1, a_2, a_3, a_4, a_6 \in \mathbb{F}_q$. Other than the Weierstrass form, elliptic curves can be represented in other well-known forms [37, §II.B], such as the Montgomery form, Legendre form, Edward curves, etc., each of which shows a wide variety of computational costs for arithmetic operations and isogeny computations in the prime field.

The set of points on an elliptic curve over $\mathbb{F}_q$, together with a special point called the point at infinity $O_E$, forms an abelian group $E(\mathbb{F}_q)$ under a geometric operation called point addition. The order of a point $P = (x_p, y_p) \in E(\mathbb{F}_q)$ is the smallest positive integer $n$ such that $nP = O_E$, where $nP$ denotes adding $P$ to itself $n$ times. The size of the group $E(\mathbb{F}_q)$, denoted by $\#E(\mathbb{F}_q)$, is approximately equal to $q$, as stated by Hasse's theorem [38, §V.1]. More precisely, cardinality of $E(\mathbb{F}_q)$ satisfies $||E(\mathbb{F}_q)| - (q+1)| \leq t$, where $t$ is the trace of Frobenius and satisfies $|t| \leq 2\sqrt{q}$. An elliptic curve over $\mathbb{F}_q$ can be classified as either supersingular or ordinary depending on the value of $t$. If $t$ is divisible by the characteristic $p$, then the curve is supersingular; otherwise, it is ordinary. Another way to distinguish between supersingular and ordinary curves is by looking at their $p$-torsion subgroups, which are the subsets of

points whose order is a power of $p$. A supersingular curve has a trivial $p$-torsion subgroup, i.e., $E[p] = 0$, while an ordinary curve has a non-trivial one [38, §V.3]. Post-quantum security researchers focus more on supersingular elliptic curves than ordinary ones since a sub-exponential-time quantum algorithm for solving the discrete logarithm problem on ordinary elliptic curves was discovered by Childs, Jao, and Soukharev in [39].

### 2.2.2 Isogeny

Suppose $f : E \rightarrow E'$ is a function that maps points on one elliptic curve $E$ to another elliptic curve $E'$ in a way that preserves their group operation, i.e., $f(O_E) = f(O_{E'})$ where $O_E$ and $O_{E'}$ are the identity elements of $E$ and $E'$ respectively. This function $f$ is called a *morphism* [40, §3]. Two elliptic curves $E$ and $E'$ over $\mathbb{F}_q$ are *isomorphic* if they have the same shape, which is measured by their $j$-invariant, i.e., $j(E) = j(E')$. Now, a surjective and non-constant morphism is called an *isogeny*. If an isogeny $\phi : E \rightarrow \tilde{E}$ exists, then $\phi(O_E) = O_{\tilde{E}}$, and $\phi$ induces a group homomorphism between $E$ and $\tilde{E}$. The two elliptic curves are said to be isogenous to each other if an isogeny exists between them. Each isogeny $\phi$ has a unique dual isogeny $\hat{\phi} : \tilde{E}/\mathbb{F}_q \rightarrow E/\mathbb{F}_q$, such that $\hat{\phi} \circ \phi = [deg\phi]$ and $\phi \circ \hat{\phi} = [deg\hat{\phi}]$, where $[n]$ denotes the multiplication-by-$n$ map on the elliptic curve [38, §III.6.1]. An equivalence relation can be defined based on the isomorphic classes of elliptic curves defined over $\mathbb{F}_q$, where the isogenous elliptic curves belong to the same isogeny class. The degree of isogeny denoted as $deg\phi$ is equal to the number of points in the kernel of isogeny, i.e., $deg\phi = |ker(\phi)|$ for a separable and non-constant isogeny. The kernel of an isogeny

$\phi$ is the list of finite subgroup order of points of $E$ over $\mathbb{F}_q$. An isogeny is typically expressed with its kernel; for example, an $\ell$-*isogeny* would have a kernel of size $\ell$. An $\ell$-isogeny, for example, would have a kernel of size $\ell$. In isogeny-based cryptography, V'elu's formula [41] is the most commonly used method to compute the isogeny, which uses the knowledge of the kernel points to compute the isogeny.

### 2.2.3 Isogeny Graph

Since all the supersingular elliptic curves over a field of characteristic $p$ belong to the same isogeny class and can be defined over either $\mathbb{F}_p$ or $\mathbb{F}_{p^2}$, the field can be chosen as $\mathbb{F}_q = \mathbb{F}_{p^2}$ [42]. The $j$-invariant is a unique element of $\mathbb{F}_{p^2}$ that characterizes the isomorphism class of an elliptic curve. An isogeny graph can be defined as a graph whose vertices are elliptic curves over $\mathbb{F}_{p^2}$ and whose edges are isogenies of a fixed degree $l$ between them. Such a graph has two components: one consisting of ordinary elliptic curves and the other of supersingular ones. The two components are disjoint because of the isogenous relation between the supersingular elliptic curves.

Pizer [43, 44] proved that the supersingular component has the property of being a Ramanujan graph [45], which is a highly connected regular graph also known as an expander graph. Here the supersingular component is the main focus since it is strongly connected and has applications in post-quantum cryptography. More details on Ramanujan graphs can be found in [45, 46]. The vertices in the isogeny graph are expressed as $j$-invariants, which can be computed directly from the curve equation. Now, in a supersingular isogeny graph, the total number of unique vertices are $\frac{p}{12} + \epsilon$ where $\epsilon \in \{0, 1, 2\}$ depends on the characteristics of the field.

Figure 2.1 [47] illustrates a 2-isogeny graph over $\mathbb{F}_{211^2}$. It has 18 vertices, corresponding to supersingular $j$-invaraints, and 2-isogenies as edges between them. Most vertices have three neighbors, except for the one with $j$-invariant 40, which has only two neighbors. However, it is widely believed that this behavior is rare, and as $p$ grows, the number of such exceptional vertices remains small.

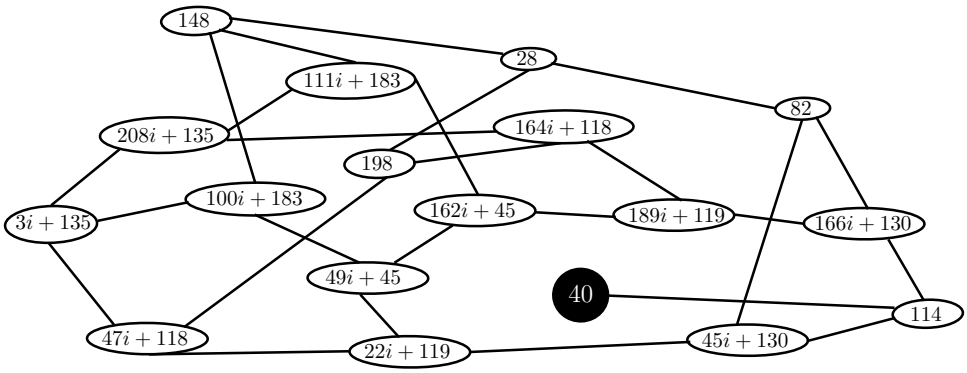



**Figure 2.1:** The 2-isogeny graph for $\mathbb{F}_{211^2}$

### 2.2.4 Supersingular Isogeny Based Hash

#### CGL

The isogeny graph possesses strong pseudo-random properties because of rapid mixing during random navigation, just like the expander graph. Cryptographic applications, such as those mentioned in [48, 49], have made use of these properties. The CGL hash function is a de-facto supersingular isogeny-based hash algorithm that exploits the pseudo-random nature of the isogeny graph to generate a collision-free hash function. The function takes a bit stream of the message and initiates a non-backtracking traversal in the 2-isogeny graph $G_2$ over $\mathbb{F}_{p^2}$ from a fixed curve (vertex). The output of the CGL hash function is the $j$-invariant of the end curve

in the traversal. Since different initial elliptic curves $E \in G_2$ give different outputs for the same input, we can get a family of hash functions indexed by $j$-invariant. Theoretically, a 2-isogeny graph $G_2$ is almost equivalent to a 3-regular graph, with each curve having three adjacent edges (isogenies) leading to neighboring curves. To avoid backtracking, one of the edges is permanently ignored, leaving only two edges to follow from each curve to the next. A consistent rule is set to assign bits 1 and 0 to these two edges, leading one curve to the next curve.

Suppose a starting vertex or supersingular elliptic curve of the form

$$E_1 : y^2 = f_1(x) = (x - x_1)(x - x_2)(x - x_3)$$

where $(x_1, x_2, x_3) \in \mathbb{F}_{p^2}$. This implies that there are three non-trivial 2 torsion points of $E_1$ that can be expressed as a set $\{(x_1, 0), (x_2, 0), (x_3, 0)\}$. Now let $E_2$ be another supersingular elliptic curve given by $y^2 = f_2(x)$, which is 2-isogenous to $E_1$, and let $\phi$ be the isogeny with kernel $\langle (x_1, 0) \rangle$. Then the images of the other two torsion points $(x_2, 0)$ and $(x_3, 0)$ under $\phi$ are the same point on $E_2$, that is, $\phi(x_2, 0) = \phi(x_3, 0)$. On the other hand, the dual isogeny $\hat{\phi}$ maps $E_2$ back to $E_1$, and its kernel is either $\langle (x_2, 0) \rangle$ or $\langle (x_3, 0) \rangle$ on $E_2$. This is what CGL calls the backtracking isogeny. To prevent backtracking, one efficient way is to factor out the new cubic function $f_2(x)$ by $(x - \hat{x}_2)$, where $\hat{x}_2$ is the $x$-coordinate of $\phi(x_2, 0)$. This results in a quadratic polynomial whose roots are the $x$-coordinates of the remaining two torsion points. The CGL hash function assigns a bit value to each torsion point according to a fixed rule. The chosen point becomes the new kernel point for the next step in the graph or the next isogenous curve. Thus, for an input string of $n$ bits, the CGL hash function

takes $n$ steps in the 2-isogeny graph. The output of the CGL hash function is the $j$-invariant of the final supersingular elliptic curve in the walk. Figure 2.2 [47] shows a flowchart of the CGL hash function.

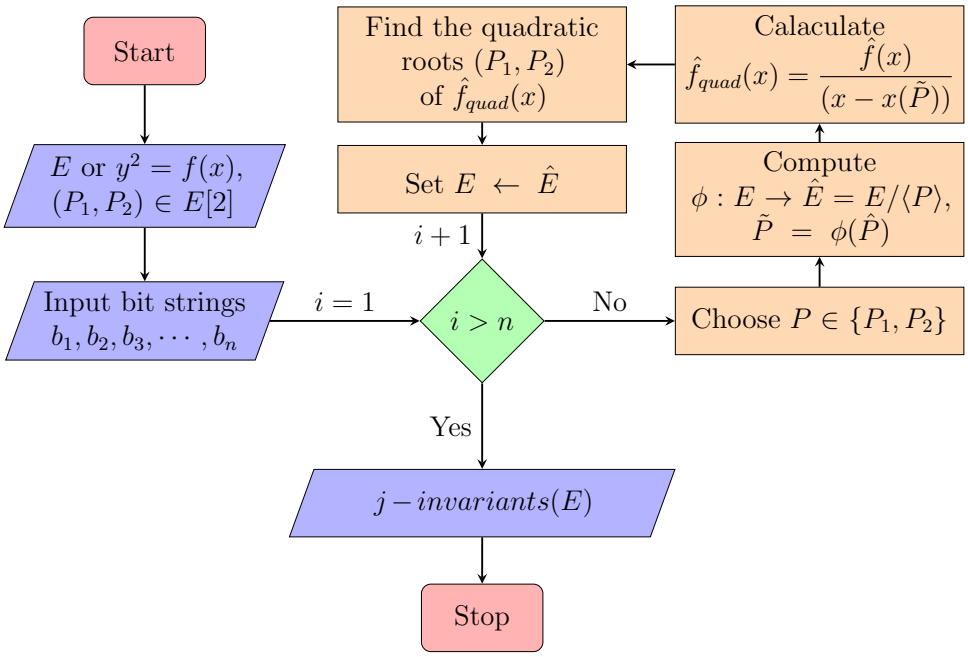



**Figure 2.2:** Flowchart of CGL Hash

Figure 2.3 depicts an example of path propagation of the CGL hash function in a supersingular 2-isogeny graph over $\mathbb{F}_{139^2}$. The graph has 12 vertices, each representing a unique isomorphic class of supersingular elliptic curves determined by their $j$-invariant. All the vertices except 60 and 100 in the graph show the expected behavior of 2-isogenies. Suppose the input bit string is "1011" and the starting vertex is $96i + 57$. Moreover, consider when the first bit is processed, the CGL function reaches vertex $103i + 123$. For the second bit, the CGL function avoid the backtracking path from the current vertex to $96i + 57$. Then, it reaches vertex $36i + 123$ by taking one of the remaining two non-backtracking edges. In a similar way, by consuming the

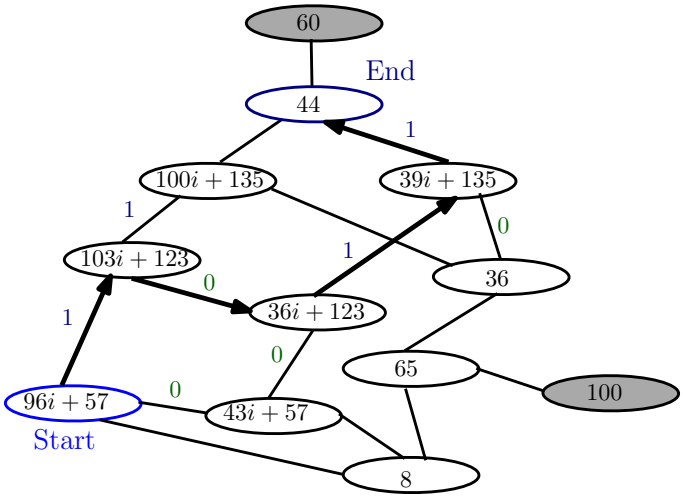



**Figure 2.3:** CGL propagation path in 2-isogeny graph of $\mathbb{F}_{139^2}$

last two bits, the CGL function finally reaches vertex 44, which is the output of the CGL hash.

**Other SI Based Hash**

Yoshida and Takashima suggested an improvement to the 2-isogeny hash function (CGL) in their paper [50]. They utilized observations on the 2-torsion point information on short Weierstrass elliptic curves to remove some redundant computations.

Doliskani, Pereira, and Barreto presented a variant of the CGL hash function in [51] that operates on field characteristics of the form $p = 2^n f \pm 1$, where $f > 0$ is a small integer. Instead of processing one bit at a time like CGL, their algorithm processes a block of length $n \approx \log p$. The authors demonstrated that their algorithm's running time is asymptotically quasi-linear in $n$, while CGL's running time is quadratic in $n$. However, it can be shown that, with an upper bound approximation, both hash algorithms yield the same running time.

Panny introduced a modified version of the SI hash function that incorporates cyclic $l$-isogeny, where $l$ varies rather than being fixed at $l = 2$ in each step [52, §2]. In a separate study, Tachibana, Takashima, and Takagi proposed another variation of the CGL function utilizing 3-isogenies [53]. This 3-isogeny-based hash function employs a straightforward representation of backtracking 3-torsion points. The input is transformed into a ternary expansion, and the mapping of $\{0, 1, 2\}$ to three non-backtracking edges of each vertex is utilized to apply 3-isogenies. Despite the different methods employed, both the computational complexities of this hash function and the CGL function remain equivalent.

# CHAPTER 3

# BLOCKCHAIN CONSENSUS AND STORAGE ARCHITECTURE

The content in this chapter has been published in the following conferences:

- M. U. Zaman, T. Shen and M. Min, "Proof of Sincerity: A New Lightweight Consensus Approach for Mobile Blockchains," 2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 2019, pp. 1-4, doi: 10.1109/CCNC.2019.8651742 [54].
- M. U. Zaman and M. Min, "Decentrally-Consented-Server-Based Blockchain System for Universal Types of Data," 2020 International Symposium on Networks, Computers and Communications (ISNCC), Montreal, QC, Canada, 2020, pp. 1-6, doi: 10.1109/ISNCC49221.2020.9297229 [55].

## 3.1 Proof of Sincerity

Bitcoin's emergence has led to the introduction of various distributed consensus methods to replace the central authority in maintaining the integrity of distributed ledgers. However, none of the existing methods have found a good balance between preventing wealth inequality and protecting the security of the ledgers. The *proof of sincerity (PoSn)* consensus method proposed in [54] addresses this imbalance by allowing even those with minimal computing power to be rewarded through the mining

process, preventing the dominance of a few large resource owners in the mining process. By appropriately setting the mining difficulty based on the number of miners and their mining capacity, PoSn can maintain the desired level of ledger security. Additionally, by changing the design of the mining task, PoSn can achieve a higher level of security than other consensus methods and reduce the unnecessary waste of resources that happens in PoW.

### 3.1.1 Unfairness in PoW

The mathematical puzzle in PoW is based on finding a hash value with a certain number of leading zeros, which depends on the difficulty level of the network. According to [12, 56], the number of leading zeros should be at least 32. The difficulty level is determined by the amount of time it takes to solve this problem and is calculated using the equation:

$$D = \frac{T_1}{T}$$

Here, $D$ represents the difficulty level, $T_1$ represents the target difficulty of 1, and $T$ represents the current target where targets are 256-bit numbers with at least 32 leading zeros [56]. For example, if the target for difficulty 1 is

$$2^{224}$$

and the current target is

$$1.5 \times 2^{220},$$

then the difficulty level is

$$\frac{2^4}{1.5} = \frac{32}{3}.$$

Consider the level of difficulty based on the number of entities involved and the amount of time needed. In that case, many entities can show high difficulty levels in an aggregate way, even if each entity is only performing a relatively inexpensive task. The probability of finding a valid hash value with $m$ leading zeros is inversely proportional to $2^m$, assuming that the hash function used in PoW, i.e., SHA-256 [19], behaves like a random oracle. Therefore, on average, a miner has to try $2^{m-1}$ different inputs to find a hash value with at least $m$ leading zeros. This means that the higher the difficulty level, the more computational power and energy are required to solve the PoW puzzle.

According to benchmark results in [57], it has been found that general-purpose CPUs are about one order of magnitude slower than GPUs and four orders of magnitude slower than ASICs. Moreover, mobile CPUs such as Cortex-A9 are two orders of magnitude slower than desktop CPUs. Based on this information, a comparison can be made between the performance of different hardware types, such as mobile CPUs, desktop CPUs, GPUs, and ASICs, in terms of hash operations and computation time. Figure 3.1 shows the expected number of hash operations required to achieve a certain level of difficulty on different types of machines.

It can be observed that the number of hash operations required is consistent across all types of machines. Additionally, Figure 3.2 shows the expected computation time required to achieve the same level of difficulty on different types of machines. The computation time required varies significantly across different types of machines, as shown. Based on this, mobile CPUs such as ARM CPUs have no chance of mining successfully due to their low speed.

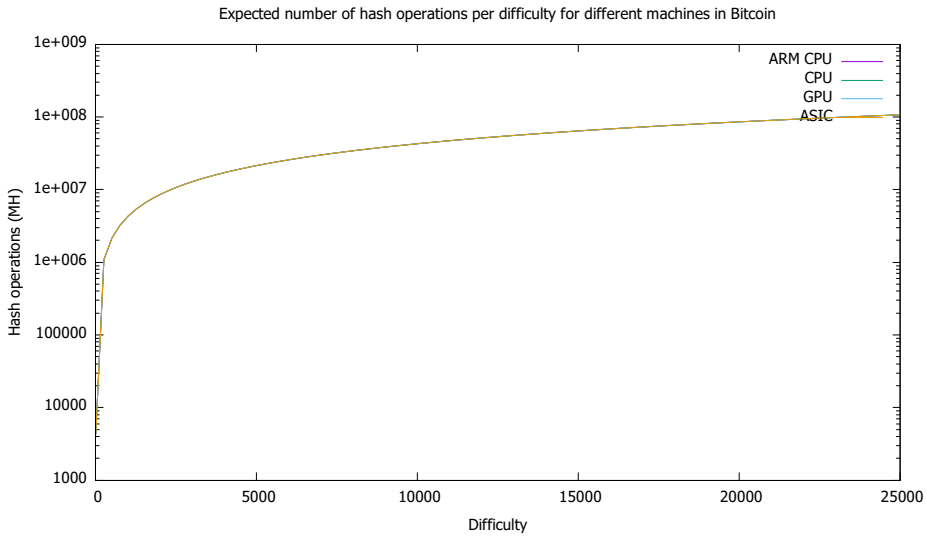



**Figure 3.1:** Expected number of hash computation per difficulty for different types of machines

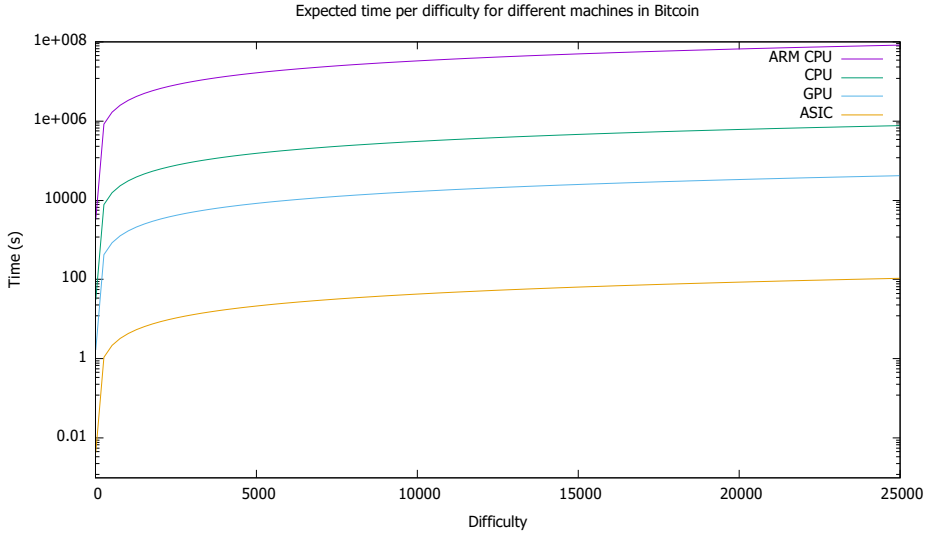



**Figure 3.2:** Expected time of hash computation per difficulty for different types of machines

### 3.1.2 Sincerity

In this study, a new way of looking at the mining process is proposed: *Sincerity*. The unit of sincerity, denoted as $\mathfrak{S}$, is defined to quantify the extent to which an entity is willing to allocate its own resources towards computing a hash code with a

specific number of leading zero bits. For example, $1\mathfrak{S}$ means the sincerity level to produce a hash code with 1 leading zero bit, and $m\mathfrak{S}$ means the sincerity level to produce a hash code with $m$ leading zero bits. Since obtaining a certain number of leading zeros in a hash function requires an exponentially increasing number of hash operations, the following properties of $\mathfrak{S}$ can be derived:

$$32\mathfrak{S}$$

$$= 2^1\ 31\mathfrak{S}s$$

$$= 2^2\ 30\mathfrak{S}s$$

$$\vdots$$

$$= 2^{31}\ 1\mathfrak{S}s.$$

As a generalization, it is observed that

$$m\ n\mathfrak{S}s$$

$$= m2^1\ n-1\mathfrak{S}s$$

$$\vdots$$

$$= m2^{n-1}\ 1\mathfrak{S}s$$

$$= (\lg(m)+n)\mathfrak{S},$$

and

$$m\mathfrak{S} = 2^{m-1}\ 1\mathfrak{S}s$$

where $\lg(m)$ is $\log_2(m)$. The above equations imply that finding one additional leading zero bit in the hash code requires twice as many hash operations. To successfully mine a block, at least one miner must achieve the predetermined number of leading zero bits. The mining will be successful if there is at least one miner with the expected difficulty level with a certain predetermined number of leading zeros. For miners who generate hash codes with fewer leading zeros, their level of contribution can be determined using the sincerity level $\mathfrak{S}$, and they can be rewarded accordingly based on their contribution. Specifically, the ratio of a miner's $\mathfrak{S}$ value to the successful miner's $\mathfrak{S}$ value will determine their reward.

The figures depicted in Figure 3.3 and 3.4 illustrate the expected number of hash operations and computation time for different levels of sincerity across various machine types such as mobile ARM CPU, desktop CPU, GPU, and ASIC, respectively. Figure 3.4 demonstrates that different machines can exhibit different levels of sincerity, i.e., $s1$ for ARM CPU, $s2$ for desktop CPU, $s3$ for GPU, and $s4$ for ASIC at a given time $t$. In contrast to difficulty level, machines can demonstrate significantly different levels of sincerity by spending the same amount of time. This characteristic can be utilized to design the PoSn consensus method.

### 3.1.3 Fairer Rewarding Scheme

The PoSn scheme can promote a fairer reward system that allows miners with lower computing power to participate and receive rewards. To explain this system, suppose that the current reward amount is represented by $R$, and $n$ and $u$ represent the necessary and sufficient sincerity level to receive a reward, which is determined

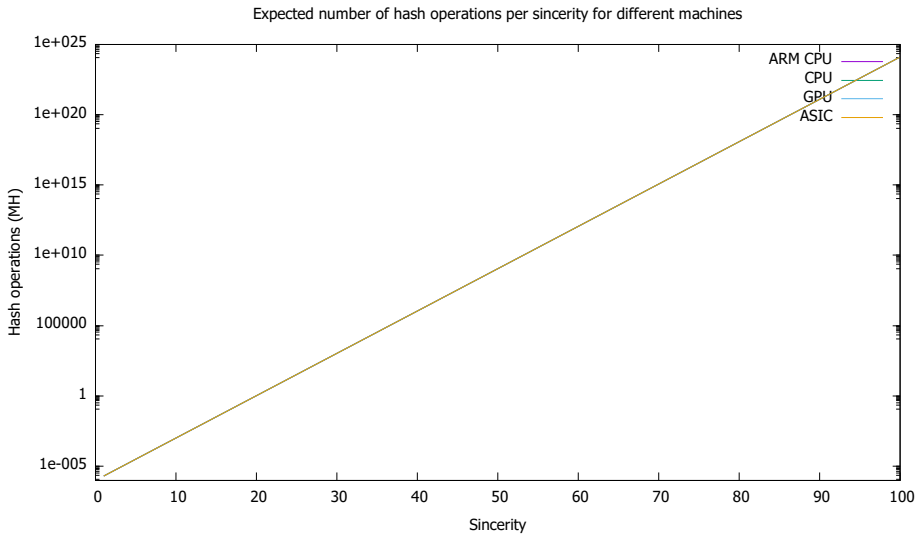



**Figure 3.3:** Expected number of hash computation per sincerity for different types of machines

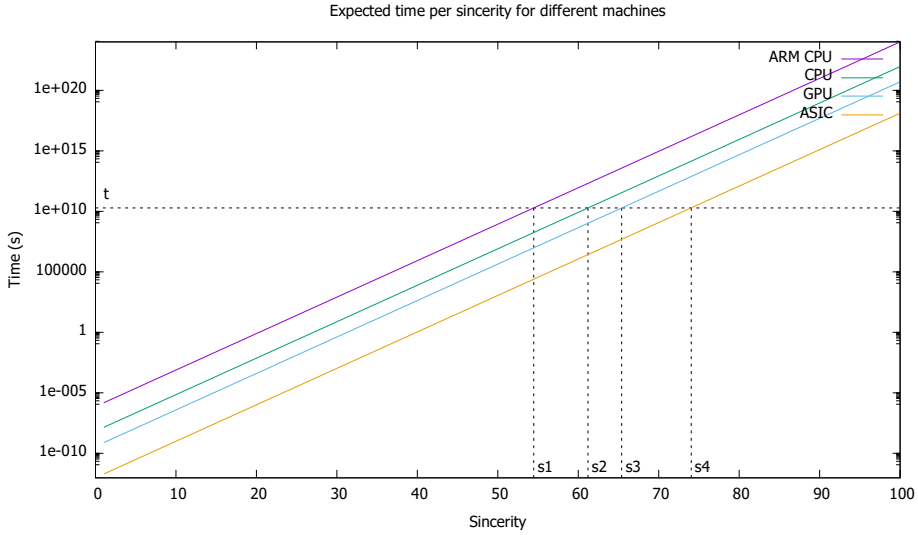



**Figure 3.4:** Expected time of hash computation per sincerity for different types of machines

by the number of leading zeros in the hash code. Additionally, consider there are three miners, A, B, and C, with sincerity levels $s_A, s_B, s_C$ where $s_A, s_B, s_C \geq n\mathfrak{S}$ and $s_A \geq u\mathfrak{S}$ but $s_B, s_C < u\mathfrak{S}$. This implies that all three miners are eligible for a reward, but only miner A is successful in generating a hash code with at least $u$ leading zeros.

In the PoW scheme, miner A would receive a reward, while miners B and C would not receive any rewards due to their insufficient resources. However, in the PoSn scheme, miners B and C would also get some reward proportional to their sincerity levels, as follows:

$$R \cdot \frac{2^{s_B-1}}{2^{u-1}} = R \cdot 2^{s_B-u}$$

and

$$R \cdot \frac{2^{s_C-1}}{2^{u-1}} = R \cdot 2^{s_C-u},$$

respectively. Miner A would get the remaining reward after subtracting the rewards of miners B and C, as follows:

$$R \cdot (1 - (2^{s_B-u} + 2^{s_C-u})).$$

Now, consider another scenario. Using Figure 3.4, if the sincerity level of

$$(2^{s1-1} + 2^{s2-1} + 2^{s3-1} + 2^{s4-1})\ 1\mathfrak{S}s$$

$$= (\lg(2^{s1-1} + 2^{s2-1} + 2^{s3-1} + 2^{s4-1}) + 1)\mathfrak{S}$$

has been achieved in an aggregate way using time $t$ and which is larger than or equal to $u$, it can be concluded that the mining operation has succeeded. The user with ARM CPU will get the reward of

$$R \cdot 2^{s1-u},$$

the user with CPU will get the reward of

$$R \cdot 2^{s2-u},$$

the user with GPU will get the reward of

$$R \cdot 2^{s3-u},$$

and the user with ASIC will get the reward of

$$R \cdot (1 - 2^{s1-u} - 2^{s2-u} - 2^{s3-u}).$$

This way, the PoSn scheme incentivizes miners to participate in mining even if they have the low computing power and rewards them according to their contribution.

### 3.1.4 Comparison of Different Consensus Mechanism

The comparison of various consensus protocols has been briefly summarized in TABLE 3.1 based on their advantages and disadvantages. The proposed PoSn consensus scheme has the benefit of allowing a more significant number of miners to participate, even those with relatively small computing power. This can be achieved by reducing the sincerity/difficulty level required for the miners to post their mining code so that smaller miners can have a chance to contribute. This enhances participation and prevents the domination of a few powerful miners who may act maliciously or compromise the security of the blockchain ledger. Safeguards against a 51% attack can be implemented by enforcing a rule where any miner can submit a mining code with $(s-1)\mathfrak{S}$ level when $s\mathfrak{S}$ level is required.

One disadvantage of the proposed scheme is that extensive resources are still required for sincere work, although this can be regulated by adjusting the sincerity/difficulty requirement. In fact, the proposed scheme deviates from the conventional 10 minutes/block rule observed in BitCoin. Instead, it focuses on ensuring security through a large number of participants, eliminating the need for an

**Table 3.1:** Comparison of different consensus mechanism

| Consensus | Advantage | Disadvantage |
|---|---|---|
| PoW | Any node can theoretically participate because of it's simplicity | Extensive energy required & vulnerable to 51% attack |
| PoS | Energy efficient & Eco-friendly | In order to participate in the network, previous resource required & vulnerable to nothing at stake attack |
| PoA | Mining open for any miner & Reduce hyperinflation | Extensive resource required & vulnerable to nothing at stake attack |
| PoT | Resource intensive & Eco-friendly | Trusting third party execution environment |
| PoSn | Any miner can practically participate & can even get rewarded & Can be configured to prevent 51% attack | More energy required than PoS, but can be configured to lower energy consumption |

excessively high sincerity/difficulty level. By fostering a robust and diverse network of miners, the scheme achieves a distributed consensus mechanism that enhances security while maintaining a more flexible block generation time.

## 3.2 Blockchain Based Storage Architecture

One of the challenges of data management is the growing volume of data that requires reliable and secure storage solutions. Blockchain systems offer a promising alternative to traditional cloud storage platforms, as they do not rely on any central authority and guarantee the properties of decentralization, immutability, and integrity of the stored data. However, most existing Blockchain consensus mechanisms are designed for transactional data only and may not be suitable for other data types. A

novel Blockchain consensus mechanism is presented in [55], which is suitable for data storage and validation and can be scaled to mobile devices. Incorporating a two-step validation process with these devices ensures that the proposed consensus mechanism is more secure and cost-effective than centralized cloud-based storage systems. The contribution of this work can be summarized as follows:

- A consensus mechanism framework is proposed for Blockchain-based distributed storage, which includes mobile devices as validators.
- A game-theoretic analysis is conducted to evaluate the consensus mechanism by modeling the strategies and outcomes of the validators under different scenarios, using Nash Equilibrium to measure the system's stability.
- A queuing-theoretic analysis is performed to study the expected congestion within the network, including metrics such as the expected waiting time and the number of participants. This will allow participants to detect unusual activities in the network.

### 3.2.1 Proposed Scheme

#### Entities

The proposed consensus mechanism involves two types of entities: block-warehouses, which store the chain of blocks, and users who need access to the chain. Within this mechanism, there are three main roles: *Selectors*, *Validators*, and *Invalidators*. Selectors are entities with high storage capabilities, while validators can be any mobile or personal device that can connect to the Blockchain network. Invalidators are a subset of validators who can detect and report any errors or misbehaviors in the validation process. To ensure the accountability of each selector,

a counterpart of the selector in the consensus mechanism is introduced. These counterparts are called Positive Selectors and Negative Selectors, respectively.

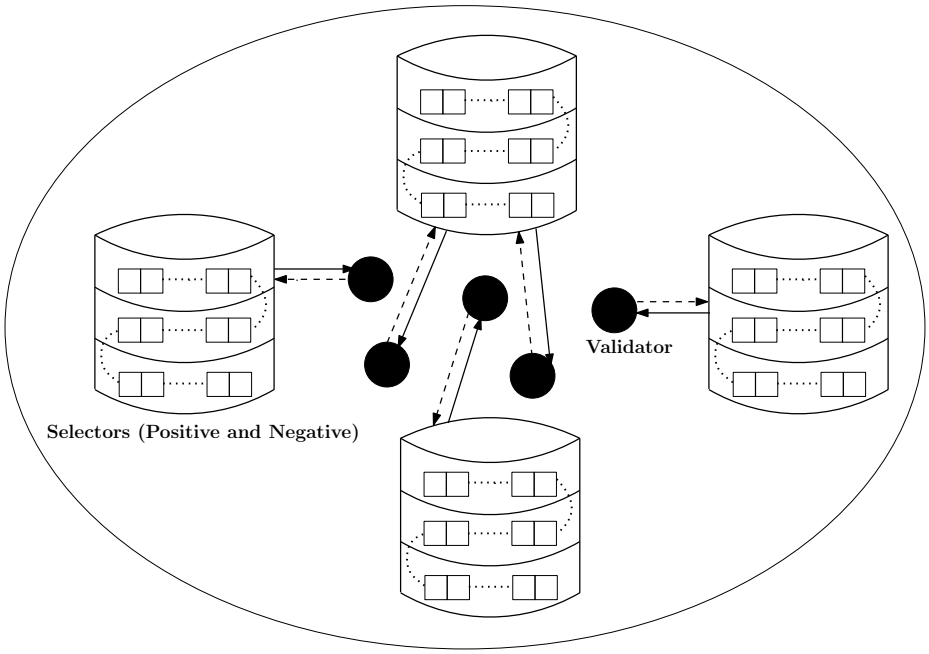



**Figure 3.5:** Diagram of role interactions

### Choosing Positive Selectors and Negative Selectors

A new method of selecting positive and negative selectors is introduced in the proposed scheme. The validations of each block in the Blockchain system are split into several parts. A generic way to define the validation proof of a block part is to use the ID of the latest block that is connected to that part. The connection can be defined for different applications; for example, for financial transaction-based blocks, a block and a section containing the same transaction entities can be connected. The validation work is low in computation and only requires verifying the proof. Since there may be multiple mining attempts and forking is not allowed, a (temporary) central entity is relied upon to collect and choose one validation work among them. The payment

or compensation for the validator can be designed according to the applications. For instance, a possible approach is to have the rewarded miner pay the validator.

The main idea of this selector scheme is to let each block-warehouse take turns to provide selector service for each block and to ensure that this rotation is as fair as possible. Achieving a completely fair rotation without the involvement of a central server would be challenging. Therefore, the goal is to achieve a rotation that is fair enough or probabilistically fair. To accomplish this, the scheme requires every block-warehouse to announce and register itself with the other block-warehouses, ensuring that every block-warehouse has the same list of participating block-warehouses. Each block-warehouse is then expected to locate the same selector based on the list and the rule.

A possible way to design the selector rotation rule is to base it on the block-warehouses' IDs, which are some public and verifiable information such as the miner's public key. The rotation of the selector service can start with the block-warehouse that has the lowest ID for the first block and then select the following lowest ID for the second block, and so on. However, this approach may not always ensure a fair selection of selectors, as it could perpetually avoid certain block-warehouses as selectors. For example, if new block-warehouses with a lower ID than the current selector join after each block verification, any block-warehouse with a higher ID than the current selector will be skipped forever. This could become a significant issue depending on the fee/compensation scheme for the selectors.

Block-related information, particularly the block's hash code, is utilized to prevent unfair selector choices and maintain a probabilistically fair rotation. This is

because the hash code of a block is pseudorandom, which means that each possible hash code is almost equally likely. The two block-warehouses with the IDs that are closest to the block-related information will be chosen as the selectors for the block. Since this type of choice depends on block-related information, fairness can be compromised if the new block announcer can manipulate this information. However, if the block-related information, such as the hash code, is appropriately processed, it can be challenging to manipulate it computationally and disrupt fairness. Thus, this approach achieves a fair rotation with a high probability.

Each block in the proposed scheme has two selectors chosen: a positive selector and a negative selector. The positive selector's main task is to divide the block into sections and collect proofs of sectional validation, from which one will be chosen for reward. On the other hand, the negative selector's primary responsibility is to collect proof of invalidation, which can point out any wrongdoing by positive selectors or validators and announce the finalization of the block otherwise. It's important to note that the two selectors can invalidate each other's previous work.

**Block Generation Process**

Once a positive selector is chosen through the process described in 3.2.1, it will begin working on two tasks sequentially: 1) validating the previous negative selector's work and 2) dividing the block into non-overlapping sections. During the validation process, if an invalidator submits a report with proper proof, the positive selector will penalize the negative selector's reward and redistribute it to the invalidator and itself. The positive selector will finalize the previous block by signing it with its public

ID, including information such as the penalty of the negative selectors and reward for the invalidators. Then it will announce the sections for the current block to the network by signing with its public ID. Validators may choose any section for validation and sign their validation proof with their public ID, which the positive selector also validates. The positive selector selects the first correct validator whose submission matches its validation proof and announces the winners for each section by signing with its public ID.

If there are no errors reported by the invalidators, the negative selector records the block by signing it and including all relevant information, such as the positive selector, negative selector, winners, and rewards for each section. If there are errors in the validation process, invalidators can notify the negative selectors with proper proof to bring themselves into the block creation process. Examples of wrongdoings in validation include intentionally or unintentionally choosing the wrong validator or choosing a validator whose submission time is significantly later than other legitimate validators. In the case of wrongdoing, the negative selector can deduct the reward from the entities involved and redistribute it to the positive selector, winner validators, and the negative selector. Once all information is recorded in the block header, the negative selector waits for the next block's positive selector to finalize the block.

The process of generating a block can be divided into two stages: 1) Announcement and 2) Sectional Validation. The main aspects of the block generation process, along with three different scenarios, have been illustrated in the workflow diagram shown in Figure 3.6. These scenarios include: 1) All the entities are honest, 2) Positive selector or Validator is dishonest, and 3) Negative selector or Invalidator is dishonest.

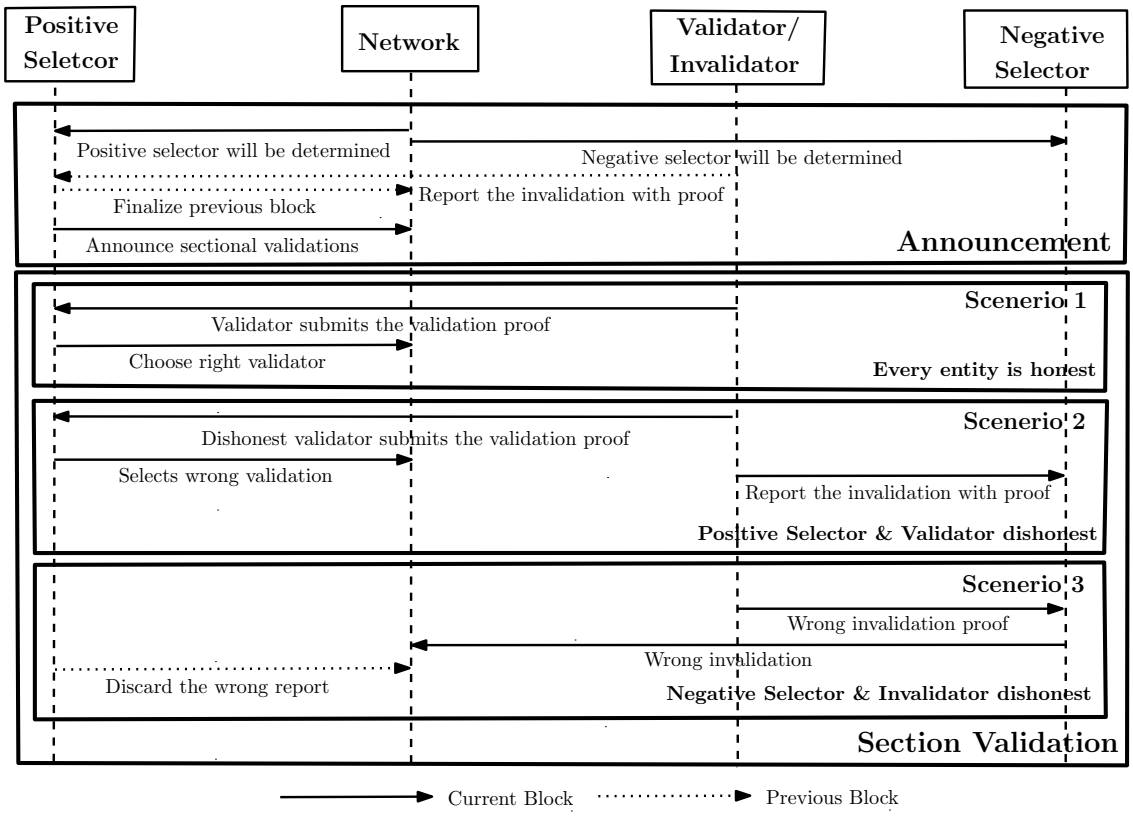



**Figure 3.6:** Workflow of section announcement and section validation

In the third scenario, any erroneous invalidation recorded by the negative selector will be addressed by the next block's positive selector. It is possible for all entities involved to be dishonest, but the likelihood of such a scenario occurring is low. We examine this scenario using the concept of Nash Equilibrium in Section 3.2.2.

### 3.2.2 Analysis

#### Game Theoretic Analysis

The proposed consensus approach operates under the assumption that entities are rational decision-makers, meaning they will strive for the maximum outcome for their strategies in their interactions regarding consensus while considering the strategies of other entities. This system can be modeled and analyzed using Game

theory tools, and the behaviors of each entity can be predicted. One of the most important and influential tools is Nash Equilibrium [58], which defines the optimal outcome of an interaction where no participants have the incentive to change their strategies. According to Osborne and Rubinstein [59], this interaction process can be categorized as *games*, and the entities are *players*.

The Nash equilibrium has been calculated to analyze the optimal condition of the consensus approach employed. Two validation games were formulated to calculate the Nash equilibrium, considering different behavioral situations (Honest and Dishonest). The payoff of each entity in the system was predicted as part of the calculation process. Dishonest behavior can be either intended or unintended. The validation game can be either "At least one honest selector is present" or "No honest selector is present". For the sake of simplicity, the constructed games were focused on only one section of a block. However, these games are valid for all sections in a block without logical difficulty. The payoff matrices for the two games, representing the strategies of the validators and selectors' behavior, were presented in Table 3.2 and Table 3.3, respectively.

**Table 3.2:** When at least one honest selector is present in the validation process

| Entities | | Positive Selector & Negative Selector | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Behavior | | 00 | | | 01 | | | 10 | | | 11 | | |
| Validators | 0 | 5 | 500 | 500 | 5 | 500 | -10000 | (10000+5)/2 | -10000 | (10000+5)/2 | 5 | -10000 | -10000 |
| | 1 | -100 | 500 | 600 | -100 | 500 | -10000 | -100 | -10000 | 10000+(500+100) | -100 | -10000 | -10000 |

In these tables, the columns denote the selectors' strategies, while the rows represent the validators' strategies. The table's data cell represents the payoff for the

validators, positive selectors, and negative selectors for each combination of strategies among the selectors and validators. Honest behavior is represented as "0", and dishonest behavior is represented as "1".

**Table 3.3:** When there is no honest selector is present in the validation process

| Entities Behavior | | Positive Selector & Negative Selector | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 00 | | | 01 | | | 10 | | | 11 | | |
| Validators | 0 | 5 | 500 | 500 | 0 | -10000 | (500+10000)/x | (5+1000)/2 | -10000 | (5+1000)/2 | 0 | 500 | 500 |
| | 1 | -100 | 500 | 500+100 | 5 | -10000 | (500+1000)/(x+1) | -100 | -10000 | 500+(100+10000) | -100 | 500 | (500+100)/x |

The Nash equilibrium was calculated using Gambit [60], an open-source collection of tools widely used in game theory. The global Newton method [61] was employed to perform the Nash equilibrium calculations. The two games discussed earlier were formulated and implemented, assuming certain payoff values. The highest payoff for validators was set to 100, while the highest payoff for selectors was set to $10,000$. For honest validators, the payoffs were assumed to be 5, 500, and 500, respectively. On the other hand, the payoffs (penalties) for dishonest validators, positive selectors, and negative selectors were assumed to be $-100$, $-10,000$, and $-10,000$, respectively.

**One Honest Selector is Present in Validation**

In this game, at least one honest selector is present, and the honest selector has the ability to identify the dishonest behavior of the entities. As a result, the dishonest entities will face a penalty, and their forfeited payoff will be distributed among the other honest entities. The Nash equilibrium analysis shows that the probability of all entities behaving honestly is 1, while the probability of all entities behaving dishonestly is 0.

**No Honest Selector is Present in Validation**

In this game, the dishonest entities will benefit from the lack of a honest selector, as they will get the payoff that the honest entities lose. In the payoff matrix, 'x' represents the number of the colluding selectors. The number of colluding selectors and the "penalty ratio," which increases the base penalties, were systematically varied to explore their impact on the Nash equilibrium. Notably, it was observed that as the number of colluding selectors exceeded a certain threshold, the Nash equilibrium transitioned towards a probability of 1 for honest entities and 0 for dishonest entities. This finding suggests that the presence of a critical mass of colluding selectors can significantly affect the equilibrium behavior of the system, favoring the dominance of honest participants. Figure 3.7 and Figure 3.8 show two 3D plots of the Nash profile for positive selectors with penalty ratios of 2 and 5, respectively. The axes are $X$, $Y$, $Z$ for the Nash probabilities of honest behavior, dishonest behavior, and the number of colluding selectors. Figure 3.9 shows a linear relationship between the number of colluders and the number of selectors, which also indicates the minimum number of dishonest selectors.

### 3.2.3 Queuing Theoretic Analysis

A first-come-first-serve queue model was employed to analyze the expected waiting time experienced by validators in the system. The model assumes that validators can submit their validation work for any section without restrictions and follow a Poisson process with an arrival rate denoted by $\lambda$. Validation checks are performed by selectors at a service rate denoted by $\mu$. Both the arrival and service times

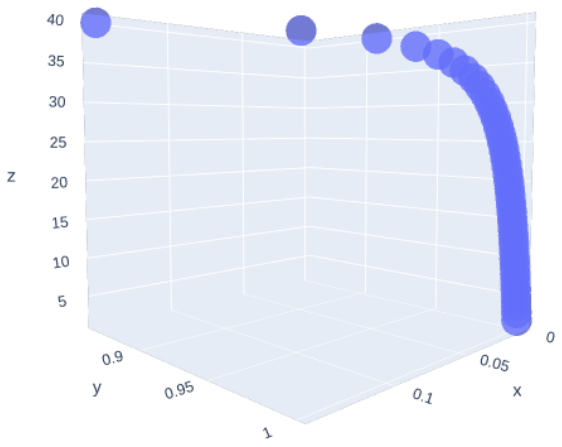



**Figure 3.7:** Nash profile for the positive selector for penalty ratio 2

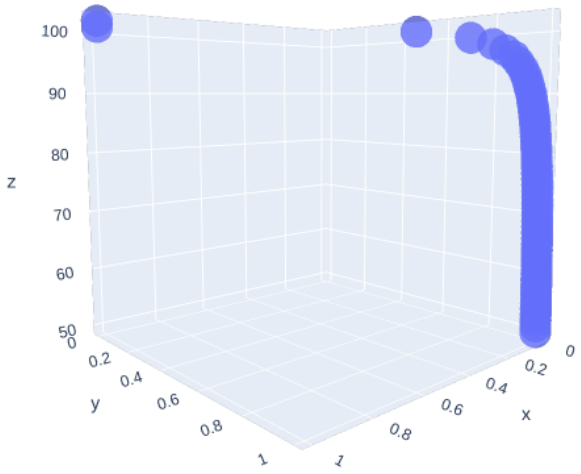



**Figure 3.8:** Nash profile for the positive selector for penalty ratio 5

are assumed to follow an exponential distribution. This queueing model corresponds to the M/M/1 queue model, a well-known concept in queueing theory. The queue model and the flow diagram for the M/M/1 queue are illustrated in Figure 3.10.

Consider the probability of the system being in state $n$, i.e., having $n$ validators in the queue, by $p_n$. The following relation can be derived from the flow diagram:

$$\lambda p_n = \mu p_{n+1} \Rightarrow p_{n+1} = \frac{\lambda}{\mu} p_n$$

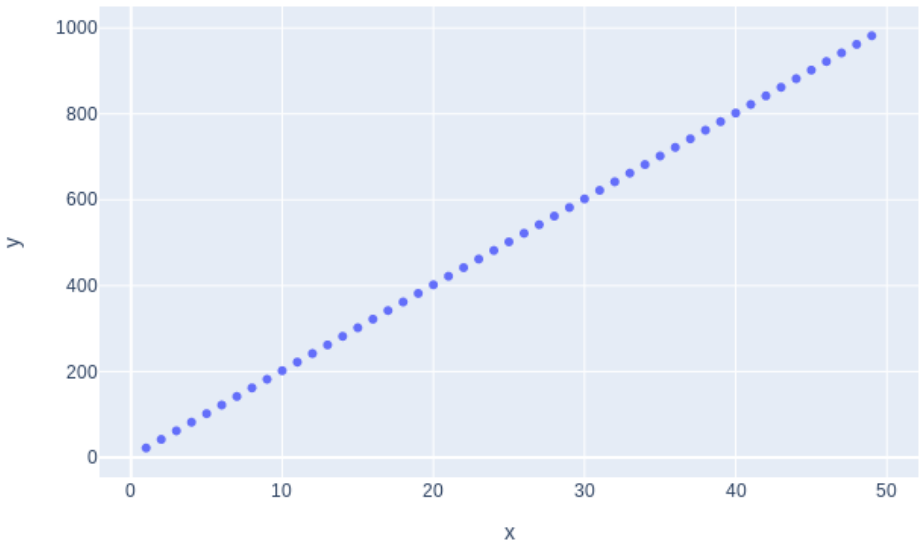



**Figure 3.9:** Number of colluder vs Penalty ratio

Let $\rho = \frac{\lambda}{\mu}$ and note that the sum of all state probabilities must be equal to 1.

$$1 = \sum_{0}^{\infty} p_i = \sum_{0}^{\infty} \rho_i p_0 = \frac{p_0}{1-\rho} \Rightarrow p_0 = 1 - \rho.$$

By applying Little's theorem [63], several useful characteristics of the system can be obtained. The expected number of validators in the system, $E[N]$, is given by

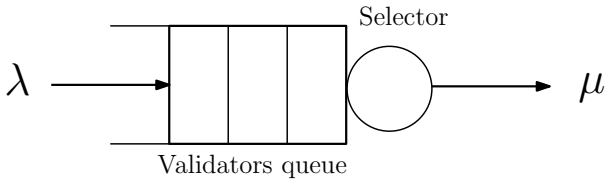
$$E[N] = \sum_{0}^{\infty} n p_n = \sum_{0}^{\infty} n(1-\rho)p^n = (1-\rho)\sum_{0}^{\infty} n p^n = \frac{\rho}{1-\rho}$$

The expected time in the queue for a validator, $E[T]$, can be computed using Little's theorem as well,

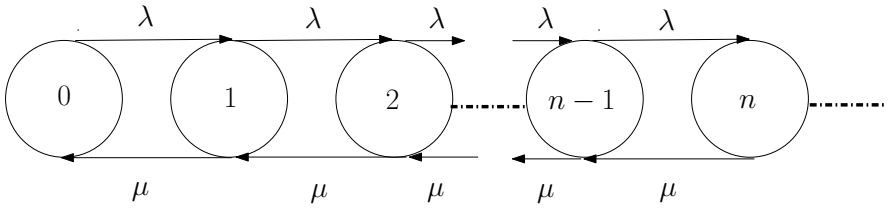
$$E[N] = \lambda E[T] \Rightarrow E[T] = \frac{\rho}{\lambda(1-\rho)} = \frac{1}{\mu - \lambda}$$

The total expected time in the queue system $T$ for a validator consists of the expected time in the queue $T_q$ and the expected validation time (service time).

$$E[T] = E[T_q] + \frac{1}{\mu} \Rightarrow E[T_q] = \frac{1}{\mu - \lambda} - \frac{1}{\mu} = \frac{\rho}{\mu(1-\rho)}$$




**(a)** $M/M/1$ queue model




**(b)** Flow diagram for $M/M/1$ queue model

**Figure 3.10:** $M/M/1$ queue

Similarly, the expected number of validators in the queue $N_q$ is given by

$$E[N_q] = \frac{\rho^2}{1-\rho}.$$

### 3.2.4 Conclusion

In this study, a new consensus mechanism for Blockchain systems is presented to address the challenges of centralization and security. The centralization issue arises from the dominance of a few powerful entities in the mining or validation process, leading to various attacks like 51% attack, DDoS attack, DNS attack, consensus delay, and selfish mining. Security concerns focus on ensuring the integrity and validity of transactions and blocks while resisting such attacks. The proposed consensus mechanism introduces two novel features: probabilistic fair rotation of selectors and block segmentation. The former ensures that the selection of validators is random and fair, preventing any entity from gaining undue influence over the network. The latter allows each block to be divided into multiple sections, which increases user participation

and reduces the computational and storage requirements for validators. Additionally, multiple validations per block enhance the overall security of the Blockchain system.

The applicability of the consensus mechanism to different applications and its support for various reward mechanisms are discussed. Game theory models demonstrate that the proposed mechanism incentivizes rational entities to behave honestly and cooperate, discouraging attacks and misbehavior. The use of queuing theory allows for performance analysis, particularly in terms of network congestion and waiting delay, providing insights into the mechanism's efficiency.

# CHAPTER 4

# SUPERSINGULAR ISOGENY BASED HASH FUNCTION

This chapter contains content that was previously published in:

M. U. Zaman, A. Hutchinson, and M. Min, "Implementation Aspects of Supersingular Isogeny-Based Cryptographic Hash Function," in Wireless Internet: 15th EAI International Conference, WiCON 2022, Virtual Event, November 2022, Proceedings, 2023, pp. 14–27.7 [47].

## 4.1 Implementation Aspects of Supersingular Isogeny-Based Cryptographic Hash Function

This section discusses the practical aspects of implementing the compact version of the CGL hash function using different types of elliptic curves (Weierstrass, Montgomery, and Legendre). The study demonstrates that the utilization of the distinct features of each form of the elliptic curve enables the elimination of certain redundant computations in the original proposals of the CGL hash function. Additionally, experiments were conducted with the implemented algorithms to compare their running time and total number of collisions.

### 4.1.1 Compact CGL Hash Algorithms in Short Weierstrass Form

An elliptic curve $E$ over a prime field $\mathbb{F}_{p^2}$ with $p > 3$ can be transformed into a short Weierstrass form,

$$E : y^2 = x^3 + \alpha x + \beta,$$

where $(\alpha, \beta) \in \mathbb{F}_{p^2}$ and $4\alpha^3 + 27\beta^2 \neq 0$. This condition ensures that the curve has three distinct roots. The $j$-invariant of the elliptic curve $E$ can be expressed as,

$$j(E) = 1728(4\alpha^3)(4\alpha^3 + 27\beta^2)^{-1}.$$

In this dissertation, the division operation within the field is represented as the product of the numerator and the multiplicative inverse of the denominator. For instance, if $c$ and $d$ are two elements in the field $\mathbb{F}_{p^2}$, the division of $c$ by $d$ is denoted as $cd^{-1}$.

**2-Isogenies**

Let $\gamma$ be a root of $x^3 + \alpha x^2 + \beta$. By performing algebraic polynomial division with $(x - \gamma)$, one can obtain $(x^2 + \gamma x + (\alpha + \gamma^2)$. Therefore, the elliptic curve $E$ can be expressed as,

$$E : y^2 = (x - \gamma)(x^2 + \gamma x + (\alpha + \gamma^2)).$$

Hence, the three torsion points of order 2 are $(\gamma, 0), ((-\gamma \pm \sqrt{-4\alpha - 3\gamma^2})2^{-1}, 0)$. According to [64, Theorem 6.13], if $(x_0, 0)$ is a torsion point of order 2, applying a 2-isogeny with $ker(\phi) = \langle (x_0, 0) \rangle$ produces the following mapping:

$$\phi : E \to \tilde{E} : y^2 = x^3 + \alpha' x + \beta'$$

$$(x, y) \mapsto \left((x^2 - x_0 x + t)(x - x_0)^{-1}, ((x - x_0)^2 - t)(x - x_0)^{-2} y\right),$$

where, $t = 3x_0^2 + \alpha$, $\alpha' = \alpha - 5t$, $w = x_0 t$, and $\beta' = \beta - 7w$.

**CGL Hash Algorithm for Short Weierstrass Form**

The algorithm 1 takes four input parameters: $\alpha$, $\beta$, $\gamma$ over the field $\mathbb{F}_{p^2}$, and an $n$-bit message $M = (b_1, b_2, \cdots, b_n)$. Here, $\alpha$ and $\beta$ define a supersingular elliptic curve $E$ in Weierstrass form $y^2 = x^3 + \alpha x + \beta$ over $\mathbb{F}_{p^2}$, and $\gamma$ is one of the roots of the equation $x^3 + \alpha x + \beta$. In steps 2 and 3 of Algorithm 1, the $x$-coordinates of the other two torsion points $(x_0, x_1)$ are computed. Steps 4 to 7 establish a uniform convention for determining $x_0$ and $x_1$ in order to navigate the next steps in the graph. The convention dictates that $x_0$ is chosen as the maximum value from step 3 for bit 1, and as the minimum value for bit 0. Similarly, $x_1$ is chosen as the opposite value for each bit. Steps 9 and 10 utilize V'elu's formula to calculate the Weierstrass parameters $\alpha$ and $\beta$ of the next isogenous elliptic curve. In step 11, the $x$-coordinate of the dual isogeny or backtracking point in the new isogenous curve is computed. The output of the algorithm, obtained in step 13, is the $j$-invariant of the last elliptic curve.

**Algorithm 1** CGL hashing using Weierstrass Curve

**Input:** $\alpha, \beta, \gamma, M$
**Output:** $j$-invariant
1: **for** *bit* in $M$ **do**
2: $\delta \leftarrow \sqrt{-4\alpha - 3\gamma^2}$
3: $(x_0, x_1) \leftarrow ((-\gamma + \delta)2^{-1}, (-\gamma - \delta)2^{-1})$
4: **if** $bit \leftarrow 1$ **then**
5: $(x_0, x_1) \leftarrow \max(x_0, x_1), \min(x_0, x_1)$
6: **else** $\{bit \leftarrow 0\}$
7: $(x_0, x_1) \leftarrow \min(x_0, x_1), \max(x_0, x_1)$
8: **end if**
9: $t \leftarrow 3x_0^2 + \alpha$, $w \leftarrow x_0 t$
10: $\alpha \leftarrow \alpha - 5t$, $\beta \leftarrow \beta - 7w$
11: $\gamma \leftarrow (x_1^2 - x_0 x_1 + t)(x_1 - x_0)^{-1}$
12: **end for**
13: **return** $j$-invariant $\leftarrow 1728(4\alpha^3)(4\alpha^3 + 27\beta^2)^{-1}$

### 4.1.2 Compact CGL Hash algorithms in Montgomery Form

The Montgomery form of an elliptic curve over the field $\mathbb{F}_{p^2}$ can be represented by the equation:

$$E_{(A,B)} : By^2 = x^3 + Ax^2 + x = x(x^2 + Ax + 1),$$

where $A$ and $B$ are elements of $\mathbb{F}_{p^2}$ and $B(A^2 - 4) \neq 0$. The parameter $A$ primarily determines the geometry of the curve $E_{(A,B)}$. The $j$-invariant of the curve can be calculated as:

$$j(E_{(A,B)}) = 256(A^2 - 3)^3(A^2 - 4)^{-1}.$$

**2 Isogenies**

From the curve equation $E_{(A,B)}$, it can be shown that $Q = E(0, 0)$ be a $K$-rational point of order 2. If $P = (x_P, 0)$ is another $K$-rational point of order 2, then $x_P^2 + Ax_P + 1 = 0$. Therefore, the other two rational points of order 2 are $((-A \pm \sqrt{A^2 - 4})2^{-1}, 0)$. According to [65, §4.2], applying a 2-isogeny with $ker(\phi) = \langle P \rangle$ generates the following mapping:

$$\phi : E \to \tilde{E} : by^2 = x^3 + ax^2 + x = x(x^2 + ax + 1)$$

$$(x, y) \mapsto (g(x), yg'(x)),$$

with $b = x_P B$, $a = 2(1-2x_P)^2 = 2-(-A\pm\sqrt{A^2 - 4})^2$, and $g(x) = x(x_P x-1)(x-x_P)^{-1}$. Moreover, [65, Corollary 1] shows that the kernel of the dual of $\phi$ is $\langle (0, 0) \rangle$. Hence, $ker(\hat{\phi}) = \langle (0, 0) \rangle$. On the other hand, the authors in [8, Eqn. (18) & (19)] outline the 2-isogeny for $ker(\phi) = \langle (0, 0) \rangle$ as:

$$\phi : E \to F : By^2 = x^3 + (A + 6)x^2 + 4(2 + A)x$$

$$(x, y) \mapsto \left((x-1)^2 x^{-1}, y(1-x^{-2})\right).$$

The authors in [65, Remark 6] describe an isomorphism of $F$ with the following mapping through considering, $A_s = \sqrt{A^2 - 4}$:

$$\psi : F \to G : BA_s^{-1}y^2 = x^3 - 2AA_s^{-1}x^2 + x$$

$$(x, y) \mapsto \left((x + A + 2)A_s^{-1}, yA_s^{-1}\right).$$

By applying a 2-isogeny with the kernel $\langle(0, 0)\rangle$ and performing a simple algebraic computation on the coefficients of $x^2$, it can be shown that the dual of $\psi$ is also $\langle(0, 0)\rangle$. Hence, $ker(\hat{\psi}) = \langle(0, 0)\rangle$.

**CGL Hash algorithm for Montgomery form**

The algorithm 2 takes two input parameters: $A \in \mathbb{F}_{p^2}$, which defines a supersingular elliptic curve in Montgomery form as $y^2 = x^3 + Ax^2 + x$, and an $n$-bit message $M$ $(b_1, b_2, \cdots, b_n)$. In this algorithm, the dual of the kernel is always $\langle(0, 0)\rangle$, so the backtracking point is $(0, 0)$. Therefore, there is no need to calculate the backtracking point, and the other two torsion points can be assigned for bit 1 and 0 based on a uniform convention. From step 2 to 8, a convention has been established to choose the next isogenous curve. The output of the hash function will be the $j$-invariant of the last isogenous curve.

### 4.1.3 Compact CGL Hash Algorithms in Legendre Form

The Legendre form is an interesting representation of an elliptic curve. If $\lambda \in \mathbb{F}_{p^2}$ and $\lambda \neq 0, 1$, then the elliptic curve $E$ in Legendre form can be expressed as

**Algorithm 2** CGL hashing algorithm using Montgomery Curve

**Input:** $A, B, M$
**Output:** $j$-invariant
1: **for** *bit* in $M$ **do**
2: $C \leftarrow \sqrt{A^2 - 4}$
3: $A_0 \leftarrow 2A^2 - 4 + AC$
4: $A_1 \leftarrow 2A^2 - 4 - AC$
5: **if** $bit \leftarrow 1$ **then**
6: $A \leftarrow 2 - \max(A_0, A_1)$
7: **else** $\{bit \leftarrow 0\}$
8: $A \leftarrow 2 - \min(A_0, A_1)$
9: **end if**
10: **end for**
11: **return** $j$-invariant $\leftarrow 256(A^2 - 3)^3(A^2 - 4)^{-1}$

follows:

$$E_\lambda : y^2 = x(x-1)(x-\lambda).$$

Here, Legendre coefficient, $\lambda$ for $E$ is not unique. In fact each of

$$\lambda, \lambda^{-1}, (1-\lambda), (1-\lambda)^{-1}, \lambda(\lambda-1)^{-1}, (\lambda-1)\lambda^{-1}$$

yields an Isomorphic $E$ curve. The $j$-invariant of $E\lambda$ is given by:

$$j(E_\lambda) = 2^8(\lambda^2 - \lambda + 1)^3\lambda^{-2}(\lambda-1)^{-2}.$$

Most supersingular elliptic curves will have three 2-isogenies. Therefore, the six possible Legendre coefficients that yield the same $j$-invariant can be grouped into three pairs, where each pair is directly related to one of the three 2-isogenies.

**2 Isogenies**

The three 2-torsion points $(0,0), (1,0), (\lambda,0)$ are readily available from the elliptic curve expression $E_\lambda$. From [66, Theorem 4 & 5], it can be shown that applying

a 2-isogeny when $ker(\phi) \neq (\lambda, 0)$ produces the following mapping:

$$\phi : E \to \tilde{E} : y^2 = x(x-1)(x-\lambda').$$

The relationship between $\lambda$ and $\lambda'$ can be expressed as follows.

$$\lambda = \begin{cases} (\lambda_s + 1)^2(4\lambda_s)^{-1} = (\lambda + 1 + 2\lambda_s)(4\lambda_s)^{-1} & \text{if } ker(\phi) = \langle(0,0)\rangle \\ -(\lambda_t - 1)^2(4\lambda_t)^{-1} = (\lambda - 2 + 2\lambda_t)(4\lambda_t)^{-1} & \text{if } ker(\phi) = \langle(1,0)\rangle \\ (\lambda_s + \lambda_u)^2(4\lambda_s\lambda_u)^{-1} = (2\lambda - 1 + 2\lambda_s\lambda_t)(4\lambda_s\lambda_u)^{-1} & \text{if } ker(\phi) = \langle(\lambda,0)\rangle \end{cases}$$

where, $\lambda_s = \sqrt{\lambda}, \lambda_t = \sqrt{1-\lambda}$, and $\lambda_u = \sqrt{\lambda - 1}$.

**CGL Hash Algorithm for Legendre Form**

The algorithm 3 takes two input parameters: the Legendre parameter $\lambda \in \mathbb{F}_{p^2}$, which defines the supersingular elliptic curve as $y^2 = x(x-1)(x-\lambda)$, and the $n$-bit message $M$ $(b_1, b_2, \cdots, b_n)$. Similar to the Montgomery form algorithm, the backtracking point is fixed at $(\lambda, 0)$. In steps 2 to 5 of the algorithm, the kernel point is determined based on the value of the current bit. For bit 1, the kernel point is set to $(1, 0)$, while for bit 0, the kernel point is set to $(0, 0)$. The output of the hash function algorithm will be the $j$-invariant of the last isogenous curve.

### 4.1.4 Yoshida et al. Proposed Method for Computing CGL Hash

Yoshida et al. [50, §5] introduced two efficient techniques for computing CGL hashes specifically designed for the Weierstrass form of supersingular elliptic curves. These methods are based on their proposed Theorem [50, Theorem 1], which establishes

**Algorithm 3** CGL hashing algorithm using Legendre Curve

**Input:** $\lambda, M$
**Output:** $j$-invariant
1: **for** *bit* in $M$ **do**
2: **if** $bit \leftarrow 1$ **then**
3: $\lambda \leftarrow (\lambda + 1 + 2\sqrt{\lambda})(4\sqrt{\lambda})^{-1}$
4: **else** $\{bit \leftarrow 0\}$
5: $\lambda \leftarrow (\lambda - 2 + 2\sqrt{1-\lambda})(4\sqrt{1-\lambda})^{-1}$
6: **end if**
7: **end for**
8: **return** $j$-invariant $\leftarrow 2^8(\lambda^2 - \lambda + 1)^3\lambda^{-2}(\lambda - 1)^{-2}$

a connection between the current and subsequent curve parameters utilizing a non-backtracking torsion point. Although both methods offer similar computational costs in theory, the first method as suggested was implemented.

### 4.1.5 Result and Discussion

#### Algorithmic Cost Comparison

The computational cost of the three algorithms described in section 4.1.1 is compared in Tables 4.1 and 4.2. The tables are based on the common assumption that I = 100M, S = 0.67M, and SR = (0.67 log p + 100)M, where I, S, SR, and M denote Inversion, Squaring, Square root, and Multiplication, respectively.

The computational cost measure for computing 2-isogeny sequences for $n$-bit input strings and finding the $j$-invariant of the final isogenous curve. The comparison shows that the algorithm for Montgomery and Legendre form has a lower computational cost than the algorithm for the Weierstrass form. A similar outcome is also verified by the simulation result from applying the algorithm discussed afterwards.

**Table 4.1:** CGL Hash operation counts for different algorithms

| Algorithm | Curve Model | Counts | | | |
|---|---|---|---|---|---|
| | | Multiplication | Square | Square root | Inversion |
| 1 | Weierstrass | $4n+1$ | $3n+2$ | $n$ | $n+1$ |
| 2 | Montgomery | $2n+1$ | $n+1$ | $n$ | 1 |
| 3 | Legendre | 2 | 3 | $n$ | $n+1$ |

**Table 4.2:** CGL Hash costs comparison for different algorithms

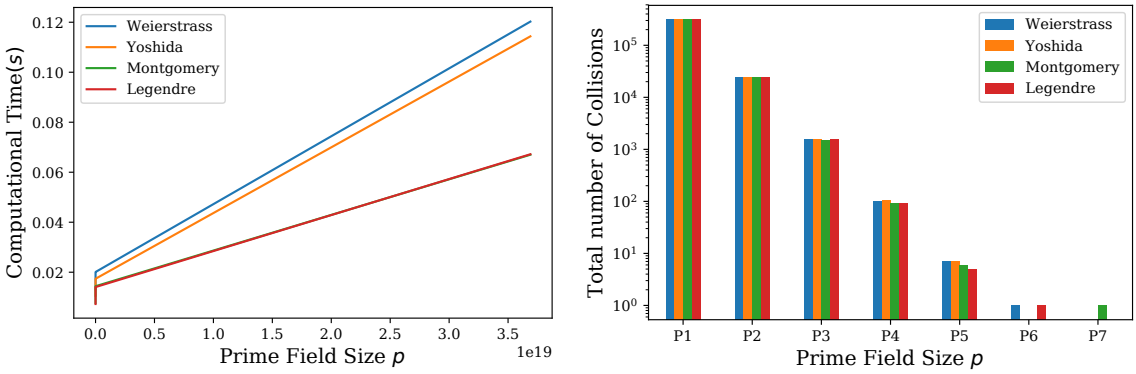
| Algorithm | Curve Model | Costs in M |
|---|---|---|
| 1 | Weierstrass | (206 + 0.67 log p)n + 102.33 |
| 2 | Montgomery | (102.67 + 0.67 log p)n + 101.67 |
| 3 | Legendre | (200 + 0.67 log p)n + 104 |

**Simulation Result**

A Python script was developed using SageMath [67] to compare the performance of algorithms 1, 2, and 3 with the algorithm proposed in [50, §5.3], which was also implemented. The simulations were conducted on an Intel® Core™ i5-7500 CPU (@ 3.40GHz x 4) running Ubuntu 18.04.6 LTS. The running time and the total number of collisions were used as the comparison criteria, as these properties are fundamental to the hash function in terms of low computational time and collision avoidance. The running time was measured by calculating the difference between the start and end times of each algorithm for 128-bit input strings, while the total number of collisions represented the count of input strings mapping to the same $j$-invariant. The evaluation was performed on $2^{20}$ unique input strings over different prime fields, and the arithmetic mean of the running times was used to measure central tendency. The starting curve for all algorithms was based on the SIKE proposed elliptic curve or its

isomorphic curve over $\mathbb{F}_{p^2}$. Specifically, the Weierstrass, Montgomery, and Legendre forms used the starting elliptic curves $y^2 = x^3 - 11x + 14$, $y^2 = x^3 + 6x^2 + x$, and $y^2 = x(x-1)(x - 17 + 12\sqrt{2})$ over $\mathbb{F}_{p^2}$, respectively.

The simulation results for all four algorithms are presented in Figure 4.1. The algorithm proposed in [50, §5.3] is labeled as Yoshida, while the other three algorithms are named after their respective curves. In Figure 4.1b, the prime numbers $2^{24} + 43$, $2^{28} + 3$, $2^{32} + 15$, $2^{36} + 31$, $2^{40} + 15$, $2^{44} + 7$, and $2^{48} + 75$ are denoted as P1, P2, P3, P4, P5, P6, and P7, respectively. The computational time, as shown in Figure 4.1a, increases with the size of the prime field, and algorithms 2 and 3 demonstrate higher efficiency compared to the other two algorithms. Figure 4.1b depicts that the total number of collisions decreases exponentially as the prime field sizes increase.




**(a)** Computational time

**(b)** Total number of collisions

**Figure 4.1:** Comparison of average computation time and total number of collisions for $2^{20}$ number of 128 input bit strings over different size of finite field $\mathbb{F}_{p^2}$.

Another experiment was conducted to analyze how the number of collisions varies with the length of the input bit string for different implemented algorithms. A collision occurs when two distinct input bit strings produce the same output elliptic

curve. In this experiment, a prime field of size $p$ was chosen, approximately $p \approx 2^{21}$, and the number of collisions was measured for all possible input bit strings up to a length of 18. Figure 4.2 illustrates that the number of collisions exhibits exponential growth as the input bit string length increases.

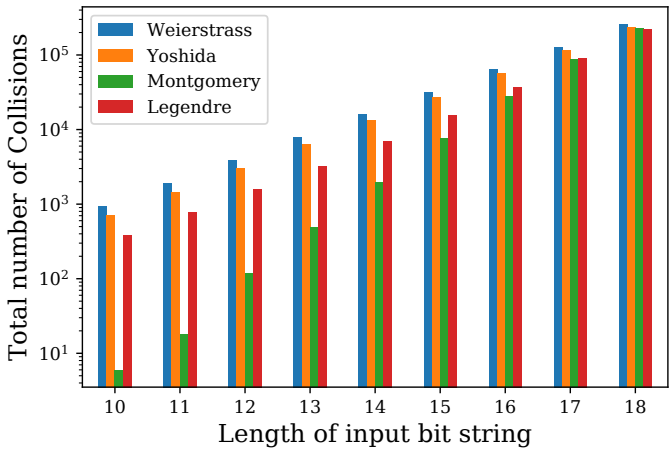



**Figure 4.2:** Total number of collisions for different length of input bit strings

It is worth noting that the curves in Montgomery form show a faster growth rate of collisions, which necessitates further analysis using different choices of elliptic curves and prime fields. However, it is evident that such behavior is undesirable for a cryptographic hash function. The results of this experiment aided in determining the appropriate size of the prime field for the design of a secure hash function.

### 4.1.6 Conclusion

In this work, four compact variants of the CGL hash function, based on 2-isogeny sequences on different forms of elliptic curves, have been presented. The performance of these variants has been compared in terms of computation time and collision rates. It has been observed that the variants utilizing Montgomery and Legendre forms

exhibit faster computation times compared to the original CGL hash function. This improvement can be attributed to the efficient computation of backtracking isogenies on these particular forms, which was a major bottleneck in the original CGL hash function. However, it has also been noted that the variant employing the Legendre form has higher collision rates compared to the variant using the Montgomery form, although still lower than the other two variants. A conjecture has been made that this disparity in collision rates might be influenced by the presence of short cycles in the 2-isogeny sequences on Legendre form curves. Further investigation of this phenomenon is planned, along with the exploration of potential solutions.

### 4.2 Supersingular Isogeny-Based Single Compression Hash Function

In this section, a novel single compression cryptographic hash function is proposed, which is based on traversal in the supersingular isogeny (2-isogeny) graph and point mapping under the isogeny. The proposed function stands out from existing supersingular isogeny-based (SI) hash functions by outputting the $X$-abscissa of a point on a supersingular elliptic curve without revealing the $j$-invariant of the traversed curve. This unique feature enhances the security of the hash function as it introduces two challenging problems for attackers: finding the curve from its $X$-coordinate and solving the supersingular isogeny problem. As a result, the proposed hash function offers improved preimage resistance. The collision-free property of the proposed hash function is ensured by the inherent randomness of mapping two different points (or isogenies) to the same point through an isogeny. Moreover, the compression function of the hash function processes bits from multiple blocks of the message in a single

step, providing it with a computational complexity advantage over other SI hash functions. Additionally, the compression function organically modifies the message to enhance resilience against parallelization attempts. The computational complexity of the proposed hash function is analyzed and compared with that of the CGL and CGL-like hash functions. Furthermore, a comprehensive collision test is conducted, considering different prime fields.

### 4.2.1 Proposed Single Compression Hash Algorithm

In this section, the details of a proposed single compression hash algorithm based on the supersingular isogeny graph $G_l$ where $l = 2$ are presented. The graph $G_2$ is defined by fixing a large prime $p$ as $2^n f - 1$, where $f = 3^{e_3}5^{e_5}7^{e_7}$. The values of $e_3, e_5$, and $e_7$ are chosen such that $3^{e_3}5^{e_5}7^{e_7} - 2^n > 0$. Additionally, the congruence $p \equiv 3 \bmod 4$ is ensured to irreducible polynomial of the field is $x^2 + 1$. Hence, the order of the curve is $(p+1)^2 = (2^n f)^2$. One of the distinctive features of the proposed function is that it does not require sequential compressions for multiple blocks of the input message. Instead, the message is divided into different blocks, and each block is processed with a single application of the compression function. Another significant property of the proposed hash function is the organic modification of the message during the function processing. This characteristic ensures that even if an attacker possesses both the input message and its hash, it remains computationally infeasible for them to make shortcuts or gain computational advantages. Thus, the security of the message is preserved even if an attacker only obtains a portion of it. Similar to CGL, the proposed hash function traverses the isogeny graph starting from

a fixed starting curve $E \in G_2$. Along with the starting curve $E$, a unique starting point $P \in E$ is computed for each message, with the point's order being $3^{e_3}5^{e_5}7^{e_7}$. Backtracking is not allowed in the proposed function, as a result, two isogenies to follow from each elliptic curve. The path to follow is determined based on a uniform convention mapping each bit, 0 or 1, to one of the two non-backtracking isogenies from each curve. The starting point is mapped through the isogeny to the next isogenous elliptic curve, i.e., if $\phi : E \to \tilde{E}$, then $P \in E$ is mapped to $\tilde{E}$ as $\phi(P)$. The order of the mapped point is preserved throughout the function's processing, as the order of the point, $3^{e_3}5^{e_5}7^{e_7}$, is coprime to the order, 2, of the isogeny. Each of the following bits determines the next isogeny dependent on both the current block of the message and the computed image point. As the message is being organically modified using the image point while processing the function, no part of the function is parallelizable. The last mapped point of the function is the output of the hash function. The proposed hash algorithm can be divided into two parts: preprocessing and hash function.

**Preprocessing**

The preprocessing part is responsible for generating a unique starting point from the initial curve based on the input message. It is computed only once in the whole hash computation, and the cost associated with the preprocessing part stays almost constant regardless of the number of blocks. Algorithm-4 demonstrates the preprocessing computation. As the function is prototyped in SageMath [67], the library function *division_points* from SageMath is employed to determine the list of points of order $3^{e_3}, 5^{e_5}, 7^{e_7}$. To simplify the description, the symbols $l_1, l_2, l_3$ are utilized

to represent the values $3, 5, 7$ respectively, and $x_1, x_2, x_3$ are used to represent the values $e_3, e_5, e_7$ respectively. Starting from the point at infinity, which is represented as $(0 : 1 : 0)$ in the projective space, the $l_i$-division points list is computed up to $x_i$ steps, where $i$ takes the values of $1, 2, 3$. At the same time, the message is converted into $x_i$-base so that a unique point can be selected at each step based on the distinct indexing of the list.

**Algorithm 4** Preprocessing of single compression hash function to generate a unique point from the starting elliptic curve

**Input:** Message $M = m_1||m_2||\cdots||m_k$; Starting Elliptic Curve $E$ over $\mathbb{F}_{p^2}$ where $p = 2^n {l_1}^{x_1} {l_2}^{x_2} {l_3}^{x_3} - 1$

**Output:** Point $P$ in elliptic curve $E$

Initialize two points $P$ & $P_1$ point at infinity $(0 : 1 : 0)$ in projective space of curve $E$; $P \leftarrow E[0]$, $P_1 \leftarrow E[0]$
**for** $i = 1$ to $3$ **do**
  $m \leftarrow M\%x_i$
  division points $\leftarrow$ List of arbitrarily sorted points $Q$ based on $x$-axis value so that $l_iQ = P_1$
  $j \leftarrow 2(m\%((l_i^2 - 2)//2)) + 1$
  $P_1 \leftarrow$ point from division point list at index $j$
  **for** $o = 1$ to $x_i - 1$ **do**
    $m \leftarrow m//x_i$
    $m' \leftarrow m\%x_i$
    division points $\leftarrow$ List of arbitrarily sorted points $Q$ based on $x$-axis value so that $l_iQ = P_1$
    $j \leftarrow m_1\%l_i^2$
    $P_1 \leftarrow$ point from division point list at index $j$
  **end for**
  $P \leftarrow P + P_1$
**end for**
**return** Point $P$

In the first step of $x_i$-steps, there are $\frac{l_i^2-1}{2}$ points to select because the list obtained from *division_points* function starts with the point at infinity $(0 : 1 : 0)$, and consecutively lists the two points for each $X$-axis value. As a result, the $X$-axis values in this list are 0, followed by each possible $X$-values repeated twice. However,

in the remaining $x_i - 1$ steps, there would be $l_i^2$ values to choose from since only one $Y$-value appears for each $X$-value in the output of *division_points*. At each step, the order of the points in the list increases as a multiple of $l_i$, and the order of the finally chosen point would be $l_i^{x_i}$. For each $l_i$, all the points are added to get the initial point $P$ from the initial elliptic curve.

In addition, the whole preprocessing computation can be further simplified and sped up by specifying two bases, denoted as $Q$ and $R$, with the order of $E[f]$. This may reduce the initial point space. In this case, $Q$ and $R$ could be the two more additional inputs of the hash algorithm. The initial point $P$ can be computed as $Q + m_1 R$ where $m_1$ is the first block of the message.

**Hash Algorithm**

Let $f(x) = x^2 + 1$ be the irreducible polynomial in the field $F$ where $i$ is the root of $f(x)$. The elements in the field can be represented as $a + bi$ where $a, b \in p$. This concept is analogous to complex numbers, where the imaginary unit $i$ is added to the real numbers. The real and imaginary parts of the element can be referred to as $a$ and $b$, respectively. The real and imaginary parts of the $X$-coordinate of a point on the elliptic curve are used to increase randomness in proposed hash algorithm.

The flowchart described in figure 4.3 showed the main steps of the proposed hash algorithm. The input of the proposed single compression hash algorithm is a supersingular elliptic curve $E$ over $\mathbb{F}_{p^2}$ and message $M$ where the prime of the field $p$ is $2^{e_2}3^{e_3}5^{e_5}7^{e_7} - 1$. Moreover, a point $P = (X_P : Y_P : Z_P) \in E$ in elliptic curve $E$ computed in 4.2.1 is also part of the algorithm. The given message $M$ is divided

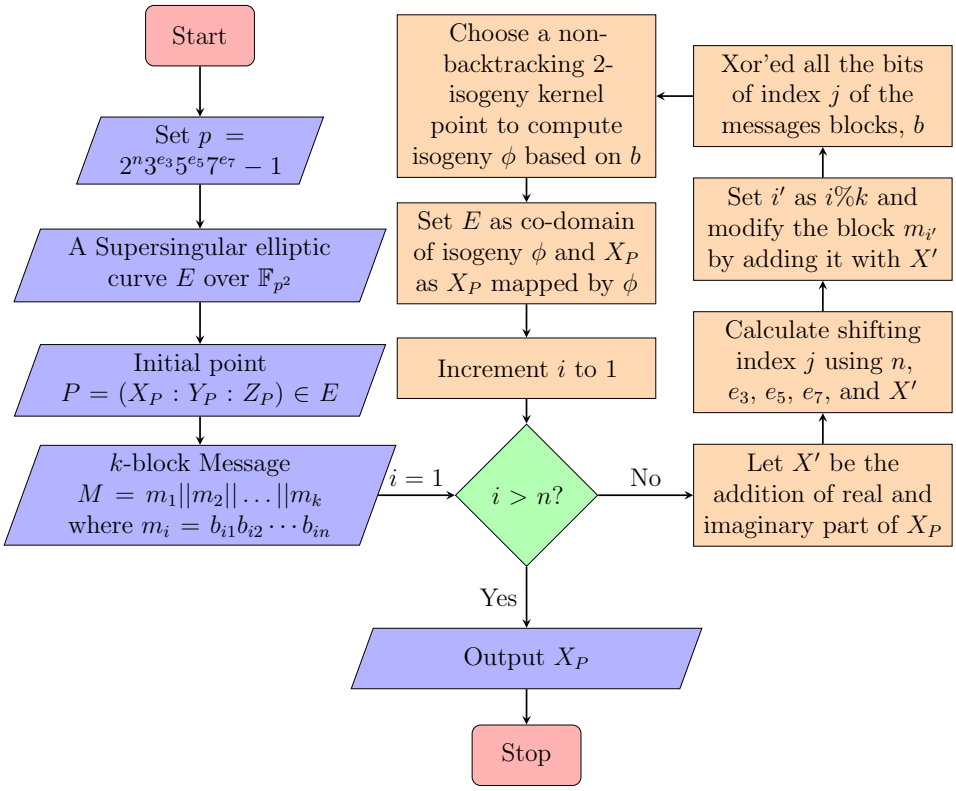



**Figure 4.3:** Flowchart of the steps of proposed hash algorithm

into $k$-different blocks so that message $M$ is the concatenation of $m_1, m_2, \cdots, m_k$ block each consisting of $n$ bits possibly except the last block $m_k$, where $n = e_2$. The proposed hash algorithm will run through $n$ rounds. In each round of the operation, the real and imaginary parts of the $X$-coordinate of the point $P$ are separated, and the arithmetic addition operation is performed to obtain $X'_P$. The value $X'_P$, together with $e_2, e_3, e_5$, and $e_7$, is used to calculate a unique index $j$. Additionally, in each round, $X'_P$ is added modulo $2^n$ to a message block where the block index is calculated as the current round number of operation modulo $k$. Now a bit $b$ is computed by taking the exclusive or (XOR) of the bit in the index $j$ from all message blocks. The bit $b$ is decided on the following path for traversing in the isogeny graph. Similar to the CGL, two non-backtracking path map to bit 0 and 1 through a uniform convention. In the

implementation, the assignment of bits 0 and 1 to the two non-backtracking torsion points is carried out to compute the 2-isogeny, denoted as $\phi$. For the next round of operations, $E$ will be set as the codomain of $\phi$ or the newly computed isogenous elliptic curve, and the $X$-abscissa of the image point $P$ or $\phi(P)$ is computed.

**Algorithm 5** Hash Algorithm

**Input:** Message $M = m_1||m_2||\cdots||m_k$; Montgomery coefficient of Starting Elliptic Curve $E$ over $\mathbb{F}_{p^2}$, $A$ where $p = 2^{e_2}3^{e_3}5^{e_5}7^{e_7} - 1$; Initial point $P = (X_P : Y_P : Z_P) \in E$ is part of the algorithm

**Output:** $X$-abscissa of Point of order $3^{e_3}5^{e_5}7^{e_7}$ in a supersingular elliptic curve

1: **for** $i = 1$ to $e_2$ **do**
2: real, imaginary $\leftarrow X_P$
3: $X'_P \leftarrow$ (real of $X_P$+ imaginary of $X_P$)$\%2^{e_2}$
4: $i' \leftarrow i\%k$
5: $m_i \leftarrow (m_{i'} + X'_P)\%2^{e_2}$
6: $j \leftarrow X'_P\%(|2^{e_2} - 3^{e_3}| + |5^{e_5} - 7^{e_7}|) + 1$
7: $b \leftarrow 0$
8: **for** $l = 1$ to $k$ **do**
9: $b \leftarrow b \bigoplus m_l[j]$
10: **end for**
11: $C \leftarrow \sqrt{A^2 - 4}$
12: $tp_1 \leftarrow -A + C$
13: $tp_2 \leftarrow -A - C$
14: **if** $b \leftarrow 1$ **then**
15: $tp \leftarrow min(tp1, tp2)$
16: **else** $\{b \leftarrow 0\}$
17: $tp \leftarrow max(tp1, tp2)$
18: **end if**
19: $A \leftarrow 2 - tp^2$
20: $X_P \leftarrow (X_P(X_P tp - 2))(2X_P - tp)^{-1}$
21: **end for**
22: **return** $X_P$

An efficient and compact implementation of the proposed hash algorithm showed in Algorithm 5. Step 2 to 10 of the algorithm maintain the randomness of the algorithm. A random number $X'_P$ from $X_P$ ($X$-abscissa of point $P$) is responsible for organically modifying a block in each round of operation (step 5). Moreover, a random

index $j$ computed in step 6 navigates the next isogenous curve in 2-isogeny graph. The results from [65, Corollary 1] are utilized in steps 11 to 20 of the algorithm to expedite the computation process. Specifically let, $(a, b) \in \mathbb{F}_p$ and $E : ay^2 = x^3 + ax^2 + x$ is a Montgomery curve. If there is a point $Q$ in $E$ so that $Q \neq (0, 0)$ and $2Q = O_E$, then

$$\phi : E \to \tilde{E} : by^2 = x^3 + Ax^2 + x = x(x^2 + Ax + 1)$$

$$(x, y) \mapsto (g(x), yg'(x)),$$

with $b = x_Q B$, $a = 2(1 - 2x_Q)^2 = 2 - (-A \pm \sqrt{A^2 - 4})^2$, and $g(x) = x(xx_Q - 1)(x - x_Q)^{-1}$. Furthermore, it has been shown in [65, Corollary 1] that the kernel of the dual of $\phi$ is $\langle(0, 0)\rangle$, which is denoted by $ker(\hat{\phi}) = \langle(0, 0)\rangle$. Steps 14 to 16 ensure a deterministic method to select the torsion point by utilizing the *minimum* and *maximum* operations in SageMath, thereby avoiding the random output of the square root function.

A point on an elliptic curve can be mapped to another point on any isogenous elliptic curve by the corresponding isogeny. Therefore, it can be stated that the image point $\phi(P)$ on $\tilde{E}$ over $\mathbb{F}_{p^2}$ of any point $P$ on $E$ over $\mathbb{F}_{p^2}$ is random under the isogeny mapping $\phi$. These random distribution characteristics of points under isogeny mapping play a significant role in making the proposed hash function collision-resistant and guarantee that the hash function's output is purely random.

### 4.2.2 Computational Problems

In this section, an overview of some of the hard problems that underlie the security and properties of the hash function is presented. Assume that a supersingular elliptic curve $E$ over $\mathbb{F}_{p^2}$ is the input of the hash function and point $P \in E$ is part of

the algorithm. Now $E$ and the generator or basis of the point $P$ are public information. However, as discussed in subsection 4.2.1, the point $P$ is unknown since it depends on the actual message. Moreover, the order of the curve $E$ and point $P$ are also known as it depends on the characteristics of the field $p$. The output of the proposed hash function is the $x$-coordinate of point $P'$ ($x(P')$) from a hidden elliptic curve $E'$. Here $E'$ is an isogenous curve of the known curve $E$, and $P'$ is a mapped point from $P \in E$ under the isogeny. Therefore, an adversary who wants to recover the message from the hash function's output must first solve the following problem.

**Problem 1.** *Given $x(P') \in E'$ over $\mathbb{F}_{p^2}$, with the order of the point $P'$ is trivially given, find $E'$.*

To solve the problem, an attacker must find another supersingular elliptic curve $E''$ over $\mathbb{F}_{p^2}$ so that point $Q \in E''$ with the same $x$-coordinate as $P'$. However, the elliptic curves can not be uniquely determined by their $x$-coordinates. The computational complexity of the problem also depends on the size of the field $p$, the algorithms required to search for the supersingular elliptic curve $E''$, and point $Q$. However, the problem can also be considered as the hardened variant of the elliptic curve discrete logarithm problem (ECDLP) [69], which is known to be computationally infeasible for sufficiently large parameters. As of now, no efficient algorithm is known to solve the ECDLP, making Problem 1 also a hard computational problem.

**Problem 2.** *Given $x(P') \in E'$ over $\mathbb{F}_{p^2}$, find a sequence of 2-isogeny path from starting curve $E$ to the curve $E'$ where $E'$ is unknown.*

An attacker needs to solve the problem 1 to find $E'$ and also needs to solve the supersingular isogeny problem (SIP) [8] to find the isogeny path from $E$ to $E'$. The best-known quantum algorithm for this problem has exponential quantum running time [70, §5]. Suppose an adversary knows the endomorphism ring of starting curve $E$ or can compute using the method described in [17]. The adversary still cannot recover the hidden message unless they solve both problem 1 and problem 2. Because of these two problems, the proposed hash function ensures the property of being resistant to preimage attacks which means finding an input that produces a given output is hard.

### 4.2.3 Computational Cost and Result

#### Computational Cost

The computational cost of the proposed hash algorithm depends primarily on Algorithm 5 since the preprocessing part 4.2.1 to compute the initial point is done only once at the beginning with almost constant complexity. To analyze the computational cost, it is assumed that the message consists of a total of $k$ blocks, with each block being $n$ bits long. From the presented algorithm, it can be observed that the hash function always executes a fixed number of $n$ rounds of operations, irrespective of the number of blocks in the message. By excluding trivial computations such as XOR and addition and assuming a constant runtime for the modulo operation, the computational cost of the algorithm can be analyzed while focusing primarily on the field operations performed in each round. Therefore, the analysis can primarily focus on the field operations performed in each round of the algorithm described in Algorithm 5. These are:

- One square root (SR) operation in step 11

- One inversion (I) operation in step 20
- Two square (S) operations in steps 11 and 19
- Four multiplication (M) operations in steps 20

Table 4.3 presents the computational cost of the proposed hash (SCH), CGL, and a CGL alike (aCGL) hash function [51]. A more detailed analysis of the computational cost for the latter two hash functions can be found in [51, §4.3]. The computation in the table is based on some frequently used assumptions $I \approx 100M$, $S \approx 0.67M$, and $SR = (0.67 \log p + 100)M$. Additionally, it assumes that $n \approx \log p$

**Table 4.3:** Comparison of computational cost

| Algorithm | Computational Cost in $M$ |
|---|---|
| SCH | $n(0.67n + 205.34)M = \Theta(n^2)$ |
| CGL | $kn(5.7n + 110)M = \Theta(kn^2)$ |
| aCGL | $kn(13.5 \log n + 42.4)M = \Theta(kn \log n)$ |

Figure 4.4 illustrates the comparison of the computational cost of the three different hash functions with an assumption of $n = 100$. The proposed SCH hash function has a constant computational cost regardless of the number of blocks in the input. In contrast, the computational cost of the other two hash functions increases linearly as the number of blocks increases.

**Simulation Result**

The implementation and evaluation of the proposed single compression hash function are performed using SageMath [67], which is an open-source and multi-platform mathematical software. The algorithm simulated on a 64 bit processor Intel®Core™ i5-7500 CPU @ 3.40GHz x 4 with Ubuntu 18.04.6 LTS operating

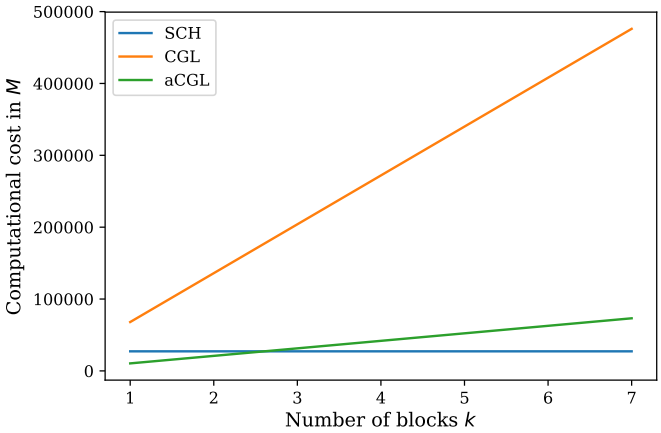



**Figure 4.4:** Comparison of computational cost for hash functions

system. Different prime fields $p$ of the form $2^{e_2}3^{e_3}5^{e_5}7^{e_7}-1$ are selected for the purpose of evaluating the proposed hash function. The total number of collisions is then determined by examining all possible combinations of $e_2$ bits, which represents a single block, for each field. Here collision refers to when two or more bitstream result same point on an elliptic curve starting from a fixed supersingular elliptic curve. The simulation results show no collision for a single block of $e_2$ bits for any of the tested fields. The initial starting curve used in the implementation is the SIKE proposed supersingular elliptic curve with the equation $y^2 = x^3 + 6x^2 + x$. Table 4.4 displays the tested block sizes $n$ and the corresponding orders of initial points $f$, where the prime of the field $p$ is $2^n f - 1$.

### 4.2.4 Conclusion

A new supersingular isogeny-based cryptographic hash function is presented in this study. The hash function comprises two steps: Preprocessing and Hash Algorithm. In the preprocessing step, a point is selected from the initial supersingular elliptic curve. During the hash algorithm process, the chosen point is mapped through a traversal of

**Table 4.4:** Parameter for single block message simulation of the proposed hash function

| Block Size $n$ (Exponent of 2) | Order of initial point $f$ |
|---|---|
| 4 | $3^2 5^1 7^0$ |
| 5 | $3^1 5^1 7^1$ |
| 6 | $3^1 5^1 7^1$ |
| 7 | $3^4 5^1 7^0$ |
| 8 | $3^1 5^1 7^1$ |
| 9 | $3^3 5^1 7^1$ |
| 10 | $3^3 5^2 7^1$ |
| 11 | $3^3 5^3 7^1$ |
| 12 | $3^3 5^1 7^2$ |
| 13 | $3^1 5^3 7^2$ |
| 14 | $3^1 5^3 7^2$ |
| 15 | $3^3 5^2 7^2$ |
| 16 | $3^1 5^4 7^3$ |

a supersingular isogeny graph based on the message until reaching the final traversed curve. Furthermore, the compact algorithm of the hash function is demonstrated by leveraging the fixed backtracking isogeny properties of the Montgomery curve. The proposed hash function shows a constant computational complexity irrespective of the number of message blocks. It is worth noting that, at present, no efficient quantum algorithm has been developed to solve the underlying computational problem, rendering the hash function resistant to quantum attacks. Consequently, the proposed hash function can be considered quantum secure.

# CHAPTER 5

# CONCLUSIONS AND FUTURE WORKS

## 5.1 Conclusions

The integration of enhanced autonomy and security within computing systems has the potential to bring about revolutionary changes in various aspects. By adopting decentralized architectures like Blockchain, computing systems can reduce dependence on centralized authorities and foster trust through peer-to-peer interactions. This shift towards decentralization enables greater transparency, immutability, and resilience in data storage and transactions. Moreover, utilizing distributed networks and redundant storage in these decentralized systems enhances their resilience and availability. Even in the face of disruptions or attacks, the distributed nature of the system ensures uninterrupted operation and mitigates the risks associated with single points of failure.

In light of the emerging threat landscape presented by quantum computing, the incorporation of quantum secure hash becomes crucial for future computing systems. Quantum secure hash functions play a significant role in enhancing the security and reliability of data and communications in the presence of quantum adversaries. These hash functions enable the development of interoperable and compatible solutions with existing cryptographic standards and protocols that rely on hash functions, ensuring a seamless transition to quantum-secure systems.

This dissertation explores two crucial tools which hold great potential for enhancing autonomy and security in future computing systems: Blockchain consensus mechanisms and supersingular isogeny hash functions. The first topic proposes two mobile-friendly consensus mechanisms which can effectively address the root causes of various attacks in Blockchain, such as centralization in mining, forking in the chain, and external attacks like DDoS and DNS attacks. The Proof of Sincerity consensus method tackles the issues of monopolization and limited user participation in Blockchain systems by combining it with existing proof of work schemes. Additionally, a Blockchain consensus mechanism for Blockchain storage was introduced, ensuring decentralized system operation and increased user participation through the probabilistically fair rotation for selecting validators and dividing blocks into multiple sections. These consensus mechanisms are designed to increase user participation in the network, promoting decentralization and preventing a single group of users from controlling the entire network. The implementation of such consensus mechanisms can significantly contribute to the development of future autonomous computing systems.

In the study of supersingular isogeny hash, four compact implementations of the CGL hash function based on 2-isogeny sequences were presented. The Legendre and Montgomery forms demonstrated the best computation time by effectively utilizing fixed backtracking isogeny properties, overcoming the performance bottleneck of the standard CGL function. Furthermore, a new cryptographic hash function based on supersingular isogenies was introduced. It involves a preprocessing step where a point is selected from the initial supersingular elliptic curve and a hash algorithm that maps the chosen point through a supersingular isogeny graph based on the message.

The resulting hash function exhibits resistance to quantum attacks due to the lack of efficient quantum algorithms to solve the underlying computational problem, ensuring its security in the face of potential quantum advancements.

## 5.2 Future Works

### 5.2.1 Study on Legendre Curve

From the given definition 4.1.3 of a Legendre form of an elliptic curve, expressed as $E_\lambda : y^2 = x(x-1)(x-\lambda)$, it can be observed that there is a distinct relationship among the Legendre coefficients. Specifically, any of the following Legendre coefficients: $\lambda, \lambda^{-1}, (1-\lambda), (1-\lambda)^{-1}, \lambda(\lambda-1)^{-1}, (\lambda-1)\lambda^{-1}$ will yield an isomorphic curve $E$. This characteristic is unique to the Legendre form, as no other form of an elliptic curve exhibits such a simple relationship among the coefficients of isomorphic curves. It is worth noting that the existing literature needs more detailed information on how these Legendre coefficients are interrelated in the context of 2-isogenous curves.

As a result, one promising future direction is to conduct an in-depth study of these relationships and leverage this knowledge to simplify the computation of non-backtracking isogenies. If efficient relations can be established, the outcomes can be utilized to design more efficient encryption and decryption algorithms. By exploiting the inherent connections among the Legendre coefficients, it may be possible to optimize cryptographic operations and enhance the overall efficiency and security of isogeny-based elliptic curve systems.

### 5.2.2 Post-Quantum Blockchain

The combination of the topics discussed in this dissertation holds potential for an intriguing application: post-quantum Blockchain. In Blockchain, various cryptographic tools are employed to ensure the system's security and integrity. One such tool is the hash function, which serves multiple purposes, such as generating unique identifiers for blocks, verifying data integrity, and creating digital signatures. Additionally, digital signatures based on public key cryptography, also known as asymmetric cryptography, are utilized in Blockchain to authenticate and ensure the integrity of digital messages and transactions. This involves using a private key to generate a signature and a corresponding public key to verify it. However, the rise of quantum computers poses a threat to these cryptographic tools. Quantum computers have the potential to break the underlying algorithms that secure hash functions and public key cryptography. As a result, there is a need to explore future research directions that seamlessly integrate post-quantum cryptography (PQC) tools into the existing Blockchain architecture. This would involve investigating new cryptographic algorithms and protocols resistant to quantum attacks while maintaining the desirable properties of security, efficiency, and scalability. By addressing the challenge of post-quantum security in Blockchain, we can ensure this transformative technology's continued resilience and long-term viability.

Some of the Blockchain consensus mechanisms, such as proof of time and proof of space-time, employ a cryptographic tool called a verifiable delay function (VDF). The purpose of a VDF is to ensure that participants in the network have actively participated for a certain amount of time in the consensus process or within the

network. A VDF can be defined as a function that requires a specified number of sequential steps to evaluate yet produces a unique output that can be efficiently and publicly verified [71]. The VDF algorithm consists of three main steps. Firstly, the setup phase generates a pair of public parameters, namely the evaluation key (ek) and the verification key (vk), based on the security and puzzle difficulty parameters. Secondly, the evaluation phase takes an input element and the evaluation key to produce an output element along with an optional proof. Finally, the verification phase determines the correctness of the evaluation output by using the verification key, input, output, and proof.

However, similar to other cryptographic schemes, VDFs are vulnerable to attacks by quantum computers. To address this, isogeny-based VDFs have emerged as a promising option in post-quantum cryptography due to their potential resistance against quantum attacks. One specific isogeny-based VDF [72] has been proposed, demonstrating partial quantum resistance. However, this algorithm requires a trusted setup process to generate the initial parameters.

The setup process involves selecting a prime number, a difficulty parameter, a supersingular elliptic curve, and a generator point. Additionally, a random non-backtracking walk of a specified length is computed as a composition of 2-isogenies. The dual of this walk is also computed, along with the image of the generator point. The resulting public parameters are the evaluation key and the necessary curve-related information. In the evaluation process, a point from the target curve is chosen, and the output of the process is the evaluation of this point using the computed 2-isogeny.

The verification process involves computing two Weil pairings [73] and checking their equality.

The public parameter of the VDF reveals the initial and final curves and the image of a point on the graph. If an adversary can break the SIDH problem using this information, as shown in [9–11], they can also break the VDF. Moreover, if the endomorphism ring of the initial curve $E$ is known or computable, the adversary can exploit it to find shortcuts on the graph. In light of these vulnerabilities, future research directions aim to identify and address the weaknesses of SIDH-inspired VDF algorithms by constructing quantum-secure VDF alternatives.

# BIBLIOGRAPHY


[1] Lier, B. Blockchain technology: The autonomy and self-organisation of cyber-physical systems. *Business Transformation Through Blockchain: Volume I*. pp. 145-167 (2019)

[2] Queralta, J., Qingqing, L., Zou, Z. & Westerlund, T. Enhancing Autonomy with Blockchain and Multi-Access Edge Computing in Distributed Robotic Systems. *2020 Fifth International Conference On Fog And Mobile Edge Computing (FMEC)*. pp. 180-187 (2020)

[3] Zhang, H., Leng, S., Wu, F. & Chai, H. A DAG Blockchain-Enhanced User-Autonomy Spectrum Sharing Framework for 6G-Enabled IoT. *IEEE Internet Of Things Journal*. **9**, 8012-8023 (2022)

[4] P. W. Shor, Algorithms for quantum computation: discrete logarithms and factoring, *Proceedings of the 35th Annual Symposium on Foundations of Computer Science, Santa Fe, NM*, (1994).

[5] Webber, M., Elfving, V., Weidt, S. & Hensinger, W. The impact of hardware specifications on reaching quantum advantage in the fault tolerant regime. *AVS Quantum Science*. **4**, 013801 (2022)

[6] Sciences, E., Medicine & Others Quantum computing: progress and prospects. (National Academies Press,2019)

[7] Jao, D., Azarderakhsh, R., Campagna, M., Costello, C., Feo, L., Hess, B., Jalali, A., Koziel, B., LaMacchia, B., Longa, P., Hutchinson, A. & Others Supersingular isogeny key encapsulation. *Submission To The NIST Post-Quantum Standardization Project*. **152** pp. 154-155 (2017)

[8] Feo, L., Jao, D. & Plût, J. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. *Journal Of Mathematical Cryptology*. **8**, 209-247 (2014), https://doi.org/10.1515/jmc-2012-0015

[9] Castryck, W. & Decru, T. An Efficient Key Recovery Attack on SIDH. *Advances*

*In Cryptology – EUROCRYPT 2023*. pp. 423-447 (2023)

[10] Maino, L. & Martindale, C. An attack on SIDH with arbitrary starting curve. (Cryptology ePrint Archive, Paper 2022/1026,2022), https://eprint.iacr.org/2022/1026, https://eprint.iacr.org/2022/1026

[11] Robert, D. Breaking SIDH in polynomial time. (Cryptology ePrint Archive, Paper 2022/1038,2022), https://eprint.iacr.org/2022/1038, https://eprint.iacr.org/2022/1038

[12] Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. *Cryptography Mailing List At Https://metzdowd.com*. (2009,3)

[13] Brassard, G., HØyer, P. & Tapp, A. Quantum cryptanalysis of hash and claw-free functions. *LATIN'98: Theoretical Informatics*. pp. 163-169 (1998),

[14] Bernstein, D. Cost analysis of hash collisions : will quantum computers make SHARCS obsolete?. (2009)

[15] Charles, D., Lauter, K. & Goren, E. Cryptographic Hash Functions from Expander Graphs. *Journal Of Cryptology*. **22** pp. 93-113 (2008,12)

[16] Booher, J., Bowden, R., Doliskani, J., Fouotsa, T., Galbraith, S., Kunzweiler, S., Merz, S., Petit, C., Smith, B., Stange, K., Ti, Y., Vincent, C., Voloch, J., Weitkämper, C. & Zobernig, L. Failing to hash into supersingular isogeny graphs. (2022,4), 31 pages, 7 figures

[17] Eisentraeger, K., Hallgren, S., Leonardi, C., Morrison, T. & Park, J. Computing endomorphism rings of supersingular elliptic curves and connections to pathfinding in isogeny graphs. (2020)

[18] Dwork, C. & Naor, M. Pricing via Processing or Combatting Junk Mail. *Proceedings Of The 12th Annual International Cryptology Conference On Advances In Cryptology*. pp. 139-147 (1993), http://dl.acm.org/citation.cfm?id=646757.705669

[19] Lilly, G. Device for and method of one-way cryptographic hashing. (2004)

[20] Möser, M., Böhme, R. & Breuker, D. An inquiry into money laundering tools in the Bitcoin ecosystem. *2013 APWG ECrime Researchers Summit*. pp. 1-14

(2013)

[21] O'Dwyer, K. & Malone, D. Bitcoin mining and its energy footprint. *25th IET Irish Signals Systems Conference 2014 And 2014 China-Ireland International Conference On Information And Communications Technologies (ISSC 2014/CIICT 2014)*. pp. 280-285 (2014,6)

[22] Taylor, M. Bitcoin and the age of Bespoke Silicon. *2013 International Conference On Compilers, Architecture And Synthesis For Embedded Systems (CASES)*. pp. 1-10 (2013)

[23] Karame, G., Androulaki, E. & Capkun, S. Double-spending Fast Payments in Bitcoin. *Proceedings Of The 2012 ACM Conference On Computer And Communications Security*. pp. 906-917 (2012), http://doi.acm.org/10.1145/2382196.2382292

[24] Narayanan, A., Bonneau, N., Felten, E., Miller, A. & Goldreich, O. Bitcoin and Cryptocurrency Security. *IEEE Security & Privacy*. (2016)

[25] King, S. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *Self-published Paper*. (2012)

[26] Bentov, I., Lee, C. & Mizrahi, A. Cryptocurrencies Without Proof of Work. *Proceedings Of The 2016 ACM SIGSAC Conference On Computer And Communications Security*. pp. 1428-1440 (2016)

[27] Larimer, D. Delegated proof-of-stake. *White Paper*. (2014), https://www.bitsharestalk.org/index.php/board,4.0.html

[28] Bentov, I., Lee, C., Mizrahi, A. & Rosenfeld, M. Proof of Activity: Extending Bitcoin's Proof of Work via Proof of Stake [Extended Abstract]Y. *SIGMETRICS Perform. Eval. Rev.*. **42**, 34-37 (2014,12), http://doi.acm.org/10.1145/2695533.2695545

[29] Dziembowski, S., Faust, S., Kolmogorov, V. & Pietrzak, K. Proofs of Space. (Cryptology ePrint Archive, Paper 2013/796,2013), https://eprint.iacr.org/2013/796, https://eprint.iacr.org/2013/796

[30] Pouwelse, J., Garbacki, P., Epema, D. & Sips, H. The Bittorrent P2P File-Sharing System: Measurements and Analysis. *Peer-to-Peer Systems IV*. pp. 205-216 (2005)

[31] Wilkinson, S. Storj A Peer-to-Peer Cloud Storage Network. (2014)

[32] Maymounkov, P. & Mazières, D. Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. *Revised Papers From The First International Workshop On Peer-to-Peer Systems*. pp. 53-65 (2002), http://dl.acm.org/citation.cfm?id=646334.687801

[33] Vorick, D. & Champine, L. Sia: Simple Decentralized Storage. (2014)

[34] Kamara, S. Proofs of Storage: Theory, Constructions and Applications. *Algebraic Informatics*. pp. 7-8 (2013)

[35] Benet, J. IPFS - Content Addressed, Versioned, P2P File System. *CoRR*. **abs/1407.3561** (2014), http://arxiv.org/abs/1407.3561

[36] Labs, P. Filecoin: A Decentralized Storage Network. (2017)

[37] Lara-Nino, C., Díaz-Pérez, A. & Morales-Sandoval, M. Elliptic Curve Lightweight Cryptography: A Survey. *IEEE Access*. **PP** pp. 1-1 (2018,11)

[38] Silverman, J. Heights and elliptic curves. *Arithmetic Geometry*. pp. 253-265 (1986)

[39] Childs, A., Jao, D. & Soukharev, V. Constructing elliptic curve isogenies in quantum subexponential time. *Journal Of Mathematical Cryptology*. **8**, 1-29 (2014), https://doi.org/10.1515/jmc-2012-0016

[40] Galbraith, S. & Vercauteren, F. Computational problems in supersingular elliptic curve isogenies. (Cryptology ePrint Archive, Paper 2017/774,2017), https://eprint.iacr.org/2017/774

[41] Vélu, J. Isogénies entre courbes elliptiques. *Comptes-Rendus De L'Académie Des Sciences, Série I*. **273** pp. 238-241 (1971)

[42] Mestre, J. La méthode des graphes. Exemples et applications. *Proceedings Of The International Conference On Class Numbers And Fundamental Units Of Algebraic Number Fields (Katata)*. pp. 217-242 (1986)

[43] Pizer, A. Ramanujan graphs and Hecke operators. *Bulletin Of The American Mathematical Society*. **23**, 127-137 (1990)

[44] Pizer, A. Ramanujan graphs. Computational perspectives on number theory (Chicago, IL, 1995), 159–178. *AMS/IP Stud. Adv. Math.* **7**

[45] Lubotzky, A., Phillips, R. & Sarnak, P. Ramanujan graphs. *Combinatorica.* **8**, 261-277 (1988)

[46] Costache, A., Feigon, B., Lauter, K., Massierer, M. & Puskás, A. Ramanujan Graphs in Cryptography. *Research Directions In Number Theory.* pp. 1-40 (2019)

[47] Zaman, M., Hutchinson, A. & Min, M. Implementation Aspects of Supersingular Isogeny-Based Cryptographic Hash Function. *Wireless Internet: 15th EAI International Conference, WiCON 2022, Virtual Event, November 2022, Proceedings.* pp. 14-27 (2023)

[48] Goldreich, O. Candidate one-way functions based on expander graphs. *Studies In Complexity And Cryptography. Miscellanea On The Interplay Between Randomness And Computation.* pp. 76-87 (2011)

[49] Lauter, K., Charles, D. & Goren, E. Pseudorandom number generation with expander graphs. (Google Patents,2011), US Patent 7,907,726

[50] Yoshida, R. & Takashima, K. Simple Algorithms for Computing a Sequence of 2-Isogenies. *Information Security And Cryptology - ICISC 2008, 11th International Conference, Seoul, Korea, December 3-5, 2008, Revised Selected Papers.* **5461** pp. 52-65 (2008), https://doi.org/10.1007/978-3-642-00730-9

[51] Doliskani, J., Pereira, G. & Barreto, P. Faster Cryptographic Hash Function From Supersingular Isogeny Graphs. *IACR Cryptol. EPrint Arch..* **2017** pp. 1202 (2017)

[52] Panny, L. Isogeny-based hashing despite known endomorphisms. *IACR Cryptol. EPrint Arch..* **2019** pp. 927 (2019)

[53] Tachibana, H., Takashima, K. & Takagi, T. Constructing an efficient hash function from 3-isogenies. *JSIAM Lett..* **9** pp. 29-32 (2017)

[54] Zaman, M., Shen, T. & Min, M. Proof of Sincerity: A New Lightweight Consensus Approach for Mobile Blockchains. *2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC).* pp. 1-4 (2019)

[55] Uz Zaman, M. & Min, M. Decentrally-Consented-Server-Based Blockchain System for Universal Types of Data. *2020 International Symposium On Networks, Computers And Communications (ISNCC)*. pp. 1-6 (2020)

[56] Beikverdi, A. & Song, J. Trend of centralization in Bitcoin's distributed network. *2015 IEEE/ACIS 16th International Conference On Software Engineering, Artificial Intelligence, Networking And Parallel/Distributed Computing (SNPD)*. pp. 1-6 (2015,6)

[57] Barkatullah, J. & Hanke, T. Goldstrike 1: CoinTerra's First-Generation Cryptocurrency Mining Processor for Bitcoin. *IEEE Micro*. **35**, 68-76 (2015,3)

[58] Nash, J. Equilibrium points in n-person games. *Proceedings Of The National Academy Of Sciences*. **36**, 48-49 (1950), https://www.pnas.org/content/36/1/48

[59] Reny, P., Osborne, M. & Rubinstein, A. A Course in Game Theory. (1994)

[60] Mckelvey, R., Mclennan, A. & Turocy, T. Gambit: Software tools for game theory, Version 16.0.1.. (2016), http://www.gambit-project.org

[61] Govindan, S. & Wilson, R. A global Newton method to compute Nash equilibria. *Journal Of Economic Theory*. **110** pp. 65-86 (2003,5)

[62] Cassandras, C. & Lafortune, S. Introduction to Discrete Event Systems. (Springer Publishing Company, Incorporated,2010)

[63] Leon-Garcia, A. Probability, Statistics, and Random Processes for Electrical Engineering. (Pearson/Prentice Hall,2008)

[64] Sutherland, A. Lecture 6: Isogeny kernels and division polynomials. *Elliptic Curves—MIT Course No. 18.783*. (2019), https://math.mit.edu/classes/18.783/2019/LectureNotes6.pdf, MIT OpenCourseWare

[65] Renes, J. Computing Isogenies Between Montgomery Curves Using the Action of (0, 0). *Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018, Fort Lauderdale, FL, USA, April 9-11, 2018, Proceedings*. **10786** pp. 229-247 (2018), https://doi.org/10.1007/978-3-319-79063-3

[66] Elliott, J. & Hutchinson, A. Supersingular Isogeny Diffie-Hellman with Legendre Form. (Cryptology ePrint Archive, Paper 2022/870,2022), https://eprint.iacr.org/2022/870, https://eprint.iacr.org/2022/870

[67] Stein, W. & Others Sage Mathematics Software (Version 9.4.0). (The Sage Development Team,2021), http://www.sagemath.org

[68] Silverman, J. Arithmetic of elliptic curves. *Springer-Verlag*. pp. 39-48 (1994), Chapter 3, Section 3

[69] Koblitz, N. Elliptic curve cryptosystems. *Mathematics Of Computation*. **48**, 203-209 (1987)

[70] Jao, D. & Feo, L. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. *International Workshop On Post-Quantum Cryptography*. pp. 19-34 (2011)

[71] Boneh, D., Bonneau, J., Bünz, B. & Fisch, B. Verifiable Delay Functions. *Advances In Cryptology – CRYPTO 2018*. pp. 757-788 (2018)

[72] De Feo, L., Masson, S., Petit, C. & Sanso, A. Verifiable Delay Functions from Supersingular Isogenies and Pairings. *Advances In Cryptology – ASIACRYPT 2019*. pp. 248-277 (2019)

[73] Miller, V. The Weil pairing, and its efficient calculation. *Journal Of Cryptology*. **17**, 235-261 (2004)