DESIGN, MODELING, AND SIMULATION OF SECURE X.509 CERTIFICATE REVOCATION

By

Sri Naga Sai Abhijith Medury

Anthony Skjellum
Professor of Computer Science
University of Tennessee at Chattanooga
(Advisor)

Amani Altarawneh
Assistant Professor of Computer Science
Colorado State University
(Committee Member)

Richard R. Brooks
Professor of Electrical &
Computer Engineering
Clemson University
(Committee Member)

Farah Kandah
Associate Professor of Computer Science
Auburn University
(Committee Member)

Donald R. Reising
Assistant Professor of Electrical Engineering
University of Tennessee at Chattanooga
(Committee Member)

DESIGN, MODELING, AND SIMULATION OF SECURE X.509 CERTIFICATE REVOCATION

By

Sri Naga Sai Abhijith Medury

A Dissertation Submitted to the Faculty of the University of
Tennessee at Chattanooga in Partial
Fulfillment of the Requirements of the Degree of
Doctor of Philosophy in Computational Science

The University of Tennessee at Chattanooga
Chattanooga, Tennessee

May 2023

ABSTRACT

TLS communication over the internet has risen rapidly in the last seven years (2015–2022), and there were over 156M active SSL certificates in 2022. The state-of-the-art Public Key Infrastructure (PKI), encompassing protocols, computational resources, and digital certificates, has evolved for 24 years to become the de-facto choice for encrypted communication over the Internet even on newer platforms such as mobile devices and Internet-of-Things (IoT) (despite being low powered with computational constraints). However, certificate revocation is one sub-protocol in TLS communication that fails to meet the rising scalability demands and remains open to exploitation.

In this dissertation, the standard for X.509 revocation is systematically reviewed and critically evaluated to identify its limitations and assess their impact on internet security. Because of fragmented revocation information and limited scalability, even the latest version of the X.509 revocation standard is susceptible to Man-in-the-Middle (MiTM) attacks. Blockchain technology can provide a decentralized and peer-to-peer distributed ledger to enable a unified, tamper-proof platform for X.509 certificate authorities to collaborate securely in a trustless environment. To understand blockchain technology's capabilities and limitations in distributing X.509 revocation information, different blockchain platforms are explored and compared in terms of scalability, degree of decentralization, and cost of operation. Moreover, the unification of the revocation lists leads to a massive expansion in the number of revoked certificates to query by a verifying

client thus increasing latency during revocation lookup. And, to minimize revocation-status lookup times, cryptographic constructions and approximate set-membership data structures are prototyped and analyzed.

The key contributions of this dissertation are twofold: 1) the novel design of a secure and robust system for distributing X.509 certificate revocation information; and, 2) the prototype, experimentation, and optimization of cascading XOR filter, fuse filter, and cuckoo filter for quick lookup with zero false positives (and zero false negatives). The Secure Certificate Revocation as a Peer Service (SCRaaPS) is designed using the Lightweight Mining consensus algorithm-based Scrybe blockchain protocol to store and distribute certificate revocation lists. And, the cascading fuse filter (demonstrating the highest space efficiency and fastest build time) is applied to minimize the revocation lookup time with zero false positives.

DEDICATION

This work is dedicated to the brilliant minds that introduced blockchain technology to the world. It is also dedicated to this country's scientists and entrepreneurs that are diligently advancing research in decentralized computing and secure internet communication. I wish them success and I hope this work contributes towards realizing the vision of Web 3.0.

In loving memory of Mahinder Reddy.

ACKNOWLEDGMENTS

Research reported in this dissertation was supported in part by the National Science Foundation under grants nos. 1821926, 1925603, and 1918987. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

I'm extremely grateful to my advisor, Dr. Anthony Skjellum, for all the support and guidance throughout this journey. He taught me how to precisely articulate complex problems and conceptualize solutions impactfully. His technical advice helped me learn how to tackle research challenges and his philosophical advice helped advance my critical thinking. I deeply appreciate him for this opportunity to apply and practice the scientific method. His mentorship and this journey transformed me into the scientist I always hoped to become.

Special thanks to all committee members for their valuable advice and feedback. Thank you, Dr. Richard Brooks, for helping us navigate the complexities of fault tolerance in distributed computing. I will always remember this key piece of wisdom from him – "*If encryption is the solution to your problem, then you are simply shifting the problem to key distribution*". Thank you, Dr. Farah Kandah, for teaching us how to critically evaluate a system and identify threats, especially insider threats that are often obscure and overlooked. Thank you, Dr. Donald Reising, for helping me distinguish practical limits from buzz-fueled limits to hardware and for helping me address the challenges with network connectivity and bandwidth in distributed computing.

Thank you, Dr. Amani Altarawneh, for your technical critique and feedback that helped me fortify this work and draw the right conclusions. Additionally, I'm grateful for your camaraderie, and I appreciate you for being there to share moments of success and for the cheerful pick-me-ups during moments of disappointment. Also, many thanks to Bailey Kirby for reviewing and editing parts of this document. And special thanks to Katie Culpepper for proofreading and grammatical review. Further, thanks to the co-authors – Brennan Huber, Conner Fulford, and others – of my research publications during this PhD.

I'm also grateful to all my colleagues at SimCenter, University of Tennessee at Chattanooga. Special thanks to Kim Sapp for all the guidance and help during this journey. I would not have finished without her help. Many thanks to Dr. Ethan Hereth for introducing me to Linux System Engineering, for helping me gain a strong understanding of Linux topics and high-performance computing (HPC) infrastructure, and for introducing me to the Vim editor. I especially appreciate all the Vim editing skills I gained from shadowing Ethan. And, thanks to Dr. Michael Ward, mainly for all the food and things he gave me during this tenure, but also for helping me improve my Linux System Administration skills and for supervising my efforts. And, thanks to Dr. Craig Tanis, Dr. Mina Sartipi, and the Computer Science and Engineering department faculty at the University of Tennessee at Chattanooga, for introducing me to advanced technical concepts including Parallel Programming, HPC, and Deep Learning. Also, thanks to my PhD cohorts: Dr. Noman Saed, Dr. Chang Phuong, Dr. Maxwell Omwenga, Dr. Mehran Ghafari, Dr. James Trimble, and others, for your camaraderie and companionship that made this journey unique.

I would also like to thank my family members for their monetary and moral support. Thanks to my father, Murali Krishna, for encouraging me to take on this journey and for listening to me ramble about computer science and philosophical topics during our conversations. Thank you, mother (Surekha Ashtaputra), for the consistent check-ins to keep me on track and for the encouragement to conclude the journey. Thank you, brother (Lalith Medury), for your moral support and encouragement.

GLOSSARY

- **X.509** – The X.509 is a standard for formatting public cryptographic key certificates by the International Telecommunication Union. It is a popular standard used in HTTPS and TLS/SSL protocols and is implemented by many Internet Certificate Authorities. The X.509 standard through certificate formatting defines the operations of Certificate Authorities and the procedures to follow (such as revocation list distribution and revocation status checking) during trust establishment. This standard was initially published in 1988 and has undergone three major version increments (v1–v3) and the latest updates to this standard were published in May 2018.

- **CRL** – Certificate Revocation List was one of two procedures defined by the X.509 standard for the purpose of revocation information distribution and revocation status checking. CRLs are lists of recently revoked certificates (that haven't yet expired) by the Certificate Authority. CRLs are published periodically in certificate distribution points mentioned in the X.509 certificate. Browser clients are expected to identify distribution points from the visiting website's X.509 certificate, download the CRL from these distribution points and parse them to check the revocation status of a certificate.

- **OCSP** – Online Certificate Status Protocol is the other procedure defined by the X.509 standard for the purpose of distributing revocation information and revocation status checking. The OCSP procedure defines the operation of OCSP responders (that must be provisioned and maintained by CAs) that will respond to browser clients' requests for the revocation status of a certificate. OCSP responder URLs must be mentioned in the X.509 certificate issued by a CA.

- **Cryptographic accumulator** – The cryptographic accumulator is a cryptographic construct introduced by Benaloh and de Mare in 1993 that involves a one-way membership hash function. Digital signatures involving public key cryptography can be accumulated into a constant-size representation—the cryptographic accumulator—using a one-way membership hash function that displays quasi-commutative properties. The resulting accumulator certifies the membership of candidate public keys without revealing the individual members. And candidate members can prove membership by producing a corresponding witness to the accumulator.

- **Probabilistic data structure** – Data structures that provide approximate responses to queries about a (large) data set. They are constant-sized representations of data sets in the form of a bit array or a hash table and can be used to verify the membership of arbitrary elements in constant computational time. Bloom filter, cuckoo filter, and XOR filter are examples of probabilistic data structures.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

- **ACL** – Access Control Lists
- **CA** – Certificate Authority
- **CCF** – Cascading cuckoo filter
- **CFF** – Cascading (binary) fuse filter
- **CRL** – Certificate Revocation Lists (X.509 standard)
- **CT** – Certificate Transparency
- **CXF** – Cascading XOR filter
- **FP** – False Positive (result of set-membership test)
- **LWM** – Lightweight Mining Algorithm
- **MiTM** – Man in the Middle (system attack)
- **OCSP** – Online Certificate Status Protocol (X.509 standard)
- **PGP** – Pretty Good Privacy (encryption program)
- **PKI** – Public Key Infrastructure
- **PoS** – Proof of Stake (blockchain consensus algorithm)
- **PoW** – Proof of Work (blockchain consensus algorithm)
- **PURL** – Persistent Uniform Resource Locator
- **SCRaaPS** – Secure Certificate Revocation as a Peer Service
- **TLS** – Transport Layer Security

CHAPTER 1

INTRODUCTION

In this chapter, the problem space of X.509 certificates is introduced, and the challenges involved with current revocation mechanisms that motivated this research work are described. We then present our approach and describe the expected outcomes. Next, we describe our experimentation plan for evaluating the proposed solution to analyze and compare with present systems. Later, we describe the originality of our idea, the potential impact of this research work, and the expected contribution to literature. Finally, we provide an overview of this research dissertation document.

1.1 Motivation

The X.509 standard [1] of certificate management for establishing trust over the internet is hierarchical and centralized. The original revocation methodology evolved from distributing Certificate Revocation Lists (CRLs) [2] to stapling certificate revocation status in TCP-handshake [3] (i.e. OCSP-must-staple) to overcome security and performance challenges. However, due to fragmented revocation information [4] and limited scalability [5] even the latest version of the X.509 Revocation standard is susceptible to Man-in-the-Middle (MiTM) attacks [6]. Implementing reliable revocation methodologies is critical in securing the billions of devices across different mobile computing platforms (smartphones, IoT, autonomous vehicles) that will be connected to the internet in the next few years.

The Secure Certificate Revocation as a Peer Service (SCRaaPS) [7] designed in this research provides robust and reliable validation of X.509 certificate revocation. SCRaaPS addresses security threats to X.509 certificate revocation process by providing a trustworthy

platform to unify certificate revocation information over an untrusted public network. SCRaaPS achieves this by utilizing a blockchain ledger for the collection and distribution of revocation information. Further, the blockchain ledger ensures a consistent global state of revocation information through immutable, append-only logs of activity that are open for public auditing.

Additionally, the cascading technique is applied to different probabilistic data structures to analyze the potential of building a probabilistic data structure that produces zero false positives. The novel cascading cuckoo filter designed in this research produces a constant-time lookup of arbitrary elements in large lists with zero-false positives. The cascading cuckoo filter helps minimize the latency of revocation status lookup and is further applicable to other areas of cybersecurity such as blocklists in ACLs.

1.2 Evaluation Metrics

The X.509 certificate revocation solution enabled by SCRaaPS is compared with state-of-the-art revocation standard (OCSP-must staple [3]) and other emerging solutions (such as CRLLite [8] by Mozilla Foundation) in terms of security, scalability, and performance.

The following key properties of revocation data must be ensured to enable reliable revocation status validation:

- *Authenticity*: X.509 certificate revocation lists must be published by the corresponding Certificate Authority (CA).
- *Integrity*: The revocation lists published by Certificate Authorities must remain untampered during revocation status lookup.
- *Availability*: The revocation lists' information must remain highly available for users to access and look up the revocation status of arbitrary certificates.
- *Non-Repudiation*: Certificate Authorities are accountable for publishing revocation information and in order for it to be trustworthy, the authorship of the published revocation list must be indisputable.

To analyze the security of SCRaaPS and its ability in maintaining the above data properties, a threat model was designed to outline known potential security threats to current revocation methodologies. The SCRaaPS design was then examined against these threats to evaluate the security of the solution. Further, the Scrybe blockchain ledger presents new threat surfaces that were investigated and mitigated.

SCRaaPS enables higher scalability in the certificate revocation procedure through the peer-to-peer distributed ledger and probabilistic data structure based on constant-time lookup. The scalability of the solution is measured and compared in terms of the number of active participants (Certificate Authorities, Certificate Owners, and Users) that are sustainable while maintaining baseline performance in revocation info distribution and lookup.

The performance of the SCRaaPS revocation approach is measured and compared in terms of bandwidth and latency. The bandwidth of revocation methodologies is derived from the total number of certificate revocations in the past period that can be meaningfully propagated across the network without causing an uptick in latency. Bandwidth is recorded in terms of the number of revocations in unit time. This is a key performance metric that can significantly impact the latency of revocation status verification. The aftermath of the Heartbleed bug discovery which led to the revocation of over 50,000 certificates [9] resulted in high latency TCP handshakes, which slowed down a majority of the internet connections for over 48 hours.

Latency in certificate revocation methodologies was evaluated and compared in terms of the average duration of propagating the certificate revocation lists to a majority of the network. Other latency metrics, such as total time for revocation status lookup or TCP handshake, are also captured and compared but the design of SCRaaPS itself focuses on achieving constant-time lookup through the application of cascading cuckoo filters with zero-false positives.

Through mathematical modeling, simulation, and experimentation, the configuration parameters in cascading cuckoo filters and the Scrybe blockchain ledger were optimized to enhance performance metrics and scalability while ensuring security against the threat model.

1.3 Originality of Idea

Blockchain-based X.509 certificate management and revocation methodologies [10, 11, 12] have been proposed before. These approaches, however, provide limited scalability as they are based on Bitcoin that implements Proof-of-Work (PoW), a decentralization-focused consensus algorithm. SCRaaPS implements the Lightweight Mining (LWM) [13] algorithm, a secure, provenance-focused consensus algorithm introduced by Scrybe, which makes it highly scalable. Multiple prior research works [14, 15, 16, 17] have explored the use of cryptographic accumulators for distributing CRL membership info to achieve enhanced scalability and strengthened preservation of anonymity. However, these approaches focus solely on minimizing latency (metric for performance, as described above), thus failing to build an exhaustive collection of CRL data. Further, cryptographic accumulators incur huge computational and communication costs for updating revocation lists; therefore, they are more applicable in PKI deployments where the frequency of updates to revocation lists is relatively low. To the best of our knowledge, the use of recently proposed cuckoo filter [18], XOR filter [19], and binary fuse filter [20]. The design of cascading fuse filter to produce zero-false positives is also novel and unique.

1.4 Contributions

SCRaaPS addresses security threats in X.509 certificate revocation process and presents a reliable and trustworthy revocation alternative for use in large networks. The novel cascading cuckoo filter data structure designed and prototyped in this research enables quick, constant time lookup of arbitrary elements in large lists with zero false positives. The use of a cascading cuckoo filter broadens the support of PKI with best practices to include billions of mobile and IoT devices. The cascading cuckoo filter is further applicable to a wide variety of computational algorithms and security procedures that require looking up arbitrary elements from a large list of elements such as blocklists, allowlists, and graph search.

Key contributions of this research work include:

- Design, analysis, and simulation of Secure Certificate Revocation as a Peer Service

- Design, prototype, analysis, and simulation of cascading techniques applied to different probabilistic data structures

- Comprehensive security review and analysis of X.509 certificate revocation methodologies

Public Key Infrastructure (PKI) is widely used over the internet [21] by HTTPS connections and the Defense Information Systems Agency (DISA). PKI for Common Access Cards [22] is one of several large-scale PKI implementations [23]. The popularity of PKI keeps growing steadily [21] with the rise in demand for secure communication and the increasing number of Internet-connected devices. Revocation plays a critical role in PKI, and unresponsive OCSP responders can severely dampen the performance of HTTPS communications, as well as increase the risk of Man-in-the-Middle attacks. Hence, the efforts of this research work focused on enabling a scalable and reliable revocation system have far-reaching impacts.

Popular alternatives[1] to TLS include Kerberos [25] and Web of Trust [26]. Both facilitate the encryption of data over authenticated sessions. However, Kerberos is also a centralized protocol [25] designed for small-scale networks relying on a third-party server for mutual authentication. As a result, it is vulnerable to having a single point of failure. Web-of-Trust has a high bar of entry[2] and faces similar problems of revocation as TLS [27]. Due to these reasons, TLS is far more popular, and HTTPS is quickly becoming the de facto mode of secure communication over the internet. Hence, pursuing research that strengthens TLS provides the best opportunity cost among related encryption techniques.

Projects like Certificate Transparency (CT) [28] also pose opportunities to research and improve PKI. CT enhances CA accountability by requiring them to maintain immutable append-only logs of activity, and methodologies like Trillian propose implementing such audit logs using blockchain technology. These methodologies motivate the CAs to act more responsibly and maintain an adequate number of active OCSP responders, but they don't necessarily improve the

[1] More alternatives exist like Convergence and Curve CP [24] but to limit the scope, we only consider solutions that have been implemented and are in active use by large so far.

[2] Web of Trust operates in a decentralized peer-to-peer paradigm where users can digitally sign each other's Public Key after verifying identity through association. New users joining WoT must have their key signed by enough number of other users before they become trustworthy, this may be especially difficult for people living in remote areas where WoT is not already popular.

scalability of revocation, neither do they help establish a unified collection of CRL data, which is the key to strengthening revocation in PKI. By solving key issues, the results of this research are more valuable in comparison to that of CT.

Other approaches [29, 30] proposing to replace PKI altogether using the decentralized blockchain-based platform for certificate management would take a long time to materialize given that blockchain technology is still in nascent stages. Proper standards need to be established before it becomes capable of supporting such a large-scale solution. Thus, SCRaaPS research provides more immediate impacts and rewards for the efforts undertaken.

Demonstrating and quantifying the potential of a blockchain-based solution for X.509 certificate revocation is a valuable addition to this field's literature. The results of this research illustrate such a solution's viability, therefore providing concrete evidence for the utility of much-hyped, blockchain technology. Although cryptographic accumulators are extensively studied by researchers, especially in the field of Identity validation and PKI-based encryption [14, 15, 17]. Blockchain technology was recently proposed; thus, most of these studies are not implemented in conjunction with blockchain technology. Moreover, none of those research studies used a cascading fuse filter because it was recently proposed and did not exist for as long as most of the other cryptographic accumulators. Further, cryptographic accumulators that are based on public-key cryptography (such as RSA Accumulators) incur high computational and communication overhead for updating the witnesses. And the use of probabilistic data structures despite false positive potential has proved to be more promising for distributing revocation lists.

1.5 Dissertation Overview

The remainder of this dissertation is organized as follows: In Chapter 2, we explain and illustrate the relevant background information related to this research topic. We then list and summarize previous research work and publications by other authors in this area and describe how we improve upon these works. In Chapter 3, we describe in detail our approach to designing and implementing the blockchain-based secure certificate revocation system. We begin by listing design goals and system architecture, followed by a list of known threats and their mitigations.

Next, we define metrics to measure performance and compare our solution with similar systems. In Chapter 4, we outline our experimentation and present results. We begin with the performance and space-efficiency analysis of a cascading cuckoo filter, followed by a cascading XOR filter and fuse filter. The cost for delta updates is also described. In Chapter 5, results are discussed and the SCRaaPS system's performance and computational requirements are compared with other similar solutions. Lastly, in Chapter 6, we summarize this dissertation by reviewing contributions and explaining potential areas for future work.

CHAPTER 2

BACKGROUND AND LITERATURE REVIEW

Encryption of communication ensures the confidentiality of data. There are two well-known techniques for encryption: Symmetric and Asymmetric. Symmetric Encryption involves using the same key for both encryption and decryption, therefore the sender and receiver must share a common key prior to communication. The Asymmetric Encryption technique involves using a key pair with a public key for encryption and a private key for decryption. There is no need for sharing the key; however, the identity corresponding to a key must be validated before a sender begins encrypting data using the intended recipient's public key. This is important because public keys are 3,072-bit base64-encoded strings that represent large prime numbers. It is hard to memorize and associate different keys to real-world identities by the human brain, therefore, we rely on a trusted authority to validate and certify identities for the purpose of establishing trust among untrusted parties. The Zooko's Triangle [31] describes a trilemma between 3 important properties required for naming the participants in a network protocol, like that of Domain Naming Systems (DNS). It was conjectured that it is impossible for any naming system to achieve a single kind of naming scheme with more than 2 of the following properties:

1. **Human-Meaningful:** Which have a meaning in a real-world context and are easy to remember by users
2. **Secure:** Able to withstand attempts to impersonate by malicious entities
3. **Decentralized:** System responsible for resolving names to entities must not be controlled by a single authority

For instance, the Domain Name System Security Extension (DNSSec), which uses a chain-of-trust hierarchical system, provides a decentralized naming system with human-readable names for internet addresses, but the root authority is capable of circumventing security and impersonating identities. Therefore, DNSSec achieves 2 out of the 3 properties mentioned in Zooko's Triangle. Likewise, the X.509 protocol for associating public keys to DNS endpoints offers human-meaningful DNS records (like google.com, utc.edu). It is hierarchically decentralized[1] but is not secure if the root or trusted authority is compromised.

Digital Certificates issued by a trusted authority are a well-known form of identity validation. The X.509 protocol [1] is the standard for issuing digital certificates for Internet Domains by trusted third parties called Certificate Authorities (CA). The rest of this chapter is organized as follows: In Section 2.1, the concept of digital identity is described and the need for establishing trust is explained by showing how phishing attacks can impact businesses and individuals accessing the internet. In Section 2.2, the X.509 protocol, which is used for certifying public keys of website owners, is described in detail including the different types of revocation mechanisms that emerged since the standard was initially proposed. In Section 2.3, the recently proposed blockchain technology is introduced, followed by a list of the opportunities, tradeoffs, and challenges involved with utilizing blockchain technology. Next, related work to blockchain-based X.509 Certificate revocation mechanisms is listed and explained. In Section 2.5, approximate set-membership data structures that provide constant-time lookup are described including an analysis of probabilistic data structures and cryptographic accumulators. The chapter concludes with Section 2.6, where we describe the novelty in the concept of SCRaaPS, and its potential benefits and trade-offs in comparison with other similar state-of-the-art revocation mechanisms.

[1]The hierarchical model for X.509 certificates begins with the list of root authorities at the top that is supposed to be well-known and trustworthy. These authorities are typically manufacturers of devices and/or device operating systems like Apple, Cisco, Google, and Microsoft. This list may vary from device to device and is supposed to be stored locally on each device. These root certificate authorities can delegate the process of validation to intermediate certificate authorities like Digi Cert, Symantec, and Comodo. These intermediate certificate authorities are responsible for issuing certificates to web servers or services. Note, this hierarchical model consisting of root certificates, intermediate certificates, and leaf certificates is decentralized through delegation but is not peer-to-peer like the PGP encryption mechanism.

2.1 Digital Identity, Establishing Trust, and Phishing Attacks

The rise in electronic transactions with expanding e-commerce and e-banking services has increased the necessity to secure an individual's digital identity. More importantly, electronic payment systems becoming ubiquitous has increased the risk of financial loss through phishing attacks and fraudulent websites. All payment systems, in one way or the other, validate the digital identity of the owner prior to authorizing a transaction. Individuals can be authenticated by recording unique information about that individual during onboarding and then verifying if the user is able to reproduce this unique information during authentication. There are three broadly categorized techniques [32] to identify individuals for authentication:

1. **Something we know** (like the username and password combination that uniquely identifies a user or an account)

2. **Something we have** (like the keycard or access to a phone number that is unique and only the owner of the account is in possession)

3. **Something we are** (like the biometrics authentication using facial recognition or fingerprint recognition)

On the other hand, organizations and businesses that serve a group of individuals/ customers, are responsible for securing transaction endpoints and authentication systems. Individuals enter account credentials in transaction endpoints for authentication and authorization purposes. An attacker may launch phishing attacks by impersonating businesses/organizations over fraudulent websites or transaction endpoints to steal user credentials. Victims of successful phishing attacks lose access to their accounts and financial assets. Therefore, it is important to establish trust based on identity before proceeding with any kind of financial transaction. Users must be able to trust an organization's website over the internet and should be able to successfully distinguish between a phishing or fraudulent website and the actual website.

Therefore, trust flows both ways. Organizations must be able to identify a user and verify if an arbitrary unknown user is indeed the owner of an account or not. And users must be able to trust that the website they are visiting is indeed the intended organization's real website and not some

phishing attempt. Specifically, users must be able to trust that the website address is the correct location of an organization's website and that the public key of a website is indeed owned by the organization.

The Secure HTTP (HTTPS) protocol ensures confidential communication over the public internet by using public key encryption. Alternative secure communication protocols like the PGP do exist but are not as popular as HTTPS. The X.509 standard was prepared to establish trust based on identity in the HTTPS protocol.

2.2 X.509 Protocol

The X.509v3 standard [1] provides specifications to implement the Public Key Infrastructure (PKI) system in which each entity has a public key, a private key, and a certificate that is typically assigned by a Certificate Authority (CA). Each certificate holds a validity period along with other important related information, and the CA may revoke a certificate before it expires for reasons including change of name, change of association between subject and CA (e.g., an employee terminates employment with an organization), and compromise or suspected compromise of the corresponding private key [1].

The standard defines one method of certificate revocation in which the CA periodically issues a CRL [1]. The CRL is a data structure signed by the CA that is made freely available in a public repository and contains a time-stamped list of serial numbers of all revoked certificates [1]. The process of certificate validation involves checking for the certificate's serial number in the "suitably recent" CRL [1]. A major advantage of this method was that the means to issue CRLs is the same as that of issuing certificates—namely, via untrusted servers and untrusted communication [1].

If a browser does not already have a fresh copy of the CRL, then it has to fetch it during the initial connection, which can increase latency in connection establishment. The problem of scalability arises when the CRL suddenly becomes large; for instance, close to 50,000 certificates were revoked at once after the HeartBleed bug. The browsers had to download large CRLs, which slowed down most website connections [9]. Moreover, the periodic nature of updating the CRLs

opens a window of opportunity for attackers to work with revoked certificates until the next updated CRL becomes available, which can be an hour, a day, a week, or even a month in some cases. Additionally, there may be instances when a CRL server is unable to handle requests from clients in which case most browsers render it as a "soft fail" [33] and accept the certificate.

SSL is susceptible to Man-in-the-Middle attacks in which an attacker can insert a CA certificate into the client's root store so that the client may communicate with the attacker's website in a way that seems perfectly secure and legitimate. Furthermore, the "CRL Distribution point" field in the certificate is an optional field and is considered non-critical [1]. Additionally, an attacker using a revoked certificate, with no CRL distribution point mentioned, will most certainly have an advantage of not being discovered.

The Online Certificate Status Protocol (OCSP) was proposed to replace CRL revocation [34]. OCSP enables a client to verify the status of a certificate dynamically during the stage of connection establishment and specifies a client-server architecture in which a client can request the revocation status of a certificate. An OCSP responder will respond with a status of good, revoked, or unknown [34]. The response was intended to be both quick and lightweight and to solve the scalability issue with large CRLs that needed to be downloaded and parsed. The window of vulnerability vanishes since the request or response is dynamic and the client always receives a response from the most updated knowledge of an OCSP responder [34]. The protocol also mentions certain standards for the encryption algorithm to be used in the request, and the responder can reject any request using a weak encryption algorithm [34].

While OCSP solved many problems inherent in CRLs, implementation was not quite up to the mark. As a single point of failure, OCSP responders incurred too many requests and became the bottleneck. They were notorious for being slow to respond [35] and for not maintaining good up-time [35]. Another case of poor implementation arose when a client would attempt to log in on a captive portal while waiting for an OCSP response, which could be blocked by the captive portal [35] that would perhaps allow a response only after a user logged in. Web browsers like Chrome

and Firefox devised a workaround to enhance user experience by implementing OCSP with a "soft-fail" approach [33] (i.e., if a certificate is within its validity time and an OCSP responder takes too long to respond, then the certificate is accepted to be valid) [35].

Recognizing all these flaws, the IETF proposed a new technique, commonly referred to as OCSP-Stapling, in which the client can request the revocation status of a certificate as part of the TLS handshake [36]. A website that enables this feature must periodically request its Certificate Authority (CA) for updated revocation status [36] and must also send the most updated information as part of the TLS handshake response. This method saves the client the burden of having to make a third-party request for revocation status, therefore resolving the problem of slowing website communication while safeguarding the client from attacks like Denial of Service. This technique would have worked well if the protocol enforced the server to always send an OCSP response, which it did not do [3]. Even though a client sends an extension "*status_request*" as part of TLS handshake (which mentions that the server must include an OCSP response in the handshake), the server may choose not to [3] append an OCSP response and the client will have to accept the certificate as valid. Alternatively, the browser client may choose to request the OCSP responder by itself, but this option would not be without all the inherent vulnerabilities of OCSP. This gap was perhaps not filled because not all CAs were equipped at that time with OCSP response capability. There would be a lot of connection failures if the clients were enforced to fail from connection establishment to a CA that has not yet implemented OCSP response methodology. Later, a modification to the TLS standard was proposed in which a client is forced to fail from establishing a connection to a server that does not respond with an OCSP response [3]. By then, most CAs had evidently implemented the capability.

In a subsequent addition to the protocol, an extension "*status_request_v2*" was introduced to carry out OCSP status checking for all the intermediate certificates present in the certificate chain [37]. A client mentioning the "*status_request_v2*" extension must also mention a list of data structures containing the list of certificates for which revocation status is requested. The server may respond by "stapling" the list of revocation statuses for all the certificates. This development was substantial in improving certificate revocation since the revocation checking of intermediate

certificates was crucial, and client browsers had to rely on CRLs or other methods to achieve it. Client browsers could mention trusted OCSP responders [37] and would accept an OCSP response coming from one of those responders, but they would also accept an OCSP response from an authorized OCSP responder [37]. (An authorized responder is one that has been delegated by a CA that issued the certificate for the website and signed the delegation using the same private key [3] that the CA used to issue the certificate.)

Presently, browsers implement various revocation methodologies. Mozilla Firefox moved away from CRL revocation in 2010 [38] and enforced OCSP response with CRL as a fallback. In 2013, the company announced its own methodology called "OneCRL" [39] in which a centralized revocation list containing the revocation status of all the intermediary certificates was maintained and pushed to clients periodically. Later, it also enabled OCSP stapling and currently enforces OCSP must-staple methodology. In 2012, Google Chrome announced that it would stop conducting any standard form of revocation checking, like CRL or OCSP [40]. Instead, Google designed its own methodology called CRLSets. The company maintains a comprehensive internal list of crawled CRLs [41], which are mostly obtained from CAs. From this internal list, only those with no reason code or the specific reason codes (Unspecified, KeyCompromise, CACompromise, or AACompromise) [41] are published to clients. CRLs are published periodically every few hours. The implementation ensures that most or all of the intermediate certificates are part of the published CRL [41].

2.3 Blockchain Technology

Blockchain technology is a peer-to-peer decentralized computing platform as opposed to centralized server-client architecture. The participants of the blockchain network, also called peers, are all responsible for maintaining the entire blockchain history from the origin of the network until the latest block. This decentralized platform does not require a third party for establishing trust among untrusted parties. The consensus algorithm is the key mechanism that restricts participants from equivocating and ensures fairness in the system. In this section, we list the opportunities,

challenges, and tradeoffs related to blockchain technology. Later, we mention known applications of blockchain technology in the digital identity space.

2.3.1 Advantages and Limitations

Blockchain technology was introduced to the world by Satoshi Nakamoto [42] as a peer-to-peer electronic cash system. Although initially proposed for currency applications, the peer-to-peer decentralized and distributed data store can be applied to a wide range of use cases.
Following is a list of **advantages** of blockchain technology:

- Decentralized trust establishment

  The peer-to-peer decentralized architecture of computing avoids the need for a trusted party to facilitate trust among trusted parties. The underlying consensus algorithm ensures a consistent view of the data at all times and restricts participants from equivocating. Further, the account names are pseudonymous, but the tight correlation between addresses and identities and the attached digital signatures ensure the non-repudiation of data or transactions.

- Highly available data store

  The peer-to-peer distributed data store requires all participating nodes (peers) to store the entire blockchain history. Therefore, any one of the peers can respond to a request for data or computation. Therefore, in case of Denial-of-Service attacks, a subset of the nodes may be unreachable, but the rest of the nodes that are potentially spread globally can serve new responses.

- Immutable, append-only data

  The blockchain protocols use cryptographic techniques like Secure Hashing Algorithms (SHA) to ensure the **integrity** of data components. And once committed, this data cannot be rewritten or modified. Therefore, blockchain data stores are append-only with high integrity, therefore, ideal for auditing use cases.

- Built-in provenance

  Each block or data sub-component of the blockchain is attached with a timestamp, and the data store is append-only with new data being added in chronological order. Additionally, each transaction or data submission requires the digital signature of the sender, which proves authorization.

Although blockchain technology is full of potential, it is relatively new and constantly evolving. Following is a list of currently known limitations of blockchain technology:

- Scalability limitations

  The blockchain-based data store is limited by Buterin's scalability trilemma [43], which states that distributed computing platforms can only achieve 2 out of the 3 qualities: scalable, decentralized, and secure. The blockchain-based peer-to-peer distributed data stores are highly decentralized and secure but lack the desired scalability. For example, Bitcoin can currently process 7-8 transactions per second [44], and Ethereum can process 13-15 transactions per second [45]. Recently developed blockchain platforms, like Algorand [46] and Ethereum 2.0 [47] with Proof-of-Stake consensus algorithms, can potentially process 1000 transactions per second [48], but they are yet to be tested in production. Even otherwise, they are no match to global payment systems that can process 100,000 transactions per second on average.

- Public data store, not privacy-preserving

  The blockchain-based data stores are peer-to-peer distributed; therefore, all nodes store the entire blockchain data. Even though encryption can help safeguard data from unauthorized access, all encryption schemes are theoretically secure, and it is only a matter of time and computes resources to crack encryption schemes. Therefore, sensitive data must not be stored on public blockchain networks.

- Unlikely but potential attack probability

  All blockchain protocols are Byzantine Fault-tolerant, which means the network is consistent and functional only if the number of malicious participants is less than one-third of the total participants. Blockchain protocols running Proof-of-Work consensus algorithms are susceptible to well-known 51% attacks, whereas if an attacker or a group of attackers gain access to 51% of the total mining power, then they can potentially rewrite blockchain history or get away with equivocating or double-spending. Ethereum Classic, the previous hard fork of Ethereum, suffered the third instance of a 51% attack in a single month on August 29, 2020 [49].

2.3.2 Related Work

We mention certain related work in this section that involves blockchain-based public key certificate revocation mechanisms.

Namecoin is an alternate, decentralized Domain Naming System that utilizes Bitcoin as its underlying blockchain and shares the proof of work mining algorithm [12]. Namecoin domains end with `.bit`, which is not registered as a top-level domain and requires an additional client-side domain name server or a service for domain name resolution [12]. Presently, the developers are working on certain tweaks [50] to client operating systems and client software [51] that can help achieve the same. Clients can reserve a domain name by first using the Namecoin protocol followed by paying for it using Namecoin cryptocurrency [52]. This transaction allocates the domain name to the client, and the IP address assigned to this domain can be verified by the public [12]. Currently, the price of Namecoin is quite low. This has encouraged a lot of domain squatting [12], a situation in which users relentlessly buy domain names for already popular companies hoping that later they might reap a substantial profit in return for relinquishing that domain name later to a legitimate party. Presently, the `.bit` domain names cannot be resolved on most mobile operating systems because the mobile operating system APIs may not permit installation of an additional domain server required to resolve `.bit` domains.

Certcoin proposes a decentralized alternative public key infrastructure based on Bitcoin and Namecoin [11]. The model implements two pairs of keys [11]: the *online* key pair that is used for authentication and encryption; and the *offline* key pair for the purpose of revoking and updating the *online* key pair in case of a key compromise. The domain is registered on the Namecoin blockchain, and Certcoin credentials are used for securing the communication between a client and the server. The proposal explores various accumulators for speeding up the key lookup and verification process [11], including a Bloom Filter data structure. Certcoin is integrated with Namecoin and does not aim to address the latter's inherent limitations.

Revocable self-signed TLS certificates – We note the concept of implementing self-signed certificates in PKI using Bitcoin as the underlying technology [10]. The author suggests that the issuance of a self-signed certificate by a server can be recorded as a transaction of Bitcoin [10], and the public key can be shared with clients to check the balance of this Bitcoin public key. In case the server decides to revoke the certificate, it may choose to spend the amount on the address, and the clients would know that the amount has changed [10], therefore, signifying that the certificate was revoked. In this case, the round-trip time to check the balance of a Bitcoin key may be slower than the present OCSP response. With Bitcoin, mining the block itself may take considerable time. Further, adding to the Bitcoin blockchain requires a lot of time as one has to wait for at least six blocks to be added on top of the current block until which time a transaction may not be considered successful or valid. This can open a substantial window of vulnerability for an attacker to exploit. Also, the author has evidently not considered the scenario in which anyone else may deposit money to this public address, which makes it difficult for clients to validate the true state of a certificate.

2.4 Secure Provenance

Here we define secure provenance and indicate how its properties can address the revocation problem.

Data provenance is metadata that can be used to track changes in data [53] over time and ensure integrity. Secure provenance is achieved in a system where the integrity of provenance data can be maintained and ensured, and the metadata is always available for queries. The

metadata collected in secure provenance must always follow the chronological order in which events occurred and must be immutable [54]; that is, once logged, the information must remain read-only and should not be susceptible to falsification. The system enforces accountability and non-repudiation where an entity cannot claim that it was not responsible for a change that occurred and is logged in the system [55]. Further, in case an error occurs, changes can be traced back chronologically to identify when and what triggered the change responsible for the error [55].

Having secure provenance data for revoked certificates will help ensure the integrity and availability of revocation. There is no confidentiality goal for such data. Secure provenance data will incorporate the revocation status of all the intermediate certificates including when, by whom, and why a given certificate was revoked (some can be revoked implicitly by virtue of an antecedent's revocation). Additionally, chronologically ordered data can be implemented with data structures that provide insight into the hierarchy of the certificate chain, thereby offering convenient revocation of all the certificates that an intermediate certificate provider may have issued.

2.4.1 Overview of the Scrybe System

We explain how we verify that Scrybe supports non-repudiation and is also robust against distributed denial of service (DDoS) attacks; in particular, we explain how Lightweight Mining (LWM) algorithm, a unique feature of Scrybe, proves resilient to such attacks.

Overview

As illustrated in Figure 2.1, there are two main components of the Scrybe blockchain: blocks and transactions. A blockchain is simply a sequence of linked blocks where the current block contains the hash of the previous block.

Blocks

As previously mentioned, each block contains the hash of the previous block, which makes the blockchain *immutable*. Blocks are added to the blockchain by *miners*, entities responsible for maintaining the integrity of the blockchain. Scrybe only allows authorized entities to mine blocks

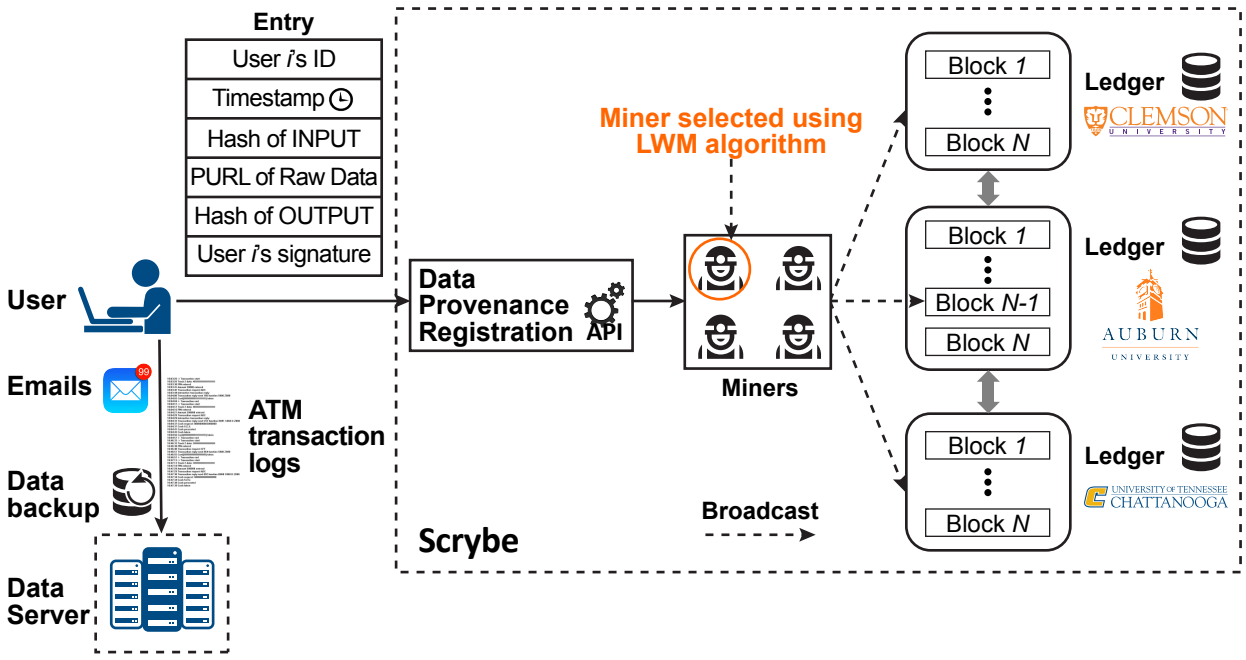

Figure 2.1 Scrybe's architecture showing (from left to right) transaction submission to the Scrybe consortium followed by miner selection and block commitment on the distributed, fully-replicated blockchain servers

through the secure LWM algorithm (comprising the Scrybe consortium). Miners are responsible for aggregating a list of transactions and calculating the Merkle root. The Merkle root allows other miners to quickly verify that every transaction is actually included in the block. When a miner is selected to add a block to the blockchain, the block is broadcast to all the other miners, and the data are verified (previous hash, Merkle root, and the miner's signature). At this stage, other miners will be able to detect if a transaction is omitted from the block, if an unauthorized miner broadcasts a block, and if the miner's signature is invalid.

Transactions

Transactions are the backbone of provenance. Conceptually, transaction input is categorized as input fields and output fields. A transaction in Scrybe takes input fields, output fields, and the submitter's details including the name, public key, and signature as input. The miner adds the timestamp as part of the transaction. Transactions can also be *genesis events*, which register the acquisition of new data. The persistent URLs (PURLs) that point to the data, along with the

SHA-3 hash of the data, ensure its validity. Note that all transactions have output fields. Genesis events **only** have an output with no input, but normal transactions have both.

By only storing the SHA-3 hash of the transaction instead of the original transaction, we can drastically reduce the size of the blockchain; consequently, there will consequently be no penalty for an extensive number of inputs and outputs in any given transaction. The original transaction will be stored on a *transaction server*, which will be locally maintained along with the *data server* and the *metadata server*.

Note that for the SCRaaPS application, we have changed the transaction format and scheme compared to the nominal Scrybe format. We store certificate information directly on the blockchain, which we choose to do because there is no need for a separate data or metadata server in this use case referred to by PURLs. Our Entries contain sufficient information to describe certificate revocation without reference either to an external data or metadata server. This simplification is essential because SCRaaPS is working to eliminate DoS/DDoS against revocation information. Furthermore, we store cuckoo filter coefficients periodically on the blockchain (handled by a single trusted agent in the first implementation and to be handled by a distributed app on the blockchain itself in the future).

2.4.2 Lightweight Mining

Scrybe introduces a novel way to mine new blocks in the blockchain, which is not a difficult proof-of-work required in cryptocurrency applications. The lightweight mining algorithm (LWM) introduced in Scrybe is presented in the following frame.

Lightweight Mining Algorithm (LWM)

Input: The number of miners N.

Algorithm: For each miner $m_i,\ 0 \leq i < N$,

- *Step 1*: m_i generates a random number r_i;
- *Step 2*: m_i broadcasts the SHA-3 hash of r_i, denoted by $H(r_i)$;

- *Step 3*: Once m_i has collected all N hashes $\{H(r_0), H(r_1), \cdots, H(r_{N-1})\}$, m_i broadcasts r_i.
- *Step 4*: Once m_i has collected all N random numbers $\{r_0, r_1, \cdots, r_{N-1}\}$, m_i calculates $l = \sum_j r_j \bmod N$.
- *Step 5*: m_l is the selected miner to create the next block from the collected transactions. (Without loss of generality, we map $m_i = i, 0 \leq i < N$ as a simple rank ordering for the registered miners.)

The Genesis block contains the information related to initial miners, and the number of miners is fixed before the beginning of every round of miner selection. Each round has a fixed timeout for synchronization. If a miner fails to send the hash within this timeout, then that miner's hash and the random number are considered to be $NULL$ for that particular round. Optimal timeout depends on the number of active participants and the desired number of transactions per round.

Considering that the network is a permissioned network, new miners wishing to join are not accepted until after the end of the current round. The purpose of LWM is to provide randomization in miner selection. In a Denial of Service (DoS) attack against Scrybe, we assume a malicious miner targets a particular user by excluding the victim's transactions from the block he or she creates. The randomization offered by LWM, coupled with the fact that each miner maintains a local pool of transactions, guarantees the victim's transactions will always be integrated sooner or later, as long as there is at least one honest miner.

The core idea of LWM is "sharing-hash-first." If every miner sends out the random number without sharing the hashes first, a miner can hold his or her own number until he or she has received everyone else's random number. This caveat allows a malicious miner to manipulate miner selection by choosing a number that produces a m_l in favor of a particular miner or deliberately excludes a particular miner. Scrybe takes a naive approach to random number generation in a peer-to-peer service. Each node generates a random number independently before sharing the hash

of that random number. This method has been proven to be robust [56]; as long as one node is generating a random number, the $\sum_j r_j \bmod N$ remains random.

"Sharing-hash-first" ensures that every miner has to share his or her own number (in the form of the hash) with others before they see others' choices. Since hash values are considered impossible to invert in practice, a miner cannot change the random number after the fact. Further, the hash is signed with the sender's digital signature, which disallows a miner from equivocating. Each miner may broadcast any number they wish, and it is in the interest of each sender to broadcast a random number to avoid predictability and a pattern that can be exploited to reduce the chances of the sender being selected. Thus, LWM can tolerate up to $N - 1$ malicious miners who collude. As long as there is one miner generating a random number, the modulo operation is randomized.

The LWM consensus ensures a one-CPU-one-vote majority, same as PoW [42], but there is no need for the concept of the longest chain. In the case of a miner trying to broadcast a block containing a previous hash entry that is not the same as that of the latest block that block will not be added to the blockchain. Furthermore, there is no chance for a fork in the blockchain for Scrybe since only one miner is chosen in each round to proposing the next block.

Servers – Locally maintained transaction servers will hold the transactions comprising the ledger. An additional metadata server can be maintained along with the transaction server wherever it makes sense. The integrity of the database can be verified by generating transaction lists for each block and ensuring that these transactions and corresponding hashes accurately display the state of the database. If there is any discrepancy, the database server is deemed disreputable. The integrity of the data and metadata can be verified by comparing the SHA-3 hash of the data to the SHA-3 hash stored in the transaction. If these hashes differ, the relevant server is considered disreputable. The method for storing data on these servers is configurable and left to the end-user's discretion.

The Certificate Authorities can maintain an optimal minimum number of transaction servers, processing OCSP-staple requests from the web servers that have certificates assigned by that CA. A quorum of all Certificate Authorities can be established with a common blockchain containing the list of all revoked certificates.

2.5 Set Membership Data Structures

In this section, we introduce set membership data structures and describe a well-known classification of such data structures. We then provide a comparison of the two types of set membership data structures with regard to insert, lookup, and update operations to a set of elements.

A set is formally defined as an unordered and well-defined collection of distinct objects [57, 58]. Each object in a set is called an element. The existence of an element in a set, also known as set membership, is depicted using the symbol $\in$ and read "is an element of." If the element is not in the given set, the $\notin$ symbol ("not an element of") is used. In computer science and engineering, set membership tests are used in many applications including but not limited to the database, authentication, and validation systems. As noted above, any computational problem where the answer is yes or no can be formulated as a set membership problem.

Testing set membership can be performed by running a search on the set, but this method can be a resource-intensive task as the set size increases. To address these limitations, researchers proposed cryptographic accumulators [18, 59, 60, 61]. The fundamental idea behind the accumulator is being able to accumulate values of a set A into a small value z in such a way that it is possible to prove only the elements of set A have been accumulated [62].

2.5.1 Classification

We describe Set Membership data structures, which are implemented using cryptographic primitives. They can be categorized as symmetric and asymmetric accumulators.

Symmetric Accumulators [63] are designed using symmetric cryptographic primitives and can verify the membership of elements without the need of a corresponding witness. The Bloom filter [64]—a type of array data structure—is a symmetric cryptographic accumulator that uses k-number of hash functions that set a unique combination of indices in the array based on the input element. It provides a limited representation of set membership with a false positive rate that grows as the number of elements in the list approaches the max capacity of the list [64]. Equation 2.1

provides the estimate of the false positive rate of a simple Bloom filter construction:

$$FPR = (1 - [1 - \frac{1}{m}]^{kn})^k \approx (1 - e^{\frac{-kn}{m}})^k \tag{2.1}$$

with m being the size of the array, k being the number of hash functions, and n being the number of accumulated elements. Variations of the Bloom filter [18] have sought to minimize this false positive rate but are unable to eliminate it entirely. Because Bloom filters are static accumulators, they cannot accommodate growing list sizes, so they must be regenerated after reaching full capacity during the transaction discovery process[2].

Table 2.1

List of well-known probabilistic data structures

Name	**Year**	**bits/ item**
Bloom [64]	1970	$1.44 \log_2(\frac{1}{\epsilon})$
Cuckoo [18]	2014	$(\log_2(\frac{1}{\epsilon})$+3)/0.955
Cuckoo Semisort [18]	2014	$(\log_2(\frac{1}{\epsilon})$+2)/0.955
XOR [19]	2020	$1.23 \log_2(\frac{1}{\epsilon})$
XOR+ [19]	2020	$1.0824 \log_2(\frac{1}{\epsilon}) + 0.5125$
Fuse [20]	2022	$1.0725 \log_2(\frac{1}{\epsilon})$

The recently proposed cuckoo filter is a dynamic data structure that functions similarly to the simple Bloom filter but with additional capabilities such as the ability to delete elements. A cuckoo filter implements a cuckoo hash table [18] to save fingerprint representations of the elements in an accumulated set. A cuckoo hash table is an array of buckets in which each stored element is associated with two indices in the array. Two associated indices allow for the

[2]As an example, a Bloom filter is employed in Bitcoin [65] to help Simplified Payment Verification (SPV) clients communicate with full nodes. A full node in the Bitcoin network follows the safest security model by downloading and maintaining a copy of the entire blockchain from the Genesis block until the most recent block.

dynamic rearrangement of the elements stored in the cuckoo hash table, providing optimized space efficiency and low false-positive rates. For a given number of elements, a cuckoo filter outperforms a space-optimized Bloom filter in terms of false-positive rate and space overhead [18].

Asymmetric Accumulators [63] require witness creations and updates for dynamic verification of set membership [59]. They are built on asymmetric cryptographic primitives [59] and require the underlying hash algorithm to exhibit the quasi-commutative property [66]. One example would be the RSA accumulator introduced by Beneloh and de Mare [66]. The RSA accumulator uses RSA modular exponentiation to achieve the quasi-commutative property. A simple RSA accumulator construction consists of the following expression for addition: $acc_n = acc_{n-1}^{x} \mod N$ where acc_n is the new accumulator value after addition, acc_{n-1} is the old accumulator value before addition, x is the element being added, and $N = pq$ where p and q are considered to be strong prime numbers whose Sophie Germain prime numbers $p' = \frac{p-1}{2}$ and $q' = \frac{q-1}{2}$ are the accumulator trapdoor. One drawback of an RSA accumulator is that it is collision-free only when accumulating prime numbers. A prime representative generator is required to accumulate composite numbers without collision[3]. In Tremel's implementation of an RSA [60] a random oracle prime representative generator provided by Barić and Pfitzmann [68] was used. Asymmetric accumulators can be further classified based on operations supported and the type of membership proofs provided. This classification will be further explained and analyzed in the following sections.

A Merkle hash tree can also be implemented as an asymmetric accumulator to prove the set membership of elements [59]. It is classified as an asymmetric accumulator because a member of its set requires a witness to prove membership (or non-membership). But, it does not use asymmetric cryptographic primitives nor does it need the underlying hash function to exhibit quasi-commutative property. A Merkle Tree takes the form of a binary tree. Each leaf node in the bottom layer of the tree is a hashed representation of a member in a set. All other nodes represent the hashed value of their two child nodes. The root node of the Merkle Tree is called the Merkle

[3]The bilinear-map accumulator introduced by Nguyen [67] uses bilinear pairings to achieve the quasi-commutative property. It can accumulate composite numbers without the need to generate prime representatives. We should note that the majority of the asymmetric accumulator constructions provided by the literature have been based on RSA and Merkle Trees.

root and its value is the pairwise accumulated hash of all of the non-root nodes in the tree. The Merkle root value must be recalculated when there is an addition or deletion of a member in the set. Checking for set membership can be done with a portion of the Merkle Tree [69], making it unnecessary to download the full data structure.

2.5.2 Comparison

In this section, we compare and contrast the two types of set-membership data structures.

Symmetric accumulators are bit-arrays representing a set of elements. Asymmetric accumulators are cryptographic hashes that also represent a set of elements, and each element has a corresponding witness to prove the membership of that element. In Table 2.2, we provide a comparison between the cuckoo Filter [18] (symmetric) accumulator and the RSA accumulator [70] (asymmetric). We list the computational complexities of each type by picking representative candidates of each type: a cuckoo filter for the symmetric accumulator and a One-Way RSA Accumulator for the asymmetric accumulator. There are other variants of each type that may provide additional benefits but they do not significantly improve operations. Those variants are application-specific, therefore, out of scope for this research.

The recently proposed probabilistic data structure cuckoo filter [18] is a symmetric accumulator. The insert and update operations are $O(1)$ and if there are N, new elements, they are each $O(N)$. However, the cuckoo filter yields a false positive rate. For this reason, they can be considered as an approximate set-membership data structure, rather than deterministic. On the other hand, the RSA Accumulator is an asymmetric accumulator that is deterministic in nature with no potential for false positives. However, each element needs a corresponding witness to prove membership, and this witness must be updated each time there is an update (insert a new element or delete a member) to the accumulator. Therefore, the update operation is highly expensive, with $O(n)$, $n \leq N$, where N elements are added or deleted from the set. This high cost of update makes it a challenging choice to implement for X.509 certificate revocation use case where thousands of valid HTTPS web certificates exist and hundreds are revoked each week.

Table 2.2

Comparison between cuckoo filter (symmetric) and RSA accumulator (asymmetric) set-membership data structures

	Symmetric Accumulators	**Asymmetric Accumulators**
Initial insertion complexity (n elements)	O(n)	O(n)
Update computation complexity (Insert or delete N elements)	O(N)	O($n \pm N$)
Lookup complexity	O(1)	O(1)
False Positive Rate	$\leq 3\%$	None

2.6 Summary

The secure certificate revocation system proposed in this research work improves the revocation schemes mentioned by the X.509 standard. It combines blockchain-based data stores for highly available data and approximate set membership data structures for quick, constant time lookup of revocation status. In this chapter, we pointed out the need for identity validation infrastructure for communicating over the internet. We then described the concept of digital identity and explained the significance of establishing identity trust over the internet. Later, we summarized the X.509 standard for establishing trust and the evolution of associated revocation mechanisms to overcome known limitations. We also introduced blockchain technology and described its advantages and limitations. We then analyzed well-known approximate set membership data structures for the purpose of implementing them in this research work.

CHAPTER 3

APPROACH

SCRaaPS is designed as an alternate service for verifying the revocation status of X.509 certificates that are issued to internet domain addresses. In this chapter, we describe our approach to implementing SCRaaPS. We begin with a review of the research questions that we plan to address in this dissertation. We then list the design goals and system requirements to address the research questions. Next, we describe the architecture and the various components of the system. We follow an evolving threat model that feeds back into design and requirements. In Section 3.4, we list various threat surfaces and potential tradeoffs. We also list mitigations to these threats that will be integrated into design and implementation.

3.1 Research questions

The latest specifications for revocation mechanisms in X.509 Standard, namely Must-Staple and Multi-Staple, close the gap in fixing certificate status availability. Browser clients can implement (with minimal impact) hard-fail strategies in case of missing revocation status during TCP handshake. However, the freshness and reliability of revocation status need further enhancement, especially given the imminent explosion [71] in the number of Internet-connected devices. Further, the revocation methodology must meet the higher rates of certificate issuance and revocation that accompany the explosion in the number of Internet-connected devices.

We identified the following 3 limitations to the X.509 revocation standard (in its most up-to-date form):

1. **Susceptibility to DoS:** The OCSP responders are susceptible to Denial of Service Attacks and enforcing hard-fail will cause outages to large portions of the internet in case OSCP

responders are not available for any reason for extended periods of time. CAs are required to maintain OCSP responders to serve certificate revocation status requests from web servers. In the case of CAs serving a large number of web servers, the amount of network traffic may be overwhelming, especially in times of occasional spikes in network traffic. Further, these OCSP responders must be distributed across fault domains to avoid impact on revocation statuses during DoS attacks.

2. **Fragmented CRL data:** The CRL data is distributed by CAs through their own distribution points. This increases the latency of revocation checking of multiple intermediate certificates in a certificate chain. Hence, it is challenging to develop any client-side implementation for spontaneous revocation checking of these intermediate certificates. This is especially valuable in the interim while OCSP Multi-Staple goes mainstream.

3. **Limited Scalability:** The OCSP responders of CAs with large clientele are required to handle huge loads of OCSP requests from the large number of users visiting client websites. They are notoriously known for being slow, therefore, they form the bottleneck in revocation. The OCSP stapling mechanism will mitigate this issue to some extent by offloading the burden of proof onto servers. However, until OCSP Stapling is widely adopted, the fallback mechanism to request an OCSP response by the client is too costly and is currently skipped by major browser clients like Firefox and Chrome [33].

We design and build SCRaaPS to mainly address these limitations and to build a more robust and reliable revocation service. In doing so, we address the following research questions:

RQ 1 Can decentralized blockchain technology improve the robustness and effectiveness of current revocation mechanisms?

The decentralized platform of blockchain technology provides a secure mechanism for establishing trust among untrusted parties. It provides a highly available data store with immutable append-only storage and cryptographically linked logs for ensuring non-repudiation. The features of blockchain technology are highly suitable to address the limitations of current revocation mechanisms, and so blockchain technology can be leveraged

to secure OCSP responders and CRL data against DDoS attacks. But, the technology is relatively new and faces challenges with scalability itself. While custom designing the blockchain and tuning its parameters can help achieve the desired results, it is important to understand the trade-offs and broader implications for doing so.

RQ 2 Can a set-membership data structure enhance the lookup performance and availability of certificate revocation in spite of potential false positives?

The set-membership data structures provide constant-time lookup for quickly determining if a given element is part of a set or not. Such data structures are potentially beneficial in quickly determining if the SHA256-bit fingerprint of a given certificate is present in the CRL set. Further, they provide a constant-sized representation of a large set that is space efficient to store and distribute. A global list combining all CRLs from all CAs can be compiled and represented by a set-membership data structure, which can be distributed to the clients and used for offline revocation status checking whenever necessary.

The probabilistic data structures, like the Bloom and cuckoo filters, are compressed representations of the set in the form of an array of bits and buckets of fingerprints, respectively. Probabilistic data structures have no false negatives, but there is a probability for false positives, which will potentially impede communication if implemented for determining revocation status against CRLs. Cryptographic accumulators are the alternate type of set-membership data structures built on public-key cryptographic primitives. They have neither false positives nor false negatives, however, they are communication intensive and require active participation to keep membership proofs up to date.

3.2 Design and Architecture

The Secure Certificate Revocation as a Peer Service (SCRaaPS) is implemented as an augmented version of the Scrybe blockchain protocol, utilizing the most recent version of Scrybe's consensus algorithm to maintain data consistency across participating nodes. In this section, we list and describe the design goals driving our implementation. Next, we present the overall architecture

of the application and explain different components, including how they are supposed to interact with each other. We then mention the rationale and our thought process in building our design.

3.2.1 Design Goals

SCRaaPS was conceptualized as an alternate revocation mechanism that is both robust and scalable. We identify the following design goals as guidelines driving our implementation towards successfully achieving our vision.

- Exhaustive repository of CRL data

  The CRL data submitted to and recorded on the SCRaaPS datastore should provide a holistic representation of all X.509 certificates that were revoked by all intermediate CAs. It is important to note here that a complete list of all intermediate CAs might be impossible to curate and even more difficult to maintain by any single party. However, this can be achieved through open participation by allowing any Intermediate CA with a valid X.509 certificate to submit CRLs and delta CRLs to SCRaaPS.

- Decentralized peer-to-peer solution without the need for trusting an authority

  The unification of CRL data and revocation status checking must be transparent and open for public participation. A single group or entity is responsible for compiling the revocation info and serving info for revocation status checking creates a point of centralization and presents single-point-of-failures. Hence, SCRaaPS must support public participation and provide full transparency.

- Quick and constant time lookup with a minimized false positive rate

  Users, through browser clients or application clients, should be able to quickly verify the revocation status of all certificates present in the website's certificate chain. This verification process should take deterministically quantifiable time that is either constant or logarithmic in the order of the SCRaaPS CRL size. Compressing large CRLs into Set-Membership data structures can help achieve constant time verification, but they may introduce new trade-offs like potential false-positive rates or additional communication overhead. Nevertheless, such

trade-offs must be avoided or eliminated to provide a fail-closed solution to revocation status checking.

- Low latency and high availability

  The size of CRLs can grow exceptionally large, especially under special circumstances like the aftermath of the Heartbleed bug discovery. SCRaaPS aims to minimize the latency introduced due to large CRLs and keep the query speed constant, even in the case of disastrous occurrences. Further, SCRaaPS reduces the impact of other influencing factors like the height of a certificate chain on the query speed. The blockchain-based data store is resistant to DoS attacks and DDoS attacks through uniform replication of data across all actively-participating nodes. However, in the case of a successful DoS attack, the most recent set-membership data structure is still available for use, and requests for updates can be made to other reachable SCRaaPS nodes.

3.2.2 System Architecture

In the blockchain paradigm of Scrybe, a miner server or a participating node is responsible for validating transactions and generating new blocks to be committed to the blockchain. The servers communicate with each other in a peer-to-peer decentralized network and participate in the Lightweight Mining (LWM) consensus algorithm to reach an agreement about the global state of the blockchain. The SCRaaPS solution is designed to be an augmented version of the Scrybe blockchain protocol where each server is additionally responsible for evaluating a probabilistic data structure to determine the set membership of a certificate or a certificate chain. While this may be the main distinction, there are additional application-specific changes implemented for optimized performance.

System components in SCRaaPS are as follows:

Transaction

The transaction is the atomic component in a blockchain ledger. A group of transactions is used to build a block in the SCRaaPS ledger. As shown in Figure 3.1, transactions in the SCRaaPS

Transaction Id	Cert Issued or Revoked	Certificate SHA256 Fingerprint	Certificate chain	CA Signature	CA public key

Figure 3.1 Structure of transaction in the SCRaaPS ledger

blockchain network consist of the X.509 certificate fingerprints (or the list of fingerprints) along with a Boolean value indicating whether each certificate (or list of certificates) has/have been newly issued or revoked. Additionally, transactions must consist of the CA's certificate chain, the CA's signature, and the CA's public key for the purpose of validating the transaction. The certificate chain is expected to be validated by each SCRaaPS node participant (during Block proposal and Block validation).

Blocks

Blocks collect a group of transactions from the mempool, validate them, prepare metadata, and are proposed to the network for validation. Note, only the node participant selected during the LWM consensus round to propose the block is required to process the transactions and prepare the block. Other participants simply validate it before committing to their copy of the ledger.

Algorithm 1 describes the block-processing operation. It begins with loading transactions from the transaction memory pool. All broadcasted transactions are initially stored in the transaction pool (in computer memory) and are accessed during Block proposal and validation. Subsequently, each transaction is validated by first validating the certificate chain and the public key of the CA. In this design, we assume that the set of root certificates in all the participating nodes is distinct. A list of root certificates can also be included in the genesis block, or the participants can simply agree on a source of root certificates to avoid forking the ledger.

Once the block limit is reached, the set of validated transactions is ready to be committed to the block. Next, the node participant generates the cascading filter free of false positives for quick set membership verification. This is achieved by parsing through the previous blocks and preparing the list of all fingerprints and categorizing them into two sets: revoked and non-revoked. This process can be sped up by caching these sets of fingerprints or by managing them in memory.

Algorithm 1: Block processing in SCRaaPS

```
for i ←0 to i≤blockLimit by i++ do
    txn ← mempool.pop_front()
    foreach cert: txn.certificate_chain do
        if valid(cert)!=true OR prevBlock.CF.contains(cert.SHA256) then
            break ;        /* Discard the transaction if validation
             fails */
        end
    end
    assert(hash(txn)==txn.hash);
    ValidateSignature(txn.ca_signature, txn.public_key, txn.hash);
    txns ← txns.push(txn)
end
CF←BuildCascades();
if blocknum%refreshperiod==0 then
    blockheader←CF;
else
    GenerateDifferentialUpdate(prevBlock.CF, CF)
end
blockhash←GenerateBlockHash(transactions)
GenerateBlockHeader(prevblock.blockhash, timestamp, blocksig, blockhash)
```

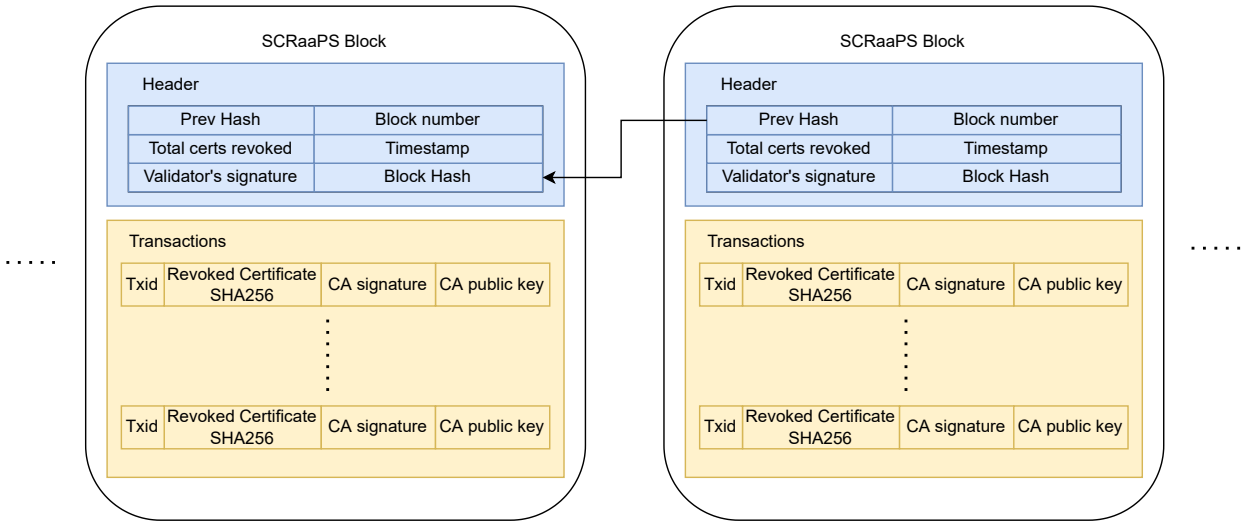

Figure 3.2 Block structure in SCRaaPS distributed ledger

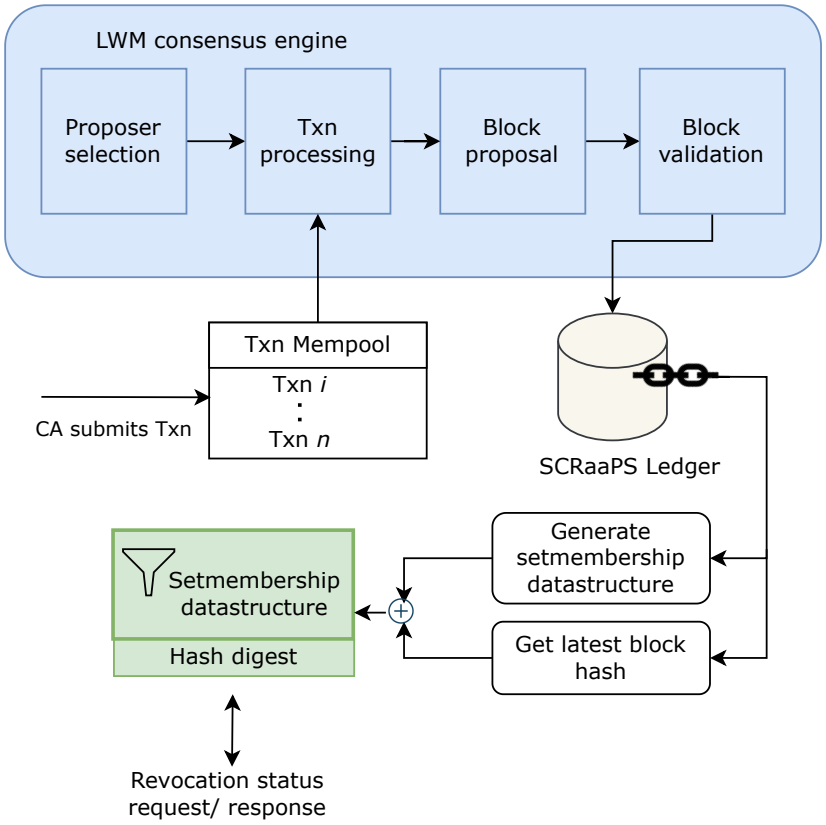

Figure 3.3 SCRaaPS server-side operations in a node participating in consensus

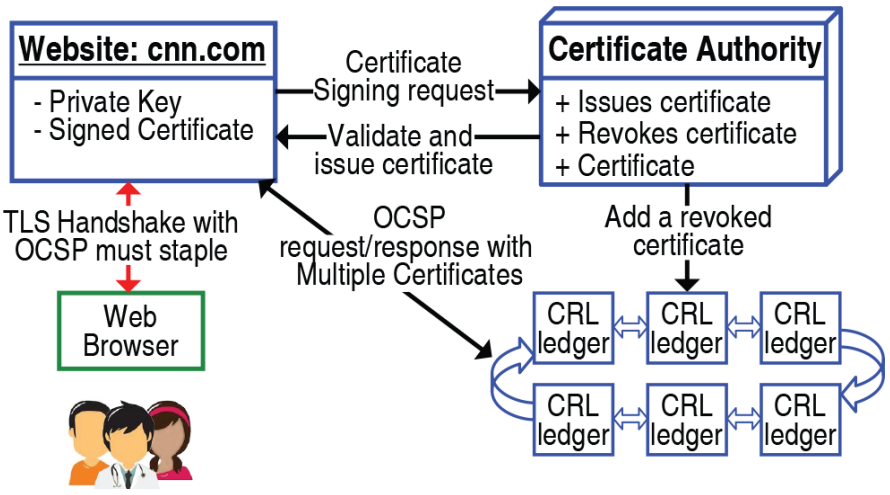

Figure 3.4 Revocation using the Scrybe distributed CRL ledger

Storing these sets of fingerprints separately on disk can help avoid blockchain traversal per query. The cascading filter is not committed in its full size in every block; rather, various updates are added to the block and the filter is committed in its full size periodically.

3.3 Cascading Algorithm on Probabilistic Data Structures

In this section, we present the design and methodology of a cascading cuckoo filter, which is a key contribution of this dissertation.

The *insert* and *lookup* operations provided by the standard Bloom filter [64] and cuckoo filter [18] are indistinguishable by design. Therefore, we observed that the filter cascading algorithm provided by [72, 73] can be applied to a cuckoo filter without making any changes. (This filter cascading algorithm will be optimized in future work by leveraging the delete operation, which is supported by a cuckoo filter.) For the sake of completeness, we briefly summarize the filter cascading algorithm here.

3.3.1 Precondition for Successful Elimination of False Positives

As previously mentioned in this document, it is possible to eliminate false positives in ASMDSs only if the set of potential false positives is known or can be identified. For practical

implementations, the false positives need to be identified in no more than polynomial time. This precondition also implies that the set of candidate elements for lookup (S) must be finite.

3.3.2 Cascading Algorithm

The core objective of the filter cascading algorithm is incrementally to represent smaller sets of false positives in subsequent layers or "*cascades*" primary ASMDSs. This process is expected to deterministically terminate with the final cascade of ASMDSs producing no false positives [8]. As shown in Fig. 3.5, given a set R, where $R \subset U$ and $R \cup S = U$, insertion into the cascading cuckoo filter will begin with primary cuckoo filter (CF_1) that represents the set R. The CF_1 would naturally produce false positives (Set FP_1) that belong to Set S. If the precondition is satisfied, it is possible to identify all the elements in Set FP_1 by conducting a lookup operation of all elements in Set S against the primary cuckoo filter CF_1. All elements in Set FP_1 can then be inserted into a separate cuckoo filter (CF_2). Note, the CF_2 is the first level cascade of the primary cuckoo filter and it represents the (false positive) elements in S, therefore, reversing the meaning of a successful lookup in CF_2. This is true for all cuckoo filter cascades at even-number layers.

The cascading process in the insertion operation is expected to continue similarly for $h + 1$ rounds until no more false positives can be identified. Cuckoo filters produce a false positive rate p (less than 3% under optimized configuration); hence, the size of set FP_1 is $p \times |U|$. The size of sets represented by subsequent cuckoo filter cascades reduces by p times. The cuckoo filter configuration parameters can be further optimized according to the size of U to reduce false-positive rates and to improve spatial efficiency [18]. Similarly, the number of filters required and the overall spatial efficiency of cascading cuckoo filter can be optimized by adjusting configuration parameters. The methodology for selection is described in [18].

3.3.3 Set-Membership Testing in a Cascading Cuckoo Filter

The primary cuckoo filter and its subsequent cascades represent a set of elements either belonging to R or S. An arbitrary input element e, where $e \in U$, can be used as input to the lookup operation to the primary cuckoo filter and its cascades. Note, it is important that the input element

Algorithm 2: Build cascades

```
Data: setR; setS;
cascades ← φ ;                                  /* Empty vector of ASMDS */
baselevel_asmds ← φ
foreach f: setR do
    baselevel_asmds.insert(f)
end
cascades.push_back(baselevel_asmds)
falsepositives ← φ
while true do
    foreach f: setS do
        if cascades.back().contains(f)==true then
            falsepositives.push_back(f);
        end
    end
    if falsepositives.size()==0 then
        break ;                  /* Exit if no more false positives */
    else
        temp_asmds ← 0;
        foreach f: falsepositives do
            temp_asmds.insert(f)
        end
        cascades.push_back(temp_asmds)
        setS ← setR
        setR ← falsepositives
    end
end
return cascades
```

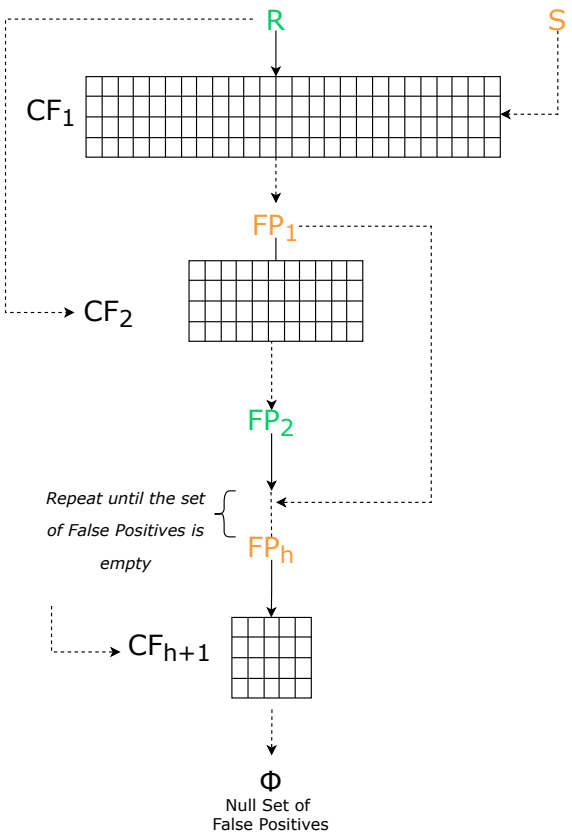

Figure 3.5 Inserting elements of a set R, where $R \subset S$, into cascading cuckoo filter. Cuckoo filter at level h is represented as CF_h, and the resulting set of false positives at level h is represented as FP_h. The dashed arrows represent "lookup" operations, and the solid arrows represent "insert" operations

e belong to Set U for successful membership testing. The ultimate output of true or false can be produced even if $e \notin U$, but this output may not be meaningful.

Algorithm 3: Lookup fingerprints in cascades

```
Data: cascades; fingerprint
i ← 0
not_found ← false
while i<cascades.size() do
    if cascades[i].contains(fingerprint)!=true then
        not_found = true; break;
    end
end
if not_found==true then
    if (i+1)%2==0 then
        return true
    else
        return false
    end
else
    if cascades.size()%2==0 then
        return true
    else
        return false
    end
end
```

Like Bloom filters, cuckoo filters do not produce false negatives. Therefore, observing a negative output to the lookup operation on any of the cuckoo filters terminates set-membership testing. Assuming no negative output occurs for any of the cuckoo filters, then the number of layers decides the result of the set-membership test. As shown in Fig. 3.6 and Algorithm 3, to test set membership of an arbitrary input element e, it is given as input to "*lookup*" operation in cuckoo filters at each level, beginning with the primary one. Unlike typical ASMDSs with a binary output of yes or no, the cascading cuckoo filter produces a multivariate output where the number of layers at which testing terminated is a deciding factor. The set-membership test output is summarized in Table 3.1 and is considered positive ($e \in R$) if

- the testing terminated with a negative output of the lookup operation at layer i and i is even; or,

- the testing terminated with a positive output on the final layer cuckoo filter (in other words, no cuckoo filter produced a negative for the input element) and the total number of layers is odd.

Otherwise, the test was unsuccessful.

Table 3.1

Summary of the multivariate output of set-membership testing in cascading cuckoo filter, showing if $e \in R$ or if $e \in S$

layer of termination	even	odd
negative output at layer i	R	S
positive output at final layer l	S	R

3.3.4 Space and Time Complexity

The cuckoo filter, like all ASMDSs, is of constant size regardless of the size of the set it represents. However, cascading cuckoo filters are of varying size and may have up to $h + 1$ filters, where $h > 1$. The amount of space occupied by a cascading cuckoo filter is proportional to h. Optimizing the configuration parameters of a cuckoo filter can help reduce false-positive rates of the filter at each layer individually. It may also collectively reduce the number of layers in, and total space occupied by, cascading cuckoo filters.

Computations involved with the insert operation in a cascading cuckoo filter can be considered as a collection of insertion operations and search operations at each layer. The largest computation occurs in preparing the first cascade layer, where all the elements in Set S must be looked up in the primary cuckoo filter (CF_1) to determine the set of false positives (FP_1). This operation alone requires computation in the order of $O(|U|)$. Computations required for subsequent layers reduce incrementally by a magnitude determined by the false-positive rate at each layer. The overall computational time complexity of preparing cascading cuckoo filters is proportional to the size of superset U and is of the order $O(|U|)$. Note that the cascading cuckoo

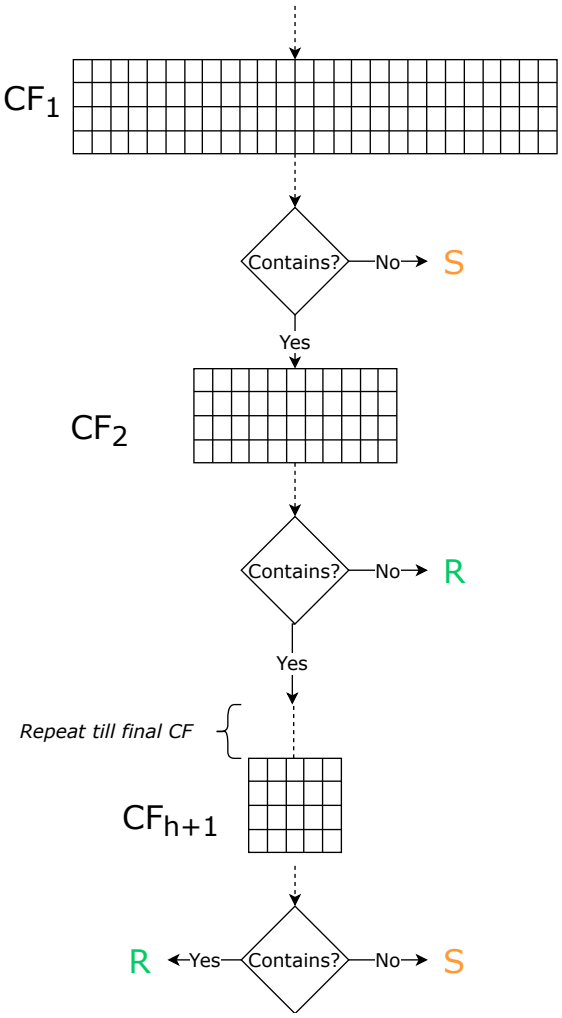

Figure 3.6 Testing set membership using cascading cuckoo filters. The input element e is looked up on the cuckoo filter at each layer until either a negative output is encountered or until the final layer is reached

filter ultimately represents the set R, which is a small subset of U, and can provide set membership definitively only for elements in R. It is important to further note that the computational time-complexity of insertion in a cuckoo filter is in fact proportional to the maximum number of *kicks* allowed during insertion and only its amortized time is of $O(1)$ [18]. The maximum number of kicks, regardless of how frequently it is reached, can have a significant impact on large sizes of set R.

The computational complexity of set-membership testing is also proportional to the number of layers h in a cascading cuckoo filter. Nevertheless, it is close to constant time since the value of h is small and does not vary significantly with variation in the cardinality of sets.

3.4 Threats and Mitigations

The SCRaaPS solution ensures robust and timely revocation status of X.509 certificates. The introduction of new components, like blockchain and probabilistic data structures, into the traditional Digital Certificate Infrastructure might open up new threat areas. Therefore, we list known potential threats and propose mitigations to each of these threats.

Threat 1: Authentication and Non-Repudiation

The open policy for submitting transactions and revocation lists to SCRaaPS can be exploited for launching Denial-of-Service Attacks. And, pseudo-anonymous account addresses leave room for Sybil attacks and false revocation information.

Mitigation A concrete account management policy will be designed to avoid any Certificate Authority (CA) from inundating the transaction pools of SCRaaPS miners. Each CA will be authenticated for a valid certificate chain leading up to well-known root certificate authority lists. The transaction validation mechanism will ensure that each submission contains a valid digital signature. This would ensure Non-Repudiation across the system and avoid Sybil attacks.

Threat 2: Blockchain Forking

In a peer-to-peer distributed storage platform, like that of blockchain technology, soft-forking occurs when there is a partition in the network. Public blockchain platforms like Bitcoin,

Ethereum, and EOS are susceptible to soft-forking. This will lead to inconsistent global state views of the blockchain data store.

Mitigation The LWM consensus algorithm is ideal for a public permissioned network. The leader-election process ensures that there is only one leader elected in each round and only the leader can propose a new block, leaving no room for possible soft forks. However, network partitions may still occur due to miner dropouts or inconsistent views of peer lists. Such threats are addressed in the later versions of the LWM consensus algorithm.

Threat 3: Availability and robustness in the event of DoS attacks

While the peer-to-peer distributed blockchain data store of SCRaaPS ensures the availability of revocation data in case of Distributed Denial-of-Service attacks, the requests for revocation statuses still need to be relayed to other peers for a response. This may cause delays in responses and encourage soft fails by web browsers.

Mitigation The client-side application of the SCRaaPS system that communicates will maintain a list of SCRaaPS endpoints for requesting revocation information of websites being visited. Furthermore, a group of SCRaaPS peers will be maintained by an authorized organization. These peers will be reliable and constantly monitored for faults and downtime. The list of peers maintained by the client-side application will include a subset of reliable peers.

Moreover, a request relaying mechanism will be designed and simulated for effectiveness. The results of the simulation will reveal expected delays in the event of DDoS attacks.

Threat 4: False Positive Rates in Cuckoo Filters

The cuckoo filter probabilistic data structure produces potential false positives, which may lead to misinterpretation of a certificate status as revoked when it might be valid.

Mitigation The probabilistic data structures compress the list of elements in a set to a constant size representation. This compression results in a loss of information bits, therefore, resulting in false positives. We will apply the cascading Bloom filter [73] approach to cuckoo filters in order to mitigate false positives. We will also explore other viable alternatives to mitigating false positives and select the best approach in terms of query latency and space efficiency.

3.5 Performance Metrics and Optimization

We evaluate SCRaaPS by recording performance metrics through experimentation and compare SCRaaPS with well-known revocation systems, like CRLSet [41], OneCRL [74], and CRLLite [8]. In this section, we list and define each metric.

- Efficiency

  Revocation mechanisms can be evaluated based on storage and memory requirements. The storage overhead caused by a cuckoo filter can be benchmarked for comparison. The size of the cuckoo filter itself does not vary with the total number of certificates revoked; therefore, the storage requirements are fixed. However, using the cascading mechanism of the cuckoo filter to avoid false positives will result in varying size requirements overall. Additionally, the communication requirements for updating the cuckoo filter will also be recorded to evaluate efficiency.

- Timeliness

  The current revocation methodologies follow periodic updates to CRLs and their subsets. The timeliness of disseminating revocation is critical for the effectiveness of any revocation mechanism. We will evaluate timeliness as the time it takes from the moment a certificate was revoked until the moment it appears as revoked for the major proportion of client devices. Based on observations, we may decide to optimize the update periods of the cuckoo filter.

- Fail-closeness

  The web browsers implemented the soft-fail feature for revocation status checking that arise from delayed and occasional failure in OCSP responses. The proprietary revocation mechanisms, like OneCRL and CRLSets, use specially aggregated subsets of CRLs but implement a fallback mechanism in times of failure to find the revocation status of a website. Such mechanisms are termed as fail-open. Instead, the CRLLite mechanism claims to be fail-closed without a need for a fallback mechanism. This is optimal, considering that

the fallback mechanisms have their own limitations and security considerations. We will evaluate the fail-closeness of SCRaaPS by experimenting with a large number of revoked X.509 certificates. This number is expected to be in the order of millions. Failure in determining the accurate revocation status in the event of a large number of certificates will prompt fallback mechanisms, which would negate fail-closedness.

- Transaction Latency

  The blockchain platform of SCRaaPS, which is adapted from the Scrybe Secure provenance system, must be able to securely validate and commit new transactions to the blockchain. The transaction latency of the blockchain platform will in turn impact the timeliness of revocation status. We will record and evaluate the average transaction latency of the underlying blockchain platform. This should ideally be in the order of a few seconds.

- Bandwidth (Transactions Per Second)

  The peer-to-peer decentralized blockchain platforms are limited by Buterin's scalability trilemma. The LWM consensus algorithm is known to be lightweight with less overhead and quick transaction commitments. We record the bandwidth of the underlying blockchain platform of SCRaaPS in terms of the platform's transactions per second capacity for benchmarking purposes.

3.6 Summary

The latest specifications for revocation mechanisms in X.509 standard is susceptible to DDoS attacks and delivers limited scalability. Due to fragmented CRL data, it is challenging to implement a client-side solution for revocation checking, which is essential for reducing overhead in computing platforms with limited resources like smartphones and tablet devices. In this chapter, we presented our approach to building a secure certificate revocation system to address these specific limitations and vulnerabilities. We also presented our design goals and described system architecture. Through this research work, we will answer two broad research questions: 1. Can a decentralized platform improve the robustness and effectiveness of current revocation

mechanisms? 2. Can the Set-Membership data structure enhance the lookup performance of certificate revocation in spite of potential false positives? Further, we listed known potential threats and associated mitigation strategies. The highly-available blockchain data store is resistant to DDoS attacks; however, there might be temporary service interruption in the event of successful DDoS attacks. The fail-close mechanism in SCRaaPS ensures that the client can still proceed with revocation checking using the latest version of ASMDS to avoid disruption of service. SCRaaPS clients may also choose to reach out to other SCRaaPS participants in the network if one participant remains unresponsive for long periods of time. We also presented cascading mechanisms to apply on ASMDs to mitigate potential false positives and defined metrics for recording the performance and scalability of our system. We capture and analyze these metrics during simulation and experimentation discussed in the following chapters.

CHAPTER 4

EXPERIMENTATION AND RESULTS

In this chapter, the experimentation methodology is described and results are presented. To begin with, the cascading technique is prototyped on Approximate Set-Membership Data structures (ASMDSs), including the cuckoo filter and XOR filter, and their computational time and space efficiency is evaluated through benchmarking. The cascading ASMDSs constructions are further optimized by formulating space requirements and improving parameter selection for reduced space. Subsequently, the revocation distribution system's operations are evaluated in terms of data storage costs and compute requirements. The total cost of system maintenance and operation is compared across different blockchain platforms.

4.1 Performance and Space-Efficiency Analysis of Cascading Cuckoo Filter

The cuckoo filter [18] probabilistic data structure is practically more space efficient than the optimal Bloom filter [18]. We prototyped the cascading cuckoo filter and ran benchmark tests with test SHA256 fingerprints to empirically evaluate the space efficiency. Experimentation began with deriving the asymptotic (upper and lower) bounds to space efficiency and benchmarking with test input (up to 100M SHA256 fingerprints). Further, the experimentation is repeated to compare results with that of CRLLite.

4.1.1 Asymptotic Behavior

A Cuckoo filter produces more space-efficient probabilistic data structures requiring $\log_2(\frac{1}{\epsilon})$ bits per item, where ϵ is the target false-positive rate [18]. On the other hand, an optimal Bloom filter requires $1.44\log_2(\frac{1}{\epsilon})$, which is 44% more bits per item. Expanding upon this key

metric, cascading Bloom filters are expected to occupy 44% more bits per item on average. The number of cascades will determine the final difference and given N number of cascades. The cascading Bloom filter is expected to occupy $44\% \times N$ more bits per item than the cascading cuckoo filter.

Given the total number of provisioned certificates u, out of which assuming r number of certificates are revoked, the size of the cuckoo filter at the first level can be calculated as $r \log_2(\frac{1}{\epsilon})$, and the size of the cuckoo filter at the second level can be calculated as $(u - r) \log_2(\frac{1}{\epsilon})$. The minimum total size of the cascading cuckoo filter can be derived as:

$$r \log_2(\frac{1}{\epsilon_r}) + (u-r) \log_2(\frac{1}{\epsilon_{u-r}}) + r\epsilon_r \log_2(\frac{1}{\epsilon_r}) + (u-r)\epsilon_{(u-r)} \log_2(\frac{1}{\epsilon_{u-r}}) + r\epsilon_r \times \epsilon_{r\epsilon_r} \log_2(\frac{1}{\epsilon_{r\epsilon_r}}) + ... \tag{4.1}$$

4.1.2 Construction and Benchmarking Setup

The cascading cuckoo filter was prototyped using C++ programming language. It builds the cuckoo filter objects using the C++ library developed by the original authors of the cuckoo filter [18]. The algorithm design for building cascades implements the pseudocode mentioned in 2. 100 Million SHA256 fingerprints of random alphanumeric fixed-length strings (prepared using the CryptoPP [75]) were generated as test input and filled in a C++ set [76] to guarantee uniqueness. These SHA256 fingerprints accurately represent, in size and data type, the actual input of SHA256 fingerprints [34] of X.509 certificates.

Assuming that the set U represents the total provisioned certificates that are active at a particular point in time, the set R represents the list of revoked certificates that remain active (unexpired), the set S represents the list of valid active certificates, then the percentage of revoked certificates can be represented as $\delta_{rev} = \frac{|R|}{|U|} \times 100$. The benchmarking code captures the following key metrics: computational time taken (in wall-clock time milliseconds) for building the cascading filter and looking up random arbitrary elements and the total space occupied (in storage Bytes) by the final cascading filter. Results are captured for varying δ_{rev}, including 8%, 25%, 50%, and 75%. On average, 8% of total provisioned X.509 certificates are expected to be revoked therefore metrics

were captured with $\delta_{rev} = 8$. The experimentation code begins by generating SHA256 fingerprints of random fixed-length strings (10 chars) and fills a set of size $|U|$. Set R is then initialized to accommodate $\delta_{rev} \times |U|$ fingerprints and filled with elements from set U. The remaining elements are filled into a separate set S of size $|U| - |R|$.

Probabilistic data structures, such as the cuckoo filter, use fixed-length arrays (two-dimensional array of fingerprints in the case of a cuckoo filter [18] and a bit-array in the case of a Bloom filter [64] and XOR filter [19]) for representing an input set and, as it is the case with the *array* data structure, the arrays must be initialized to a fixed-size before adding elements. The size that these arrays are initialized impacts the overall *load factor*, *false-positive rate*, and *space usage*. Hence, given the knowledge of the final size of the input set, the configuration parameters of the probabilistic data structures (such as *size of initialized filter array*, *bits per item*, and *number of hash functions*) can be optimized to reduce the resulting total space of the probabilistic data structure.

At the beginning of the experimentation, a simple approach to selecting configuration parameters was followed with the top-level cascade being initialized to support the size of set R. The subsequent cascades were initialized to 3% of the corresponding input's set size as cuckoo filters are proven to utilize fewer bits per item in comparison to Bloom filters for false-positive rates below 3%. Performance and space-efficiency metrics were initially captured using these default configuration parameters prior to optimization.

4.1.3 Testbed

All experiments were run on the Firefly cluster at the University of Tennessee at Chattanooga where each node is equipped with an Intel(R) Xeon(R) Gold 6148 CPU with a maximum clock speed of 2.40GHz. This CPU also had L1d cache and L1i cache of 32K each, L2 cache of 1024K, and L3 cache of 28160K. Experiment jobs were provisioned using SLURM [77] job scheduler requesting a maximum of 12 GB memory. To avoid the administrative overhead of installing C++ library dependencies on the cluster, a Singularity [78] container was prepared with all the required C++ library dependencies and used during the SLURM job submission.

4.1.4 Initial Results and Observations

The cascading cuckoo filter (CCF) is a constant-time set membership data structure that produces zero false positives (and zero false negatives). Structurally, it is a vector of cuckoo filters (of fixed size) representing a set U of elements where a subset (set R) of elements are to be tested for membership. Although the cuckoo filter is a probabilistic data structure that produces false positives at a small rate, the iterative cascading technique is guaranteed to terminate and the CCF can provide a deterministic evaluation of whether an arbitrary element ($\in$ set U) belongs to the subset (set R) in computational order of constant time.

Initial results were captured using the construction and benchmarking efforts described in Sec 4.1.2 for a 100M test SHA256 fingerprints to observe the growth in total space and performance (construction time & lookup speed) of cascading cuckoo filters with varying bits per fingerprint setting and for varying percentages of revocation subset size (in comparison to total provisioned – 100M). *Total space* was calculated by adding the space occupied by each cuckoo filter in the cascade vector and *average bits per item* was calculated by dividing the total size of set U with the total size of the CCF. Construction time was captured using the C++ std::high_resolution_clock [79] binary. Time points were recorded before beginning the building of CCF and after the completion of the CCF building, and the difference is recorded in wall clock time (milliseconds). Lookup speed was evaluated by capturing the duration of validating zero false positives and zero false negatives (through a lookup of each item in set R and set S) and dividing this duration (measured in the count of wall clock milliseconds) by the total size of set U.

As a cuckoo filter enables dynamic addition and deletion of elements (instead of bulk insert at once), the generation of CCF avoids the need for any additional memory to store the intermittent set of false positives identified at each cascade level. However, the cuckoo filter must be initialized to fit a set of fixed sizes and will begin producing false positives as the load percentage increases. Conversely, over-provisioning the cuckoo filter may result in a waste of space at each level. Hence, the choice of initialized size of a cuckoo filter cascade impacts the total size of the CCF. Since cuckoo filters are more space efficient than Bloom filters for false-positive rates below 3%, each cuckoo filter is initialized to fit 3% of the set that is tested for false positives at each cascade level.

Table 4.1

Performance and space-efficiency comparison of cascading cuckoo filters. Results were captured using a list of 100M SHA256 fingerprints that were randomly generated to simulate 100M total provisioned certificates ($\in$ set U) among which 8% (8M) were assumed to be valid and revoked ($\in$ set R)

	Size(bytes)	**bits/item**	**Construction Time (ms)**	**Lookup speed (items/ms)**
CCF8	**17,434,896**	**17.43**	183,577	1,087
CCF12	19,074,048	19.07	176,985	1,117
CCF13	19,267,605	19.27	176,738	1,084
CCF16	25,427,968	25.43	176,738	1,084
CCF32	33,554,432	33.55	**161,716**	**1,118**

Cuckoo filters can be built in two variations and with different bits per fingerprint setting [18]. The simple cuckoo filter uses the Single Table for arranging the fingerprints, and the optimized cuckoo filter uses the Packed Table semi-sorted approach for arranging the fingerprints. Results were collected for CCF of the simple cuckoo filter with bits per fingerprint of 8, 12, 16, and 32 and of the semi-sorted cuckoo filter with 13 bits per fingerprint. As shown in Table 4.1, for a subset of 8% revoked fingerprints, the CCF with 8 bits per fingerprint setting produces the CCF of the smallest size among the other types using just 17 bits per item on average. As cuckoo filters with larger bits per fingerprint settings produce lower numbers of false positives and accommodate larger sets, the CCF with 32 bits per fingerprint is the fastest CCF to produce (161716 ms) and yields the highest lookup speed (1118 items/ms).

The total size of CCF is directly proportional to the difference in sizes of set S and set R. As the difference between the sizes of set S and set R reduces, the space efficiency of the CCF increases. Table 4.2 shows the performance and space efficiency results captured with 25% revoked SHA256 fingerprints (set R), where the CCF with 8 bits per fingerprint uses 13.8 bits per item on average in comparison to the 17.42 bits per item with 8% revoked fingerprints (see Table 4.1). Moreover, the fastest CCF (using 32 bits per fingerprint) is faster in the case of 25% revoked fingerprints than 8% revoked fingerprints.

Table 4.2

Performance and space-efficiency comparison of cascading cuckoo filters. Results were captured using a list of 100M SHA256 fingerprints that were randomly generated to simulate 100M total provisioned certificates ($\in$ set U) among which 25% (8M) were assumed to be valid and revoked ($\in$ set R)

	Size(bytes)	**bits/item**	**Construction Time (ms)**	**Lookup speed (items/ms)**
CCF8	**43,061,408**	**13.78**	200,868	1,002
CCF12	57,412,608	18.37	193,754	1,002
CCF13	57,409,557	18.37	196,123	993
CCF16	76,546,048	24.49	193,614	958
CCF32	134,217,728	42.95	**159,297**	**1,125**

Table 4.3

Performance and space-efficiency comparison of cascading cuckoo filters. Results were captured using a list of 100M SHA256 fingerprints that were randomly generated to simulate 100M total provisioned certificates ($\in$ set U) among which 50% (8M) were assumed to be valid and revoked ($\in$ set R)

	Size(bytes)	**bits/item**	**Construction Time (ms)**	**Lookup speed (items/ms)**
CCF8	**73,449,664**	**11.75**	224,228	879
CCF12	10,538,424	16.86	219,494	894
CCF13	10,528,909	16.86	219,804	890
CCF16	14,509,184	22.48	158,014	893
CCF32	268,435,456	42.95	**158,014**	**1,104**

CCF produces the most efficient results when the sizes of set R and set S (See Eq. 4.1.1 for asymptotic lower bounds on CCF's total space) are similar. And as shown in Table 4.3, the most space-efficient results with the least average bits per fingerprint in CCF are observed with 50% revoked fingerprints $\in$ set R (and 50% valid $\in$ set S). Performance speeds with 50% revoked fingerprints are also similar to the best results (with 25%) revoked fingerprints but are not the best given the higher load factor and a high number of kicks resulting from a higher number of items inserted into cuckoo filters at each layer.

Table 4.4

Performance and space-efficiency comparison of cascading cuckoo filters. Results were captured using a list of 100M SHA256 fingerprints that were randomly generated to simulate 100M total provisioned certificates ($\in$ set U) among which 75% (8M) were assumed to be valid and revoked ($\in$ set R)

	Size(bytes)	**bits/item**	**Construction Time (ms)**	**Lookup speed (items/ms)**
CCF8	**140,533,808**	**14.99**	246,352	816
CCF12	206,045,184	21.98	246,732	806
CCF13	206,045,205	21.98	248,407	811
CCF16	274,726,912	29.3	244,206	797
CCF32	536,870,912	57.27	**159,558**	**1,065**

When the size of set R becomes larger than the set S, the number of false positives found at each cascade layer is lower (as opposed to when |set R|<|set S|) as the input set to search for false positives is smaller. However, a larger set R results in a larger base layer cuckoo filter and potentially larger and more subsequent layers of cuckoo filters due to false positives emerging from set R. Table 4.4 shows benchmark results for 75% revoked fingerprints where the CCF with 32 bits per fingerprint is the largest in size and average bits per item in comparison to its results for 50% or 25% fingerprints revoked.

4.2 Performance and Space-Efficiency Analysis of the Cascading XOR Filter

The more recently proposed XOR filter [19] is shown to be more space-efficient and faster in comparison to cuckoo filters and Bloom filters. By logical extension, the cascading XOR

filter is also expected to be relatively more space-efficient and faster and it was prototyped and benchmarked to achieve an empirical comparison.

4.2.1 Asymptotic Behavior

The authors of the XOR filter showed that the minimum *bits per fingerprint* required by the XOR filter is $1.23\log_2(\frac{1}{\epsilon})$ [19] and is l bitwise in overhead in comparison to that of the Bloom filter ($1.44\log_2(\frac{1}{\epsilon})$). It is, however, larger bitwise in comparison to that of the cuckoo filter ($\log_2(\frac{1}{\epsilon})$). But, the practical crossover point where the cuckoo filter produces is more space efficiency is 5.6×10^{-6}, which is rarely applicable to real-world use cases.

Given the total number of provisioned certificates u, out of which assuming r number of certificates are revoked, the size of the cuckoo filter at first level can be calculated as $r \times 1.23\log_2(\frac{1}{\epsilon})$ and the size of the cuckoo filter at the second level can be calculated as $(u-r) \times 1.23\log_2(\frac{1}{\epsilon})$. The minimum total size of the cascading cuckoo filter can be derived as:

$$\begin{aligned} r \times 1.23 \times \log_2(\frac{1}{\epsilon_r}) + (u-r) \times 1.23 \times \log_2(\frac{1}{\epsilon_{u-r}}) + r\epsilon_r \times 1.23\log_2(\frac{1}{\epsilon_r}) \\ +(u-r)\epsilon_{(u-r)} \times 1.23\log_2(\frac{1}{\epsilon_{u-r}}) + r\epsilon_r \times 1.23\epsilon_{r\epsilon_r}\log_2(\frac{1}{\epsilon_{r\epsilon_r}}) + ... \end{aligned} \tag{4.2}$$

4.2.2 Construction and Benchmarking Setup

The cascading XOR filter (CXF) was prototyped using the C++ programming language. It builds the XOR filter objects using the XOR filter C++ library [80] developed by the original authors of the XOR filter [19]. The cascading algorithm from 3.3.2 was applied to build the XOR filter cascades representing a set U of 100M randomly generated SHA256 fingerprints. Space-efficiency and performance results were captured for different variations of the XOR filter and for different percentages of revoked certificates ($\in$ set R) among the total provisioned certificates.

The XOR filter library supports input elements of integer type only and does not support the insertion of SHA256 string digests [80]. For this reason, the benchmarking algorithm converts the SHA256 string digests into unique integers using the Zobrist Hashing [81] perfect hash function

[82]. The time for hashing SHA256 string digests is also included in benchmarking algorithms that record construction time and lookup speed. Each benchmarking algorithm also validates all elements in set U by performing lookups to ensure that there are no false positives or false negatives. Since the XOR filter only supports bulk inserts [19], and the memory space allocation is optimized based on the size of the input set, there is no further optimization possible to minimize the size of a cascading XOR filter. The same testbed environment described in Sec 4.1.3 was used to collect benchmarking results of CCFs.

4.2.3 Results and Observations

Performance and space-efficiency results were captured using the construction and benchmarking efforts described in Sec 4.1.2 for 100M test SHA256 fingerprints. Patterns in performance and space efficiency were captured to observe the growth in total space and performance (construction time & lookup speed) of cascading XOR filters with varying bits per fingerprint setting and for varying percentages of revocation subset size (in comparison to total provisioned – 100M). *Total space* was calculated by adding the space occupied by each XOR filter in the cascade vector. *Average bits per item* was calculated by dividing the total size of set U with the total size of the CXF. Construction time was captured using the C++ std::high_resolution_clock [79] binary. Time points were recorded before beginning the building of CXF and after the completion of the CXF building, and the difference is recorded in wall clock time (milliseconds). Lookup speed was evaluated by capturing the duration of validating zero false positives and zero false negatives (through a lookup of each item in set R and set S) and dividing this duration (measured in the count of wall clock milliseconds) by the total size of set U.

The XOR filter [80] can be built to allocate 8 bits per fingerprint or 16 bits per fingerprint, and its authors have also built an optimized version of the XOR filter using a buffered population approach called the XOR+ filter. Separately, the same authors have also designed and implemented the binary fuse filter [20], a more space-efficient alternative to XOR filters inspired by [83], that sacrifices query speed to produce probabilistic set-membership data structures of a smaller size and with fewer false positives. Benchmarking results of CXF were captured for different variations

Algorithm 4: Benchmarking space efficiency and performance of cascading cuckoo filters

Data: $U \geq 0$; $R \geq 0$
set$U \leftarrow$ GenSHA2Fingerprints(size: U) ; `/* Random SHA256 fingerprints */`
set$R \leftarrow$ CopySet(set$U$, 0, R)
set$S \leftarrow$ CopySet(set$U$, R, $U - R$)
$t_1 \leftarrow$ high_resolution_clock::now()
$CCF \leftarrow$ BuildCFCascades(set$R$, set$S$)
$t_2 \leftarrow$ high_resolution_clock::now()
$size_{CCF} \leftarrow 0$
foreach *cf: CCF* **do**
    $size_{CCF}$ += $cf.SizeInBytes()$; `/* Add the size of each filter cascade */`
end
$t_{construction} \leftarrow (t_2 - t_1)$
$t_1 \leftarrow$ high_resolution_clock::now()
$fp \leftarrow 0; fn \leftarrow 0$
foreach *f: set* R **do**
    if *CCF.Lookup(f)==False* **then**
        fn++;
    end
end
foreach *f: set* S **do**
    if *CCF.Lookup(f)==True* **then**
        fp++;
    end
end
$t_2 \leftarrow$ high_resolution_clock::now()
$t_{lookupspeed} \leftarrow \frac{(t_2 - t_1)}{U}$
assert(fp== 0 && fn== 0) ; `/* Confirm no false positives/negatives */`
return $size_{CCf}$, $t_{construction}$, $t_{lookupspeed}$

Table 4.5

Performance and space-efficiency comparison of cascading XOR filters for distributing X.509 certificate revocation lists. Results were captured using a list of 100M SHA256 fingerprints that were randomly generated to simulate 100M total provisioned certificates ($\in$ set U) among which 8% (8M) were assumed to be valid and revoked ($\in$ set R) and the rest are valid and non-revoked

	Size(bytes)	**bits/item**	**Construction Time (ms)**	**Lookup speed (items/ms)**
XOR8	10,321,884	10.32	353,698	1,286
XOR16	19,684,014	19.68	358,339	1,388
XOR+8	10,322,727	10.32	349,132	1,312
XOR+16	19,683,894	19.68	338,697	**1,409**
Fuse8	**9,547,752**	**9.548**	**326,098**	1,066

of XOR and fuse filters in combination with different bits per fingerprint setting to compare performance trade-offs and total sizes. As shown in Table 4.5, in the case of 100M provisioned SHA256 fingerprints and 8% revoked, the fuse filter using 8 bits per fingerprint produces the most space-efficient CXF that is slightly smaller than the one produced with the XOR filter using the same 8 bits per fingerprint. However, as expected, the fuse 8 filter is comparatively slower in query speed, and the XOR filter using 16 bits per fingerprint produces the CXF with the fastest query speeds as there are lower numbers of cascades in its CXF due to allocating higher bits per fingerprint.

Similar to our observations with cascading cuckoo filter benchmarking, the cascading XOR filter improves in space efficiency as the size of set R gets closer to the size of set S. As shown in Table 4.6 (showing performance and space-efficiency metrics of CXF with set R of size 25M and set S of size 75M), the fuse filter using 8 bits per fingerprint produces more space-efficient CXF in the case of 25% revoked fingerprints than in the case of 8% revoked fingerprints. The best cases for space-efficiency and query speeds are consistent in the case of 25% revoked fingerprints with that of 8% revoked fingerprints where the fuse filter using 8 bits per fingerprint still produces the most space-efficient CXF of the smallest size, and the XOR filter using 16 bits per fingerprint produces the CXF with the fastest query speeds.

Table 4.6

Performance and space-efficiency comparison of cascading XOR filters for distributing X.509 certificate revocation lists. Results were captured using a list of 100M SHA256 fingerprints that were randomly generated to simulate 100M total provisioned certificates ($\in$ set U) among which 25% (25M) were assumed to be valid and revoked ($\in$ set R) and the rest valid and non-revoked

	Size(bytes)	**bits/item**	**Construction Time (ms)**	**Lookup speed (items/ms)**
XOR8	31,231,192	9.994	385,644	1,529
XOR16	61,504,080	19.68	370,746	1,927
XOR+8	31,233,513	9.995	367,946	1,538
XOR+16	61,504,068	19.68	359,335	**1,941**
Fuse8	**28,930,224**	**9.258**	**341,992**	1,481

Table 4.7

Performance and space-efficiency comparison of cascading XOR filters for distributing X.509 certificate revocation lists. Results were captured using a list of 100M SHA256 fingerprints that were randomly generated to simulate 100M total provisioned certificates ($\in$ set U) among which 50% (50M) were assumed to be valid and revoked ($\in$ set R) and the rest valid and non-revoked

	Size(bytes)	**bits/item**	**Construction Time (ms)**	**Lookup speed (items/ms)**
XOR8	61,984,596	9.918	413,598	1,969
XOR16	123,004,026	19.68	400,306	2,092
XOR+8	61,983,756	9.917	413,793	1,978
XOR+16	123,003,984	19.68	**397,889**	**2,124**
Fuse8	**57,328,656**	**9.173**	504,971	19,245

When the size of set R is similar to that of set S (in the case of 50% revoked fingerprints), CXFs become more space efficient with higher lookup speeds. As observed in Table 4.7, the CXFs are more space efficient and support faster query speeds when 50% of the total provisioned fingerprints are revoked.

Table 4.8

Performance and space-efficiency comparison of cascading XOR filters for distributing X.509 certificate revocation lists. Results were captured using a list of 100M SHA256 fingerprints that were randomly generated to simulate 100M total provisioned certificates ($\in$ set U) among which 75% (75M) were assumed to be valid and revoked ($\in$ set R) and the rest valid and non-revoked

	Size(bytes)	**bits/item**	**Construction Time (ms)**	**Lookup speed (items/ms)**
XOR8	92,734,757	9.891	**456,187**	2,381
XOR16	184,504,050	19.68	505,834	2,434
XOR+8	92,733,807	9.892	549,257	2,381
XOR+16	184,504,014	19.68	562,486	**2,503**
Fuse8	**85,830,108**	**9.155**	522,119	2,223

The cascading XOR filters have consistently outperformed the cascading cuckoo filters in terms of space efficiency, construction speed, and lookup speeds. Based on these initial results with default (non-optimized) CCF and CXF constructions, building a CXF using the fuse filter with 8 bits per fingerprint is the recommended choice for representing the overall set of X.509 certificate revocation lists. For the rest of this experimentation, we focus on the fuse filter with 8 bits per fingerprint.

4.3 Updates and Differentials

The cascading fuse filter (CFF) is highly space efficient, fast to construct, and produces zero false positives. The cascading algorithm applied in CFF guarantees zero false positives only to a fixed set of total provisioned certificates and cannot guarantee zero false positives with dynamic additions or deletions. Hence, the cascading fuse filter must be regenerated with each change to the total set of provisioned certificates (additions/expirations to revoked/non-revoked certificates). For this reason, the fuse filter is the preferred choice for cascading probabilistic data

structures in SCRaaPS, on the other hand, the cuckoo filter that supports dynamic additions and deletions (whereas the fuse filter doesn't) would have been a suitable option to optimize space and computation. In this section, we analyze the growth in space of cascading fuse filters for varying rates of changes in the global set of provisioned certificates.

The set of SHA256 fingerprints of total provisioned certificates (set U) can change in one of four ways:

1. Valid certificate is issued.
2. Valid certificate expires.
3. Valid certificate is revoked.
4. Revoked certificate expires.

Assuming that certificates are issued at rate at r_{iss} per day, valid certificates expire at rate $r_{val-exp}$ per day, valid certificates are revoked at the rate r_{rev} per day, and revoked certificates expire at $r_{rev-exp}$. The sets change as follows:

$$\delta(|setU|) = r_{iss} - r_{val-exp} - r_{rev-exp} \tag{4.3}$$

$$\delta(|setR|) = r_{rev} - r_{rev-exp} \tag{4.4}$$

$$\delta(|setS|) = r_{iss} - r_{val-exp} - r_{rev} \tag{4.5}$$

Until recently (circa 2017), the average age of X.509 certificates was 3 years, and the set of total unexpired certificates was $\sim$43 million [8]. The advent of Let's Encrypt and the general change in trend towards shorter validity in X.509 certificates [84] has reduced the average age of certificates to $\sim$1 year. Hence, more certificates are issued than are being revoked ($r_{iss} \gg r_{rev}$), and more certificates are expiring now than they used to in 2017.

To observe the size of cascading fuse filters and differential updates (bitwise difference needed to update the previous block's version of cascading fuse filters to the latest version) with periodic issuance of new certificates and revocation of valid certificates, the sizes of resulting cascading fuse filters were recorded by simulating updates to set U and by subsequently evaluating

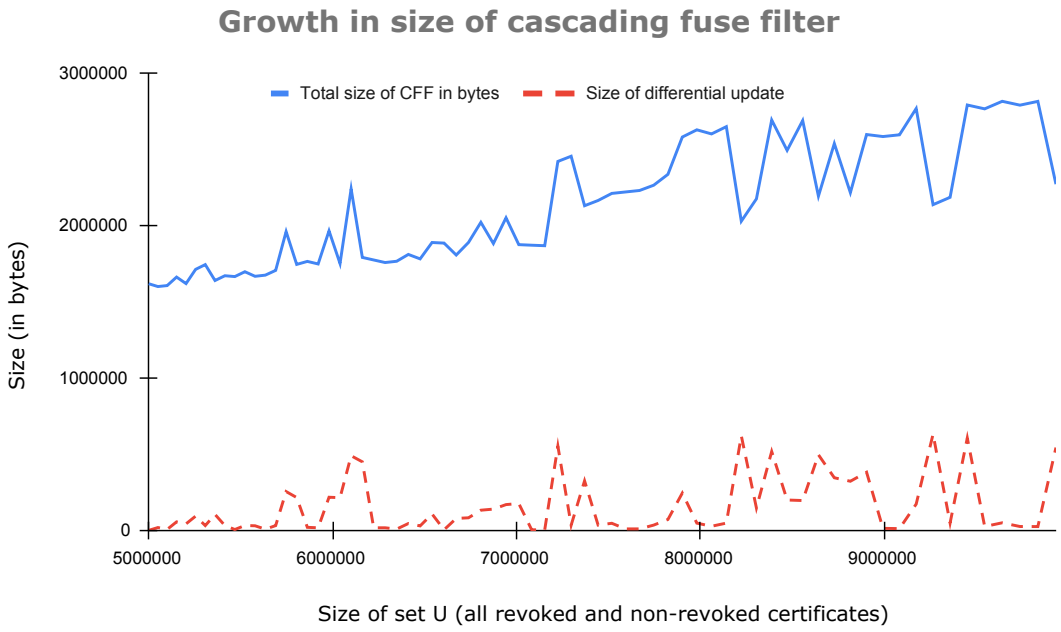

Figure 4.1 Growth in size of a cascading fuse filter and its differential updates assuming 1% certificates are newly issued, 0.1% certificates are revoked, 0.01% valid certificates expired, and 0.001% revoked certificates expired. Sizes of cascading fuse filters were captured beginning with a total of 5M total provisioned certificates among which 1.25M (25%) were revoked and 3.75M (75% were valid)

the corresponding new size of cascading fuse filters and the size of differential updates. As shown in Fig. 4.1, the size of a cascading fuse filter grows linearly in size with each change to the set U while roughly maintaining similar space efficiency close to 9.2 bits per revoked fingerprint. The differential updates are observed to be $\sim$146 KB on average with as low as 11KB and occasionally as high as $\sim$630 KB. We observed that large differential updates have been repeatedly found when the new cascading fuse filter is significantly lesser in size than the previous filter, which indicates that the previous cascading fuse filter was over-provisioned and with growth in the size of set U, the cascades had reached their maximum capacity. The over-provisioning of fuse filters can be avoided by predicting the pattern of growth and choosing a size that accommodates size fluctuations in the short term. Optimizing the fuse filter configuration parameters is a promising area for future work and can further reduce the total size. However, it must be noted that such optimization efforts must consider general trends in changes to the sets and must avoid over-optimizing to avoid large differential updates in extraordinary circumstances, such as the aftermath of the Heartbleed bug [9] where a large percent of certificates were revoked instantaneously.

4.4 Summary

In this chapter, we explained our experimental setup and presented benchmarking results. We implemented the cascading algorithm [8] on newer approximate set-membership data structures including the cuckoo filter, XOR filter, and fuse filter. Then we recorded resource utilization in building cascades given varying sizes of revocation lists. All three approximate, set-membership data structures produce zero false positives when the cascading algorithm is applied, and the iterative process of building cascades successfully terminates as expected. Benchmarking results with a cascading cuckoo filter show that using a cuckoo filter with 8-bits per fingerprint produces the most space-efficient cuckoo filter cascades regardless of the percentage of certificates revoked (8-75%) and using the cuckoo filter with semi-sorted buckets does produce smaller cascades. Subsequently, we found that the cascading XOR filter is more space efficient (requiring 9.8 - 10.32 bits per revoked SHA256 fingerprint) than the cascading cuckoo filter (requiring 11.75

- 17.43 bits per revoked SHA256 fingerprint), and the cascading fuse filter is the most space-efficient choice for the secure certificate revocation system for quick, constant time lookup of revocation information. The experimentation and benchmarking efforts in this chapter elucidate the difference in practical results and asymptotic expectations as the cuckoo filter should in theory produce more space-efficient cascades (requiring $\log_2 \frac{1}{\epsilon}$ bits per fingerprint where ϵ is false positive rate) than the XOR filter (requiring $1.23 \log_2 \frac{1}{\epsilon}$). Further, growth in size of cascading fuse filters and differential updates were observed by simulating periodic issuance and revocation of certificates. Results show that the cascading fuse filter increases linearly in total size with an increase in the total set of issued certificates, and the differential updates are generally small ($\sim$ 146 KB on average) and occasionally increase in size. Hence, the cascading fuse filter is a more space-efficient alternative to a cascading Bloom filter and other cascading approximate data structures due to its high space efficiency and high computational efficiency with the minimum requirement for optimizing parameters according to the input set.

CHAPTER 5

ANALYSIS OF SCRAAPS SYSTEM

In this chapter, the SCRaaPS system is analyzed based on the experimentation results from the previous chapter. Server-side and client-side computational requirements are evaluated and compared with other alternatives. Security analysis is then presented describing potential threats and how SCRaaPS mitigates them. Finally, SCRaaPS results are compared with that of current solutions such as CRLLite, OneCRL, and CRLSet.

5.1 Server-side Computational Requirements

The SCRaaPS system is meant to be a lightweight alternative to OCSP server operations and is meant to serve X.509 certificate revocation status to arbitrary public clients. In this section, the SCRaaPS server-side, computational requirements are analyzed and discussed.

SCRaaPS uses blockchain networks for distributing X.509 certificate revocation lists. Aggregators are additionally responsible for generating the cascading fuse filter for quick revocation checking by browser clients.

Benchmarking results from Section 4.1 and 4.2 show that the fuse filter is the most space-efficient choice for representing the global CRL lists aggregated on the SCRaaPS system. In 2017, Larisch et al. [8] reported 30 million valid non-revoked existed and 12.7 million valid, revoked certificates were present in various CRLs distributed by the CAs. Given these numbers, Table 5.1 shows that the fuse filter with 8 bits per fingerprint is the most space-efficient choice with an approximate size of $\sim$14MB (9.223 bits per revoked fingerprint) on average.

The Lightweight Mining (LWM) consensus engine enables SCRaaPS to be a cost-effective network for aggregating SHA256 fingerprints of revoked certificates. Since LWM does not

Table 5.1

Performance and space-efficiency comparison of cascading XOR filters, cascading cuckoo filters, and cascading fuse filter for distributing X.509 certificate revocation lists. Results were captured using a list of 42.7M SHA256 fingerprints that were randomly generated to simulate 42.7M total provisioned certificates ($\in$ set U) among which 29.7% (12.7M) were assumed to be valid and revoked ($\in$ set R) and the rest valid and non-revoked

	Size(bytes)	**bits/item**	**Construction Time (ms)**
CXOR8	15,827,373	9.97	276,088
CXOR16	31,243,728	19.68	273,580
CXOR+8	15,827,754	9.97	280,476
CXOR+16	31,243,950	19.68	270,919
CFuse8	**14,641,248**	**9.223**	140,274
CCuckoo8	18,375,316	11.58	118,606
CCuckoo12	25,313,430	15.95	101,635
CCuckoo13	25,239,623	15.9	100,671
CCuckoo16	33,566,720	21.14	99,301
CCuckoo32	67,108,864	42.27	**82,793**

implement cryptocurrency and participation rewards, there is no additional cost for using the network. Moreover, there is low computational overhead for participating in the LWM consensus. As shown in Table 5.2, the ongoing dollar cost of committing and updating the cascading fuse filter on Ethereum is US$560 (at the time of writing this document) which is significantly more than operating SCRaaPS node participants. Algorand and EOS are cost-effective blockchain solutions in comparison to Ethereum, however, they require a larger amount of computational resources, such as 200 GB SSD and 8 GB RAM, for operating full-node blockchain participants. The need for such high amount of computational resources makes Algorand and EOS unsuitable in resource-constrained systems such as Vehicle-to-Vehicle communication and IoT environments. Alternatively, the Hyperledger [85] framework provides a similar technology to that of SCRaaPS but in a permissioned network setting restricted to authorized participants. In this case, the dollar cost of committing and maintaining the cascading fuse filter is comparable to that of using the SCRaaPS blockchain but the storage requirement is double than that of SCRaaPS.

Table 5.2

Cost analysis and compute requirements for using different blockchain networks to distribute the Cascading Fuse filter

Blockchain network	**Network access type**	**Additional cost for storing CFF**	**Disk requirements**	**Memory requirements**
Ethereum	Public	$560	100 GB SSD	4-8 GB RAM
Algorand	Public	$0.01	200 GB SSD	4-8 GB RAM
SCRaaPS	Public-permissioned	none	10GB HDD	2-4GB RAM
EOS	Public-permissioned	$4.48	10GB HDD	2-4GB RAM
Hyperledger	Permissioned	none	20GB HDD	4-8GB RAM

Table 5.3

SCRaaPS system comparison with CRLLite, OneCRL, and CRLSet

	Total Revoked certs	**bits per item**	**Fail-closed**	**Trust requirements**	**Authority**
CRLSet	∼14k	110	No	Centralized	Google
OneCRL	∼357	1,928	No	Centralized	Mozilla Foundation
CRLLite	∼12M	6.6	Yes	Centralized	Mozilla Foundation
SCRaaPS	∼12M	9.2	Yes	Decentralized	None

5.2 Security Analysis

Here we analyze the security of the SCRaaPS system.

DoS resistance. SCRaaPS inherits security features from Scrybe's Lightweight Mining (LWM) consensus algorithm and relies on it for network liveness [86] and availability [87] guarantees. Altarawneh et. al. [86] proved that the Lightweight Mining consensus algorithm enables the network to withstand up to $\frac{N}{3} - 1$ misbehaving participants (for N total participants). Attacks to overwhelm the transaction pool are also covered in [86] and threats to the consensus algorithm not covered in [86] are beyond the scope of this research. Recall from Section 3.2 that the transactions in SCRaaPS include the CA's public key, CA certificate, CA certificate chain, and digital signature. And SCRaaPS block proposers and validators process the transaction by validating the submitting CA's signature along with the transaction's digital signature. This prevents rogue CAs from overwhelming the ledger storage to cause Denial-of-Service attacks.

Man in the middle attacks (MiTM). SCRaaPS enables client devices to download block headers and maintain metadata to validate the integrity of new blocks. Each block in SCRaaPS also includes digital signatures of each transaction and consensus metadata (such as the hashes and corresponding random numbers exchanged during LWM phases), which can be used to validate the integrity of blocks and avoid any MiTM attacks. Communication between the consensus participants and pertaining to consensus is secured by the Lightweight Mining algorithm. SCRaaPS consensus protocol ensures participants exchange cryptographic keys and X.509 certificates to establish trust and secure subsequent communication. Consensus (Overlay) network management operations and their security are beyond the scope of this research and are assumed to be covered by the Lightweight Mining algorithm (LWM).

Misbehaving or Compromised CAs. The SCRaaPS consensus protocol includes transaction processing operations where each consensus participant ensures that each transaction includes a valid digital signature by a valid certificate authority (whose X.509 certificate and certificate chain are valid). Further, the CA's certificate identifier is checked against the previous block's cascading ASMDS to check the revocation status of a CA before accepting a transaction from this CA. This avoids any revoked CA to submit or tamper with the SCRaaPS blockchain ledger. It is

possible that an attacker can compromise a CA and submit invalid SHA256 fingerprints in an effort to flood the blockchain and grow the size of cascading fuse filter significantly. However, this would be considered an anomalous pattern and will only alert the network of a compromised CA and lead to revoking that certificate. On the other hand, a compromised CA can intentionally revoke a good certificate to launch a denial-of-service attack. However, this would also alert the party whose certificate was revoked and will result in revoking the compromised CA. Hence, misbehaving or compromised CAs are not a threat to SCRaaPS, and attackers would rather issue new certificates with a compromised CA's cryptographic key than revoke any certificate.

5.3 Summary

In this chapter, we discussed the results of experimentation and benchmarking of cascading probabilistic data structures. SCRaaPS provides a more space-efficient alternative to current revocation methodologies with low computational requirements and uses just 9.25 bits per revoked SHA256 fingerprint to provide constant-time, fail-closed, zero-false-positive revocation query response. The Lightweight Mining (LWM) consensus engine lowers the server-side computational requirements for deploying SCRaaPS blockchain software with no additional cost (in cryptocurrency) to submit transactions. The low-cost, highly-available SCRaaPS system helps unify the revocation information in a PKI environment allowing the Certificate Authorities to collaborate in an untrusted, p2p network. Therefore, enabling an exhaustive collection of X.509 certificate info and a fail-closed system for revocation status checking. Moreover, by committing a cascading fuse filter and its differential updates, SCRaaPS provides its clients with a succinct set-membership data structure to independently validate the revocation status of an arbitrary certificate. This is especially useful for SCRaaPS clients to avoid service disruption in the case of Denial-of-Service (DoS) attacks or Man-in-the-Middle (MiTM) attacks.

CHAPTER 6

CONCLUSION AND FUTURE WORK

The revocation system proposed in the X.509 standard produces fragmented revocation lists and fail-open systems making it susceptible to Denial-of-Service (DoS) and Man-in-the-Middle (MiTM) attacks. In this dissertation, we presented the Secure Certificate Revocation as a Peer Service (SCRaaPS) for robust and reliable validation of X.509 certificate revocation. SCRaaPS provides a blockchain-based, distributed storage network for collecting and distributing X.509 certificate information. SCRaaPS additionally includes a cascading fuse filter (designed, prototyped, and benchmarked in this dissertation) for constant-time, zero-false-positives lookup of X.509 revocation status.

Developed as a customized version of the Scrybe blockchain protocol, SCRaaPS incorporates Scrybe's decentralization, scalability, and high availability attributes. It provides a peer-to-peer platform for Certificate Authorities (CAs) to collaborate in an untrusted environment and unify certificate revocation information. The Lightweight Mining (LWM) consensus engine (inherited from Scrybe) avoids a single point of failure and the need for trust establishment between the collaborating CAs. Further, the cascading fuse filter added to the blockchain-based data store in SCRaaPS to answer revocation queries produces zero false positives while using as little as 9.25 bits per revoked fingerprint.

The blockchain-based unification of revocation lists presents an exhaustive collection of revocation information (including revoked CA certificate information) leading to a fail-closed system of revocation checking. Since blockchain technology is relatively new, previous attempts to unify revocation lists lacked the use of blockchain technology. Although cryptographic accumulators (such as One-Way RSA accumulator) provide a compact, cryptographic representation of an

authenticated set and have been studied for distributing revocation lists, updating a cryptographic accumulator requires an exchange of information between certificate owners and CAs and incur a high cost of communication overhead making it challenging to use in a large dynamic environment with a frequently updating set of issued and revoked certificates. Probabilistic set-membership data structures, on the other hand, also provide a compact representation of an arbitrary set of elements and are more suitable for distributing revocation lists as they can be independently updated with the knowledge of the set and don't require participation from certificate owners. However, they do produce false positives, which are not desirable in a fail-closed system. The application of the cascading algorithm to eliminate false positives has been shown to be effective in eliminating false positives even for larger sets (200M SHA256 fingerprints with 50% revoked) with little storage overhead ($\sim$1.2 bits per revoked fingerprint with fuse filter). Moreover, space efficiency has been demonstrated to vary little with varying proportions of revoked versus non-revoked certificate fingerprints.

We set out to investigate the following research questions:

RQ 1 Can a p2p decentralized and distributed storage platform improve the robustness and effectiveness of current revocation mechanisms?

RQ 2 Can the set membership data structure enhance the lookup performance of certificate revocation queries despite potential false positives?

OCSP servers must support heavy traffic from web servers and web browser clients to serve the revocation status of a leaf certificate and of the certificates in the CA certificate chain. OCSP servers are notorious for poor availability and response service. Therefore, it is hard to establish reliable and up-to-date collections of revocation certificates given unreliable OCSP servers. This motivated web browser developers to skip revocation checking during HTTPS handshake to reduce connection latency. The SCRaaPS design shows that the use of Scrybe's Lightweight Mining consensus algorithm strengthens its network liveness and availability guarantees and can even withstand attempts by misbehaving participants. The LWM consensus algorithm requires no additional cost (in cryptocurrency) and low computational overhead for operation, making it a

suitable choice for unifying the collection of X.509 certificate fingerprints. And the blockchain-based SCRaaPS ledger enables an exhaustive and up-to-date collection of certificate fingerprints in the system, therefore, improving the effectiveness of revocation queries.

While the blockchain technology helps establish trust among untrusted CAs and their corresponding OCSP responders, the collection of revocation lists built on the SCRaaPS system is not exhaustive unless every CA decides to participate. The cryptographically linked logs combined with additional metadata (such as timestamps and CA signatures) secure the SCRaaPS system against non-repudiation of CA activity. Therefore, a submission or a nonsubmission is identifiable through careful analysis of the blockchain ledger. But, it is impractical to assume all CAs issuing certificates to Second-Level Domain websites will participate in the SCRaaPS system, and there may still be cases where a CA fails to participate for reasons including planned or unplanned downtime of critical infrastructure, legal restrictions, or non-cooperative behavior. This limits the effectiveness of utilizing the resulting cascading fuse filter from SCRaaPS to determine the revocation status of X.509 certificates. However, it must be noted that a fully exhaustive collection of certificate revocation information cannot be achieved without further centralization of authority and infrastructure (which is antithetical to the decentralized PKI infrastructure). So, we argue that enabling a p2p decentralized infrastructure for CAs to participate does not tradeoff decentralization, but its effectiveness is limited by the percentage of total CAs that decide and continue to participate in the SCRaaPS system.

Prototyping and benchmarking results show that all three approximate set-membership data structures (including cuckoo filter, XOR filter, and fuse filter) produce zero false positives when the cascading algorithm is applied and the iterative process of building cascades successfully terminates as expected. Benchmarking results with cascading cuckoo filter show that using a cuckoo filter with 8-bits per fingerprint produces the most space-efficient cuckoo filter cascades regardless of the percentage of certificates revoked (8%-75%), and using the cuckoo filter with semi-sorted buckets does produce smaller cascades. Subsequently, we found that the cascading XOR filter is more space efficient (requiring 9.8 - 10.32 bits per revoked SHA256 fingerprint) than the cascading cuckoo filter (requiring 11.75 - 17.43 bits per revoked SHA256 fingerprint), and the

cascading fuse filter is the most space-efficient choice for the secure certificate revocation system for quick, constant time lookup of revocation information. The experimentation and benchmarking efforts in this dissertation elucidate the difference in practical results and asymptotic expectations as the cuckoo filter should, in theory, produce more space-efficient cascades (requiring $\log_2 \frac{1}{\epsilon}$ bits per fingerprint where ϵ is the false positive rate) than XOR filter (requiring $1.23 \log_2 \frac{1}{\epsilon}$ bits per fingerprint). Further, growth in the size of the cascading fuse filter and differential updates were observed by simulating periodic issuance and revocation of certificates. Results show that the cascading fuse filter increases linearly ($\sim$9.23R bits; where R is the number of revoked certificates) in total size with an increase in the total set of issued certificates, and the differential updates are generally small ($\sim$ 146 KB on average) and occasionally increase in size. Hence, the cascading fuse filter is a more space-efficient alternative to the cascading Bloom filter and other cascading approximate data structures due to its high space efficiency and high computational efficiency with a minimum requirement for optimizing parameters according to the input set.

Applying succinct computational proofs to build a verifiable cascading probabilistic data structure is an interesting area for future work. A computational proof that the SCRaaPS block proposer has computed the cascading data structure honestly and accurately would avoid validation of the cascading probabilistic data structure by SCRaaPS block validators. This computational proof must be verifiable in sub-polynomial computational time to be an effective addition.

REFERENCES

[1] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," RFC 5280, RFC Editor, May 2008. http://www.rfc-editor.org/rfc/rfc5280.txt. 1, 9, 11, 12

[2] R. Housley, W. Ford, W. Polk, and D. Solo, "RFC2459: Internet X. 509 public key infrastructure certificate and CRL profile," 1999. 1

[3] P. Hallam-Baker, "X.509v3 Transport Layer Security (TLS) Feature Extension," RFC 7633, RFC Editor, October 2015. 1, 2, 13, 14

[4] R. Duncan, "How certificate revocation (doesn't) work in practice | Netcraft News." https://news.netcraft.com/archives/2013/05/13/how-certificate-revocation-doesnt-work-in-practice.html, May 2013. [Online, Accessed: Feb 20, 2023]. 1

[5] T. Chung, "Is the web ready for ocsp must-staple?." https://blog.apnic.net/2019/01/15/is-the-web-ready-for-ocsp-must-staple/, January 2019. [Online, Accessed: 11/24/2019]. 1

[6] E. de la Hoz, G. Cochrane, J. M. Moreira-Lemus, R. Paez-Reyes, I. Marsa-Maestre, and B. Alarcos, "Detecting and defeating advanced man-in-the-middle attacks against TLS," in *2014 6th International Conference On Cyber Conflict (CyCon 2014)*, (Tallinn, Estonia), pp. 209–221, IEEE, June 2014. 1

[7] S. Medury, A. Skjellum, R. R. Brooks, and L. Yu, "SCRaaPS: X. 509 Certificate Revocation Using the Blockchain-based Scrybe Secure Provenance System," in *2018 13th International Conference on Malicious and Unwanted Software (MALWARE)*, pp. 145–152, IEEE, 2018. 1

[8] J. Larisch, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, and C. Wilson, "CRLLite: a scalable system for pushing all tls revocations to all browsers," in *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 539–556, IEEE, 2017. 2, 38, 46, 62, 64, 66

[9] P. Mutton, "Heartbleed: Revoke! The time is nigh!." "https://news.netcraft.com/archives/2014/04/15/revoke-the-time-is-nigh.html". [Online, Accessed:3/9/2018]. 3, 11, 64

[10] C. Allen, "Revocable Self-Signed TLS Certificates Hack." https://github.com/ChristopherA/revocable-self-signed-tls-certificates-hack, 2015. [Online, Accessed:2/3/2018]. 4, 18

[11] S. Y. Conner Fromknecht, Dragos Velicanu, "Decentralized Public Key Infrastructure with Identity Retention," *International Association for Cryptographic Research*, 2014. 4, 18

[12] A. Loibl and J. Naab, "Namecoin." https://www.namecoin.org/resources/whitepaper/, 2014. [Online; Accessed: Feb 20, 2023]. 4, 17

[13] R. Brooks and A. Skjellum, "Using the blockchain to secure provenance meta-data (a CCoE webinar presentation)," June 2017. Technical presentation. NCfoE seminar. 4

[14] F. Baldimtsi, J. Camenisch, M. Dubovitskaya, A. Lysyanskaya, L. Reyzin, K. Samelin, and S. Yakoubov, "Accumulators with applications to anonymity-preserving revocation," in *2017 IEEE European Symposium on Security and Privacy (EuroS P)*, pp. 301–315, April 2017. 4, 6

[15] H. Kikuchi, "Dual RSA accumulators and its application for private revocation check," in *20th International Conference on Advanced Information Networking and Applications - Volume 1 (AINA'06)*, vol. 1, pp. 6 pp.–242, April 2006. 4, 6

[16] Y. Yu, Y. Zhao, Y. Li, L. Wang, X. Du, and M. Guizani, "Blockchain-based anonymous authentication with selective revocation for smart industrial applications," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2019. 4

[17] M. Cebe and K. Akkaya, "Efficient public-key revocation management for secure smart meter communications using one-way cryptographic accumulators," in *2018 IEEE International Conference on Communications (ICC)*, pp. 1–6, May 2018. 4, 6

[18] B. Fan, D. G. Andersen, M. Kaminsky, and M. D. Mitzenmacher, "Cuckoo filter: Practically better than bloom," in *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*, pp. 75–88, ACM, 2014. 4, 24, 25, 26, 27, 37, 38, 44, 49, 50, 51, 53

[19] T. M. Graf and D. Lemire, "XOR filters: Faster and smaller than bloom and cuckoo filters," *Journal of Experimental Algorithmics (JEA)*, vol. 25, pp. 1–16, 2020. 4, 25, 51, 55, 56, 57

[20] T. M. Graf and D. Lemire, "Binary fuse filters: Faster and smaller than xor filters," *Journal of Experimental Algorithmics (JEA)*, vol. 27, no. 1, pp. 1–15, 2022. 4, 25, 57

[21] Flexera, "Digital Trends driving PKI usage | Encryption Consulting." https://www.encryptionconsulting.com/2019/11/23/digital-trends-driving-pki-usage/. 5

[22] Wikipedia contributors, "Common access card — Wikipedia, the free encyclopedia." https://en.wikipedia.org/w/index.php?title=Common_Access_Card&oldid=1135147582, 2023. [Online; accessed 21-February-2023]. 5

[23] Wikipedia contributors, "Public key infrastructure — Wikipedia, the free encyclopedia." https://en.wikipedia.org/w/index.php?title=Public_key_infrastructure&oldid=1140539321, 2023. [Online; accessed 21-February-2023]. 5

[24] D. Bernstein, "Curvecp: Usable security for the internet," *URL: http://curvecp. org*, 2011. 5

[25] B. C. Neuman and T. Ts'o, "Kerberos: An authentication service for computer networks," *IEEE Communications magazine*, vol. 32, no. 9, pp. 33–38, 1994. 5

[26] P. R. Zimmermann and P. R. Zimmermann, *The official PGP user's guide*, vol. 5. MIT press Cambridge, 1995. 5

[27] P. Todd, "Solving the pgp revocation problem with opentimestamps for git commits." https://petertodd.org/2016/opentimestamps-git-integration, Sep/26/2016. [Online, Accessed: Jan,06,2019]. 5

[28] B. Laurie, A. Langley, and E. Kasper, "Certificate transparency," *ACM Queue*, vol. 12, pp. 10–19, 2014. 5

[29] A. Zhang and X. Ma, "Decentralized Digital Certificate Revocation System Based on Blockchain," *Journal of Physics: Conference Series*, vol. 1069, p. 012125, Aug. 2018. 6

[30] E. Karaarslan and E. Adiguzel, "Blockchain based DNS and PKI solutions," *IEEE Communications Standards Magazine*, vol. 2, pp. 52–57, SEPTEMBER 2018. 6

[31] Z. Wilcox-O'Hearn, "Names: Decentralized, secure, human-meaningful: Choose two." https://web.archive.org/web/20011020191610/http://zooko.com/distnames, 2003. 8

[32] J. Brainard, A. Juels, R. L. Rivest, M. Szydlo, and M. Yung, "Fourth-factor authentication: Somebody you know," in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, (New York, NY, USA), p. 168–178, Association for Computing Machinery, 2006. 10

[33] H. Bock, "The problem with ocsp stapling and must staple and why certificate revocation is still broken." https://blog.hboeck.de/archives/886-The-Problem-with-OCSP-Stapling-and-Must-Staple-and-why-Certificate-Revocation-is-still-broken.html, 2017. [Online, Accessed:2/14/2018]. 12, 13, 30

[34] S. Santesson, M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams, "X.509 internet public key infrastructure online certificate status protocol - ocsp," RFC 6960, RFC Editor, June 2013. "http://www.rfc-editor.org/rfc/rfc6960.txt". 12, 50

[35] M. Almeshekah, "Proposal for better revocation model of SSL certificates." https://wiki.mozilla.org/images/e/e3/SSLcertRevocation.pdf, 2013. [Online, Accessed:2/21/2018]. 12, 13

[36] D. Eastlake, "Transport Layer Security (TLS) extensions: Extension definitions," RFC 6066, RFC Editor, January 2011. http://www.rfc-editor.org/rfc/rfc6066.txt. 13

[37] Y. Pettersen, "The transport layer security (TLS) multiple certificate status request extension," RFC 6961, RFC Editor, June 2013. http://www.rfc-editor.org/rfc/rfc6961.txt. 13, 14

[38] M. F. community, "CA: Improving Revocation." https://wiki.mozilla.org/CA:ImprovingRevocation. [Online, Accessed:2/15/2018]. 14

[39] M. Goodwin, "Revoking Intermediate Certificates: Introducing OneCRL." https://blog.mozilla.org/security/2015/03/03/revoking-intermediate-certificates-introducing-onecrl/, 2015. [Online, Accessed:1/29/2018]. 14

[40] ImperialViolet, "Revocation checking and chrome's crl." https://www.imperialviolet.org/2012/02/05/crlsets.html. [Online, Accessed:2/4/2018]. 14

[41] Chromium.org, "Crlsets." https://dev.chromium.org/Home/chromium-security/crlsets. [Online, Accessed:2/26/2018]. 14, 46

[42] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008. 15, 23

[43] A. Altarawneh, T. Herschberg, S. Medury, F. Kandah, and A. Skjellum, "Buterin's Scalability Trilemma viewed through a state-change-based classification for common consensus algorithms," in *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 0727–0736, IEEE, 2020. 16

[44] Blockchain.com, "Transaction rate per second." https://www.blockchain.com/charts/transactions-per-second. [Online; Accessed: Sep/07/2020]. 16

[45] A. Krishnan, "Vitalik on ethereum: "right now it can process 15 transactions per second. really, we need 100,000"." https://www.investinblockchain.com/vitalik-ethereum-needs-100k-transactions-per-second/, March, 21, 2019. [Online, Accessed: 2020-04-02]. 16

[46] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proceedings of the 26th Symposium on Operating Systems Principles*, pp. 51–68, 2017. 16

[47] EthHub, "Zk-rollups." https://docs.ethhub.io/ethereum-roadmap/layer-2-scaling/zk-rollups/, 2020. [Online, Accessed: Sep/05/2020]. 16

[48] REYNALDO, "Vitalik Buterin: Ethereum 2.0 on track for launch in july." https://www.crypto-news-flash.com/vitalik-buterin-ethereum-2-0-on-track-for-launch-in-july/, May/12/2020. [Online, Accessed: Sep/05/2020]. 16

[49] Z. Voell, "Ethereum classic hit by third 51% attack in a month." https://www.coindesk.com/ethereum-classic-blockchain-subject-to-yet-another-51-attack. [Online; Accessed: Sep/05/2020]. 17

[50] J. Rand, "34c3 chaoswest - namecoin as a decentralized alternative to certificate authorities for tls." https://www.youtube.com/watch?v=nt261DzHdNU. [Online, Accessed:3/26/2018]. 17

[51] J. Rand, "Positive tls certificate overrides for nss." https://namecoin.org/2018/02/18/positive-tls-overrides-for-nss.html. [Online, Accessed:3/26/2018]. 17

[52] D. Gilson, "What are namecoins and bit domains," 2013. [Online, Accessed:2/7/2018]. 17

[53] Technopedia, "Data lineage." https://www.techopedia.com/definition/28040/data-lineage. [Online, Accessed:2/27/2018]. 18

[54] M. Benchoufi, R. Porcher, and P. Ravaud, "Blockchain protocols in clinical trials: Transparency and traceability of consent," *F1000Research*, vol. 6, 2017. 19

[55] M. Benchoufi, R. Porcher, and P. Ravaud, "traceability of consent [version 3; referees: 1 approved, 2," 2017. 19

[56] B. Awerbuch and C. Scheideler, "Robust random number generation for peer-to-peer systems," *Theor. Comput. Sci.*, vol. 410, pp. 453–466, 2006. 23

[57] W. K. Lun, "Sets." https://math.hawaii.edu/~wongkl/Notes1.pdf, 2020. [Online; Accessed:March/2020]. 24

[58] B. R. A., "Section set sets." http://linear.ups.edu/jsmath/0290/fcla-jsmath-2.90li70.html, 2004. [Online; Accessed: March/2020]. 24

[59] F. Baldimtsi, J. Camenisch, M. Dubovitskaya, A. Lysyanskaya, L. Reyzin, K. Samelin, and S. Yakoubov, "Accumulators with applications to anonymity-preserving revocation," in *2017 IEEE European Symp. on Security & Privacy (EuroS&P)*, pp. 301–315, IEEE, 2017. 24, 26

[60] E. Tremel, "Real-world performance of cryptographic accumulators," *Undergraduate Honors Thesis, Brown University*, 2013. 24, 26

[61] L. Reyzin and S. Yakoubov, "Efficient asynchronous accumulators for distributed pki," in *International Conference on Security and Cryptography for Networks*, pp. 292–309, Springer, 2016. 24

[62] N. Fazio and A. Nicolosi, "Cryptographic accumulators: Definitions, constructions and applications," *Paper written for course at New York University: www. cs. nyu. edu/nicolosi/papers/accumulators. pdf*, 2002. 24

[63] A. Kumar, P. Lafourcade, and C. Lauradoux, "Performances of cryptographic accumulators," in *39th Annual IEEE Conference on Local Computer Networks*, pp. 366–369, IEEE, 2014. 24, 26

[64] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Comm. of the ACM*, vol. 13, no. 7, pp. 422–426, 1970. 24, 25, 37, 51

[65] Bitcoin.org, "Simplified payment verification (spv)." https://bitcoin.org/en/operating-modes-guide#simplified-payment-verification-spv, 2020. [Online; Accessed:March/2020]. 25

[66] J. Benaloh and M. De Mare, "One-way accumulators: A decentralized alternative to digital signatures," in *Workshop on the Theory and Application of of Cryptographic Techniques*, pp. 274–285, Springer, 1993. 26

[67] L. Nguyen, "Accumulators from bilinear pairings and applications," in *Cryptographers' track at the RSA conference*, pp. 275–292, Springer, 2005. 26

[68] N. Barić and B. Pfitzmann, "Collision-free accumulators and fail-stop signature schemes without trees," in *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 480–494, Springer, 1997. 26

[69] R. C. Merkle, "A certified digital signature," in *Conf. on the Theory and Application of Cryptology*, pp. 218–238, Springer, 1989. 27

[70] J. Benaloh and M. de Mare, "One-way accumulators: A decentralized alternative to digital signatures," in *Advances in Cryptology — EUROCRYPT '93* (T. Helleseth, ed.), (Berlin, Heidelberg), pp. 274–285, Springer Berlin Heidelberg, 1994. 27

[71] hostingtribunal.com, "21+ ssl statistics that show why security matters so much." https://hostingtribunal.com/blog/ssl-stats/. [Online; Accessed: Sep/05/2020]. 29

[72] B. Chazelle, J. Kilian, R. Rubinfeld, and A. Tal, "The bloomier filter: an efficient data structure for static support lookup tables," in *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 30–39, Citeseer, 2004. 37

[73] K. Salikhov, G. Sacomoto, and G. Kucherov, "Using cascading bloom filters to improve the memory usage for de brujin graphs," *Algorithms for Molecular Biology*, vol. 9, no. 1, p. 2, 2014. 37, 45

[74] M. Org, "Ca:revocation plan." https://wiki.allizom.org/CA:RevocationPlan. [Online; Accessed: Sep/07/2020]. 46

[75] W. Dai, "Crypto++ library 8.7." https://cryptopp.com/. [Online; Accessed: Oct 10, 2022]. 50

[76] cppreference.com, "std::set." https://en.cppreference.com/w/cpp/container/set. [Online; Accessed: Sep 25, 2022]. 50

[77] A. B. Yoo, M. A. Jette, and M. Grondona, "Slurm: Simple linux utility for resource management," in *Workshop on job scheduling strategies for parallel processing*, pp. 44–60, Springer, 2003. 51

[78] G. M. Kurtzer, V. Sochat, and M. W. Bauer, "Singularity: Scientific containers for mobility of compute," *PloS one*, vol. 12, no. 5, p. e0177459, 2017. 51

[79] C. of cppreference.com, "std::chrono::high_resolution_clock::now." https://en.cppreference.com/w/cpp/chrono/high_resolution_clock/now. [Online; Accessed: Oct 15, 2022]. 52, 57

[80] T. Mueler, D. Lemire, and T. Even, "fastfilter_cpp." https://github.com/FastFilter/fastfilter_cpp. [Online; Accessed: Oct 10, 2022]. 56, 57

[81] A. L. Zobrist, "A new hashing method with application for game playing," *ICGA Journal*, vol. 13, no. 2, pp. 69–73, 1990. 56

[82] W. contributors, "Perfect hash function — Wikipedia, the free encyclopedia." https://en.wikipedia.org/w/index.php?title=Perfect_hash_function&oldid=1109906535, 2022. [Online; accessed 8-October-2022]. 57

[83] M. Dietzfelbinger and S. Walzer, "Dense peelable random uniform hypergraphs," *arXiv preprint arXiv:1907.04749*, 2019. 57

[84] K. Holzhauser, "An analysis of bloom filter cascades-crlite," *ETH Scholar, Swiss Federal Institute of Technology Zurich*, 2020. 62

[85] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolić, S. W. Cocco, and J. Yellick, "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proceedings of the Thirteenth EuroSys Conference*, EuroSys '18, (New York, NY, USA), pp. 30:1–30:15, ACM, 2018. 67

[86] A. Altarawneh, "Liveness analysis, modeling, and simulation of blockchain consensus algorithms' ability to tolerate malicious miners," *UTC Scholar*, 2021. 69

[87] A. Altarawneh, F. Sun, R. R. Brooks, O. Hambolu, L. Yu, and A. Skjellum, "Availability analysis of a permissioned blockchain with a lightweight consensus protocol," *Computers & Security*, vol. 102, p. 102098, 2021. 69

VITA

Sri Naga Sai Abhijith Medury (a.k.a Sai Medury) was born in Hyderabad, Andhra Pradesh, India, to the parents of Murali Krishna and Surekha Ashtaputra. He is the first of two children, a younger brother. He attended Kendriya Vidyalaya, Nanal Nagar, Hyderabad, from third grade till tenth grade and finished high school in 2009. He then pursued junior college (11^{th} grade & 12^{th} grade) in Sri Chaitanya, S. R. Nagar, focusing on STEM subjects: Maths, Physics, and Chemistry. After finishing junior college in 2011 with distinction grades, he joined the Gandhi Institute of Technology and Management (GITAM) to pursue undergraduate education majoring in Computer Science. Introduced to computer science and electrical engineering concepts, he developed an interest in programming and cybersecurity and decided to pursue higher education to gain a deeper understanding of these technical concepts. He joined Auburn University in 2015 to pursue the Master in Software Engineering degree and gain expertise in software engineering design, optimization, and operation processes. Sai graduated in May 2017 with an M.Sw. degree and joined the Ph.D. in Computational Science program at the University of Tennessee at Chattanooga in Aug 2017. After multiple publications, internships, and successful completion of projects, he graduated with Ph.D. in May 2023 and launched his career in System Engineering at Vanderbilt University, Nashville, TN. Sai is interested in research topics: zero-knowledge cryptography, BFT consensus design, probabilistic data structures, and digital identity.