A NEW BLOCKCHAIN-BASED PKI AND A DIGITAL SIGNATURE FORMAT FOR LONG-TERM VALIDATION OF DIGITAL SIGNATURES

ELEKTRONİK İMZALARIN UZUN VADELİ DOĞRULANMASI İÇİN YENİ BLOKZİNCİRİ TABANLI PKI VE DİJİTAL İMZA FORMATI

ERHAN TURAN

PROF. DR. SEVIL ŞEN

Supervisor

DR. TAMER ERGUN

2nd Supervisor

Submitted to

Graduate School of Science and Engineering of Hacettepe University

as a Partial Fulfillment to the Requirements

for the Award of the Degree of Doctorate of Philosophy

in Computer Engineering

Jun 2024

ABSTRACT

A NEW BLOCKCHAIN-BASED PKI AND A DIGITAL SIGNATURE FORMAT FOR LONG-TERM VALIDATION OF DIGITAL SIGNATURES

Erhan TURAN

Doctorate of Philosophy, Computer Engineering

Supervisor: Prof. Dr. Sevil ŞEN

2nd Supervisor: Dr. Tamer ERGUN

Jun 2024, 175 pages

Traditional Public Key Infrastructure (PKI) has long been grappling with security issues arising from its centralized and non-transparent design. Despite Google's implementation of the Certificate Transparency (CT) project in 2013, aimed at addressing SSL certificate tracking, it has continued to be vulnerable to attacks such as split-world attacks. In recent years, blockchain-based PKI architectures have emerged as promising solutions to overcome such issues. However, existing research has predominantly focused on SSL certificates, overlooking other critical certificate types such as electronic signatures/seals, code signing, and S/MIME — all of which are dependent on the foundational PKI infrastructure. In this study, we present a semi-decentralized new blockchain-based PKI (SemiDec-PKI) architecture designed to accommodate various certificate types.

Our SemiDec-PKI, combining the principles of the Web of Trust with a centralized model, establishes a resilient, distributed infrastructure. This unparalleled fusion minimizes reliance on a single central authority, reducing security vulnerabilities associated with the

single points of failure in traditional PKI systems. Through the collective consensus of multiple trusted organizations, SemiDec-PKI achieves higher fault tolerance, protecting the system against potential issues such as certificate misissuance and compromise of certificate authority. Additionally, within SemiDec-PKI, we introduce a stake-based reward-punishment mechanism that incentivizes honest behavior and penalizes malicious actions, serving as a robust deterrent against impersonation attacks.

In addition to the pioneering SemiDec-PKI, this study introduces a new blockchain-compatible electronic signature format known as Blockchain-based Advanced Electronic Signature (BlAdES). This format not only adheres to blockchain principles but also provides interoperable services through smart contracts. It introduces an optimized form of multiple signatures — a BLAdES form using threshold signatures and Musig. This innovative approach ensures secure, collaborative, and scalable electronic transactions, aligning with the dynamic needs of the digital era. The study's multifaceted contributions extend beyond SSL certificates, encompassing various critical certificate types and establishing a resilient and secure foundation for diverse applications in the evolving landscape of digital transactions.

Keywords: PKI, Smart Contract, Blockchain, Electronic Signature, Certificate Transparency

ÖZET

ELEKTRONİK İMZALARIN UZUN VADELİ DOĞRULANMASI İÇİN YENİ BLOKZİNCİRİ TABANLI PKI VE DİJİTAL İMZA FORMATI

Erhan TURAN

Doktora, Bilgisayar Mühendisliği

Danışman: Prof. Dr. Sevil ŞEN

Eş Danışman: Dr. Tamer ERGUN

Ocak 2024, 175 sayfa

Geleneksel Genel Anahtar Altyapısı (PKI), merkezi ve şeffaf olmayan tasarımından kaynaklanan güvenlik sorunlarıyla uzun süredir mücadele etmektedir. Google'ın 2013 yılında SSL sertifikası takibine yönelik olarak Sertifika Şeffaflığı (CT) projesini uygulamaya koymasına rağmen, bu altyapı hala split-world saldırıları gibi saldırılara karşı savunmasız kalmıştır. Son yıllarda, blockchain tabanlı PKI mimarileri bu tür sorunları aşmak için umut verici çözümler olarak ortaya çıkmıştır. Ancak mevcut araştırmalar ağırlıklı olarak SSL sertifikalarına odaklanmış olup, elektronik imzalar/mühürler, kod imzalama ve S/MIME gibi diğer kritik sertifika türlerini göz ardı etmiştir — bunların tümü temel PKI altyapısına bağımlıdır. Bu çalışmada, çeşitli sertifika türlerini kapsayacak şekilde tasarlanmış yarı merkeziyetsiz yeni bir blockchain tabanlı PKI (SemiDec-PKI) mimarisini sunuyoruz.

Web of Trust (Güven Ağı) ilkeleri ile merkezi bir modeli birleştiren SemiDec-PKI, dayanıklı ve dağıtılmış bir altyapı oluşturur. Bu benzersiz birleşim, tek bir merkezi otoriteye olan bağımlılığı en aza indirerek, geleneksel PKI sistemlerindeki tek hata noktalarına bağlı güvenlik açıklarını azaltır. Birden fazla güvenilir kuruluşun ortak kararıyla, SemiDec-PKI

daha yüksek hata toleransı sağlar ve sertifika hatalı düzenlenmesi ve sertifika otoritesinin tehlikeye girmesi gibi potansiyel sorunlara karşı sistemi korur. Ayrıca, SemiDec-PKI içinde dürüst davranışı teşvik eden ve kötü niyetli eylemleri cezalandıran stake tabanlı bir ödül-ceza mekanizması tanıtarak sahtecilik saldırılarına karşı sağlam bir caydırıcı sağlar.

Öncü SemiDec-PKI'nın yanı sıra, bu çalışma Blockchain tabanlı Gelişmiş Elektronik İmza (BlAdES) olarak bilinen yeni bir blockchain uyumlu elektronik imza formatını tanıtmaktadır. Bu format, blockchain ilkelerine uymakla kalmayıp, aynı zamanda akıllı sözleşmeler aracılığıyla birlikte çalışabilir hizmetler sunar. Çoklu imzaların optimize edilmiş bir formunu — threshold ve Musig imzaları kullanan bir BlAdES formunu — tanıtarak güvenli, işbirlikçi ve ölçeklenebilir elektronik işlemleri garanti eder, dijital çağın dinamik ihtiyaçlarıyla uyum sağlar. Çalışmanın çok yönlü katkıları SSL sertifikalarının ötesine geçerek, çeşitli kritik sertifika türlerini kapsar ve dijital işlemlerin gelişen ortamında çeşitli uygulamalar için dayanıklı ve güvenli bir temel oluşturur.

Keywords: PKI, Smart Contract, Blockchain, Electronic Signature, Certificate Transparency

ACKNOWLEDGEMENTS

This dissertation is dedicated to my beloved wife Özge, my wonderful daughter Ece Arya, and my precious son Kaan. I extend my gratitude to my mother, father, and brother. I am also grateful to Dr. Tamer ERGUN for his support throughout my professional journey. I owe a special thanks to the professors at Hacettepe University for their invaluable teachings during my doctoral studies, the esteemed members of the Thesis Monitoring Committee, the Computer Engineering Department at Hacettepe University for offering me a doctoral opportunity, and HAVELSAN. Furthermore, I would like to give a special mention to my advisor and mentor, Prof. Dr. Sevil ŞEN, who has been a guiding light throughout my thesis work and serves as a great inspiration to me.

CONTENTS

Page

TABLES

FIGURES

ABBREVIATIONS

ASN	:	**A**bstract **S**yntax **N**otation
BlAdES	:	**B**lockchain-based **A**dvanced **E**lectronic **S**ignature
CA	:	**C**ertificate **A**uthorithy
CAdES	:	**C**MS **A**dvanced **E**lectronic **S**ignature
CIRT	:	**C**ertificate **I**ssuance and **R**evocation **T**ransparency
CLM	:	**C**ertificate **L**og and **M**ainteiner
CMS	:	**C**ryptographic **M**essage **S**yntax
CRL	:	**C**ertificate **R**evocation **L**ist
CT	:	**C**ertificate **T**ransparency
DTKI	:	**D**istributed **T**ransparent **K**ey **I**nfrastructure
EE	:	**E**nd **E**ntity
ECC	:	**E**lliptic **C**urve **C**ryptography
ERC20	:	**E**thereum **R**equest for **C**omments
ETSI	:	**E**thereum **T**elecommunications **S**tandards **I**nstitute
ITU	:	**I**nternational **T**elecommunication **U**nion
IV	:	**I**nitialization **V**ector
JAdES	:	**J**SON **A**dvanced **E**lectronic **S**ignature
JSON	:	**J**ava **S**cript **O**bject **N**otation
OCSP	:	**O**nline **C**ertificate **S**tatues **P**rotocol
PAdES	:	**P**df **A**dvanced **E**lectronic **S**ignature
PKI	:	**P**ublic **K**ey **I**nfrastructure
QES	:	**Q**ualified **E**lectronic **S**ignature
PoS	:	**P**roof **o**f **S**take
PoW	:	**P**roof **o**f **W**ork
RA	:	**R**egistration **A**uthority
RFC	:	**R**equest **F**or **C**omments

SB	:	**S**upervisor **B**ody
SC	:	**S**mart **C**ontract
SHA	:	**S**ecure **H**ash **A**gorithm
SPoF	:	**S**ingle **P**oint **o**f **F**ailure
SSL	:	**S**ecure **S**ocket **L**ayer
TS	:	**T**ime **S**tamp
TSP	:	**T**rust **S**ervice **P**rovider
TSS	:	**T**hreshold **S**ignature **S**cheme
XML	:	e**X**tensible **M**arkup **L**anguage
XAdES	:	**X**ML **A**dvanced **E**lectronic **S**ignature

1. INTRODUCTION

In today's interconnected and digitalized world, ensuring secure communication and protecting sensitive information has become a critical priority. Public Key Infrastructure (PKI) stands as a cornerstone in the realm of modern cryptographic systems, providing a robust framework for secure communication and digital transactions. At its essence, PKI is a comprehensive set of policies, technologies, and standards that facilitate the secure exchange of information in the digital domain. The fundamental principle of PKI revolves around the use of asymmetric cryptography, wherein pairs of public and private keys are employed to encrypt and decrypt sensitive data. This infrastructure not only ensures the confidentiality and integrity of digital communications but also verifies the authenticity of parties involved. As cyber threats and the demand for secure digital transactions continue to escalate, the importance of PKI cannot be overstated. Its role extends beyond mere encryption; it encompasses the issuance and management of digital certificates, establishing a trust framework vital for online identities and transactions [19–22].

Certification Authorities (CAs) emerge as key players within the PKI ecosystem. These entities, often trusted third parties, take center stage in validating the authenticity of entities involved in online transactions. By issuing digital certificates, CAs confirm the legitimacy of public keys, thereby enhancing the overall security of digital communications. In essence, CAs form an integral part of the PKI framework, reinforcing the trustworthiness of online interactions and ensuring the safeguarding of sensitive data. Understanding the seamless connection between PKI and the role of Certification Authorities is crucial for navigating the intricate landscape of digital security and establishing a secure foundation for online transactions [23–32].

While CAs play a vital role in the system, they expose PKI to potential vulnerabilities and incidents where faulty certificates might be issued. A number of high-profile incidents have highlighted the risks associated with faulty certificate issuance. One notable example is the 2011 breach of DigiNotar, where attackers managed to issue fraudulent certificates for

popular websites including Google, Yahoo, and Skype [33]. This incident demonstrated the CA's weak security controls and the potential impact of compromised certificates on their users' data security and trust. As another example, Symantec mistakenly issued certificates that included a domain name in violation of CA industry policies and procedures in 2015 [34]. A rigorous vetting process as well as adherence to industry standards by CAs are crucial to preventing such unauthorized certificate issuances.

As shown in the incidents above, CAs constitute a *Single Point of Failure (SPoF)* in the certificate issuance mechanism. Therefore, CAs should provide guarantee regarding the accuracy of the information included in the certificate or the integrity of their infrastructure. CAs offers financial compensation or remedies in situations where the CA's negligence or failure to meet industry standards has caused harm to a subject/entity. As part of the current PKI model, *punishment issues* are handled by insurance companies, and entities must prove that they have been victimized in court. Another issue is the *management of trust lists*. In PKI, a certificate chain is a list of certificates that usually starts with an end-entity certificate followed by one or more CA certificates. CAs at the top of the hierarchy are called root CAs. They can issue certificates directly for users, or delegate some of the task to intermediate CAs. Root CAs are self-signed and pre-installed in the Operating Systems (OS), browsers, and applications. Besides the trust lists of SSL certificates, applications could have their own trust lists of CAs. For example, Adobe has a trust list for electronic signature [35], Java has a trust list for code signing certificates. Since CA needs to contact every application, browser, and OS, so it is hard to manage trust lists for each software.

Moreover, CAs have a crucial role in tasks like identity validation and application assessment. However, their involvement also introduces the risk of human errors. To enhance security and mitigate potential risks, it becomes essential to implement a crosscheck mechanism. This process should involve personnel from diverse authorities to maximize its effectiveness. Currently, the conventional certificate lifecycle lacks this protective measure. Hence, impersonation attacks that impersonate legitimate entities and fraudulently obtain certificates could occur in this registration step. Therefore, trust and security of the infrastructure could be compromised due to these *certificate registration issues*. The last

issue called *the revocation monopoly issues* is related to certificate revocation, which is the process of declaring a previously issued digital certificate as invalid or no longer trustworthy before its expiration date. Certificate owners or CAs can initiate the revocation process for the certificate to the responsible CA. In an emergency, it can be problematic if the CA cannot be reached immediately, such as during non-working hours.

Transfer Layer Security (TLS) [36] is one of the most popular protocols, which relies upon absolute trust among Certificate Authorities (CAs) [37–40]. However, CAs are susceptible to certificate misissuance, leading to attacks such as man-in-the-middle, Certificate Transparency (CT) [41] is proposed as a solution to such problems by monitoring and auditing SSL certificates. However, Certificate Transparency is prone to split-world attacks [42]. In order to facilitate the certificate lifecycle management of CAs and to address known attacks, new PKI architectures based on blockchain have been proposed in the literature [1–7]. However these studies adopted a blockchain-based approach specifically for SSL but other type of certificates are also important.

Although not used as much as SSL certificates, in many countries, especially in the European Union, a qualified electronic signature has the same validity as a wet signature. For example in Turkey, the landscape of secure digital transactions is overseen by seven authorized electronic certificate and as of the end of June 2023, these electronic certificate service providers have collectively issued a substantial total of 7.8 million electronic certificates. Within this figure, 6.9 million are designated as electronic signature certificate, emphasizing their crucial role in validating the authenticity of digital communications. Furthermore, 903 thousand mobile signature [43–45] certificates contribute to the multifaceted use of these certificates in the mobile ecosystem. The prevalence of such qualified electronic certificates underscores their importance in enhancing the security posture of digital transactions, establishing a framework that goes beyond encryption to ensure the trustworthiness and integrity of online identities and interactions. Importantly, the role of qualified electronic certificates is comparable in significance to the widely recognized SSL certificates, further emphasizing their importance in securing digital transactions and communications [46].

For this reason, the person with the Qualified Electronic Certificate (QEC) can also have the wet signature power of the person. As we mentioned before, although CAs have passed many audits, incorrect issuance situations in SSL can also be experienced in Qualified Electronic Certificates. For this reason, it is very important that the certificates issued on their behalf can be followed by individuals and that individuals can manage the certificates. For this reason, it is very important that electronic signatures are created/verified correctly and that electronic signatures can be tracked. Certificates can be generated, and electronic signatures can be signed on users' behalf without the users' knowledge. While creating an electronic signature, transactions such as revocation query and withdrawal of the CA certificate are made to the Trust Service Provider (TSP), but these operations can also be done offline. Signers can create signatures in much different software by simply entering their pins with a USB smart card, and anyone who has physical access to this smart card and knows the pin can also create these signatures. In this case, the signature transactions made with a user's certificate cannot be followed. With the principle of non-repudiation, all responsibility has been given to the signatory, and the signatory will inevitably be responsible for legal provisions.

From this perspective, the current study aims to expand upon the scope of PKI-based trust services managed by the blockchain technology.

Blockchain is a distributed, decentralized, and immutable ledger composed of transaction records, and serves as a foundation for various protocols such as Bitcoin [47], Ethereum [48]. All of these have a common aim, to ease the burden of adding transparency to real-life applications such as in financial services, asset tracking, and many other instances. A game-changer approach known as smart contracts (SCs) was introduced with Ethereum, which are self-executing programs stored on a blockchain that are executed when certain declared conditions are met. SCs help to automate the execution of agreements between parties so that both parties can trust the outcome without need for any third-party involvement or time loss. They also make it possible to automate a workflow by triggering a sequence of actions.

This study addresses several critical problems in the traditional Public Key Infrastructure

(PKI) system and proposes a novel blockchain-based approach called SemiDec-PKI (Semi Decentralized-PKI). The issues identified include challenges in the management of trust lists, the presence of *a Single Point of Failure (SPoF)*, concerns related to Certificate Authorities (CAs) and their potential mistakes, deficiencies in the certificate lifecycle such as registration and revocation, consequences and punishment mechanisms for fraudulent CAs, and limitations in monitoring and logging. The proposed SemiDec-PKI aims to overcome these challenges by combining centralized and Web-of-trust paradigms, utilizing blockchain technology, smart contracts, and a voting mechanism.

Our approach actively involves multiple CAs in the certificate issuance process, introduces a *punishment mechanism* for security breaches, and ensures seamless validation, issuance, and revocation of certificates. The system also incorporates a hierarchical structure for auditing and cross-checking, providing a more secure and efficient framework for managing trust lists and addressing the shortcomings of traditional PKI operations. Overall, SemiDec-PKI represents a promising advancement in the field of Public Key Infrastructures, contributing to enhanced security, resilience, and integrity.

In addition to the identified challenges in the traditional Public Key Infrastructure (PKI) system, another noteworthy issue is the management of trusted lists. The European Union (EU) has enacted regulations for the management of trust lists, and industry standards are set by the CA/Browser Forum (CA/B Forum) [49] to guide the issuance and management of digital certificates. However, the lack of a common global trust list for Certificate Authorities (CAs) issuing certificates makes it difficult to determine the validity of trust lists from an application's perspective. This further exacerbates the complexity of *managing trust lists*, as different browsers or operating system vendors may run certificate inclusion programs independently, contributing to a fragmented landscape.

To address these challenges comprehensively, the thesis study introduces the SemiDec-PKI solution. SemiDec-PKI combines centralized and Web-of-trust paradigms, leveraging blockchain technology and smart contracts to establish a robust and transparent PKI system. Specifically, SemiDec-PKI addresses the management of trust lists by employing a

collaborative approach involving Supervisor Bodies (SBs) and CAs. The system introduces a voting process where existing Supervisor Bodies can vote for new SBs, allowing for the inclusion of participants from new countries. This not only enhances the inclusivity of the trust list management process but also fosters a globally harmonized approach to maintaining trusted CAs.

By incorporating a hierarchical structure, SemiDec-PKI facilitates auditing and cross-checking of the entire PKI ecosystem, contributing to the effective management of trust lists. The use of smart contracts ensures a secure and decentralized decision-making process, reducing the reliance on a Single Point of Failure (SPoF). Furthermore, SemiDec-PKI's punishment mechanism acts as a deterrent against malicious behavior, reinforcing the integrity and security of the trust list management process.

The electronic signature is another popular use of certificates. Within the scope of eIDAS, advanced signature formats such as CAdES [50, 51], XAdES [52], PAdES [53] and JAdES [54] have been introduced and used for many years in order to establish interoperability and to fix security issues. For each format, the methods for creation [55], preservation [56] and validation [55] [57], [58] of signatures are clearly defined; however, interoperability among applications is still a major problem due to their different implementations as shown in ETSI plug tests [59]. Therefore, in addition to SemiDec-PKI, the current study also proposes a new format known as Blockchain-based Advanced Electronic Signature (BlAdES) which contributes to interoperability by managing signature infrastructure via SCs and evading interaction of various applications and integrate it to SemiDec-PKI.

In a conceivable attack scenario, an adversary has the potential to acquire a smart card and, upon discovering its password [60–62], clandestinely sign a document, evading detection. This covert action poses a significant threat as it remains untraceable, with possible severe consequences for the victim [63]. Recognizing these gaps, the integration of Blockchain-based Advanced Electronic Signature (BlAdES) to SemiDec-PKI becomes paramount. BlAdES leverages blockchain technology to introduce transparency to the

signing process for digital signatures. By recording each Qualified Electronic Signature on a decentralized and tamper-resistant ledger, BlAdES ensures traceability and accountability, significantly enhancing the overall security and trustworthiness of digitally signed content.

BlAdES represents a pioneering step towards establishing transparency and traceability in the signing process, specifically for Qualified Electronic Signatures. This comprehensive approach strengthens the security posture of digital transactions, offering a proactive solution for ensuring the authenticity and integrity of electronically signed documents in the digital age.

In real-world scenarios, digital signature on document management systems [64, 65] play a pivotal role where signers consecutively sign the same document, a process commonly referred to as multiple signatures [66]. This practice is prevalent in various industries where collaboration and sequential approval are essential. In our study, we build upon our previous work with BlAdES (Blockchain-based Ledger for Advanced Digital Signatures) by incorporating threshold signatures and MuSig [67]. This extension allows for an optimized signing flow specifically tailored for multiple signatures. Our approach not only enhances the efficiency of the signature process but also introduces an additional layer of security through the utilization of threshold signatures, contributing to a more robust and streamlined digital signature workflow in document management systems.

1.1. Scope Of The Dissertation

This dissertation delves into the inherent security challenges confronted by the conventional Public Key Infrastructure (PKI) due to its centralized and non-transparent design. Despite Google's commendable effort with the Certificate Transparency (CT) project, which aimed to mitigate SSL certificate tracking, vulnerabilities such as split-world attacks persist. Recent years have witnessed the emergence of blockchain-based PKI architectures as promising alternatives. However, current research predominantly fixates on SSL certificates, neglecting crucial certificate types like electronic signatures/seals, code signing, and S/MIME, all

intricately linked to the foundational PKI infrastructure. This study introduces a novel blockchain-based PKI architecture capable of accommodating diverse certificate types.

Our innovative SemiDec-PKI combines the Web of Trust principles with a centralized model, establishing a resilient, distributed infrastructure. This fusion minimizes dependence on a singular central authority, thereby mitigating security risks associated with single points of failure in traditional PKI systems. Through collective consensus from multiple trusted organizations, SemiDec-PKI attains higher fault tolerance, safeguarding against potential issues like certificate misissuance and compromise of certificate authority. Additionally, a stake-based reward-punishment mechanism is introduced within SemiDec-PKI, incentivizing integrity and penalizing malicious actions as a robust deterrent against impersonation attacks.

Beyond SemiDec-PKI, this study unveils Blockchain-based Advanced Electronic Signature (BlAdES), a novel blockchain-compatible electronic signature format. BlAdES not only aligns with blockchain principles but also facilitates interoperable services through smart contracts. The thesis introduces an optimized form of multiple signatures — a BLAdES form utilizing threshold signatures. This groundbreaking approach ensures secure, collaborative, and scalable electronic transactions, catering to the dynamic requirements of the digital era. The study's contributions extend beyond SSL certificates, encompassing critical certificate types, providing a resilient foundation for diverse applications in the evolving landscape of digital transactions.

1.2. Contributions

In this dissertation, we address those gaps by means of introducing a brand new, honest, and effective approach. the key contributions of this research are as follows:

- It introduces a semi-decentralized extensive PKI based on blockchain that supports any type of certificate and replaces the traditional X.509 certificate through its redefinition

within SCs, thus reducing the complexity of storing, parsing, and decoding X.509 certificates.

- It detects misissuance and eliminates split-world-attacks; prevents TSPs and supervisory bodies from being compromised by using a reward-punishment mechanism that uses Ethereum Request for Comments (ERC20) tokens [68].
- SemiDec-PKI could be adapted to new services for different types of certificates, such as qualified validation/preservation services for electronic signature/seal.
- SemiDec-PKI eliminates distinct timestamping processes with a slight delay caused by Ethereum block delays, hence timestamping a signature is no longer an issue.
- The signature submission process allows the signer and the submitter to be different entities, and the signer's Ethereum addresses can remain anonymous.
- BlAdES provides signature creation operations based on conventional private keys or transaction keys by utilizing SCs and can preserve signatures for long-term validation with or without content, by not forcing signers to append documents to blockchain for the purposes of ensuring confidentiality.
- BlAdES includes signature validation via a reward-punishment mechanism using ERC-20 tokens [68] in order to eliminate issues related to interoperability or single point of failure.
- The implementation of Semidec-PKI and BlAdES are provided[1].
- It is provided an BC-ECTSS and MuSig integration of BlAdES for multiple signatures.

1.3. Organization

The organization of the thesis is as follows:

[1] `https://github.com/erhaanturan/thesis`

- Chapter 1, we lay the foundation for our thesis by presenting the motivation that propels our research, outlining the specific contributions we make to the field, and defining the scope of our work. By providing a comprehensive overview, this chapter serves as the gateway to understanding the significance of the subsequent chapters.

- In Chapter 2, we delve into the historical and theoretical underpinnings relevant to our research. This background overview establishes the necessary context for the reader, elucidating key concepts, methodologies, and technologies that form the backdrop for our contributions. By synthesizing existing knowledge, this chapter creates a solid framework upon which our innovative work is built.

- Building on the foundation laid in Chapter 2, Chapter 3 conducts an in-depth exploration of related work within the domain of our research. We critically analyze existing literature, identifying key advancements, methodologies, and challenges encountered by researchers in similar fields. This chapter aims to position our work within the broader academic landscape while highlighting the gaps and opportunities that our contributions address.

- Chapter 4 introduces Semidec-PKI, a novel framework or methodology that represents a significant contribution to the field. This chapter elucidates the intricacies of Semidec-PKI, detailing its architecture, functionalities, and the specific problems it addresses within the realm of public key infrastructure. Through a thorough exploration, we showcase how Semidec-PKI stands as a noteworthy advancement, laying the groundwork for subsequent chapters.

- In Chapter 5, we introduce BlAdES, a groundbreaking electronic signature format based on blockchain technology. This chapter outlines the motivations behind BlAdES, its unique features, and the advantages it brings to the field of long-term electronic signatures. Through practical demonstrations and theoretical foundations, we illustrate the transformative potential of BlAdES in enhancing the security and durability of electronic signatures.

- Chapter 6 presents our final contribution, a new optimized method that combines threshold signatures and MuSig [67] with the innovative BlAdES format. This chapter outlines the intricacies of this hybrid approach, demonstrating how it addresses specific challenges in the realm of multiple signatures. Additionally, we provide a comprehensive summary of the entire thesis, emphasizing its collective contributions and impact on the field. Furthermore, we discuss potential future directions, paving the way for continued advancements in the domains explored throughout this thesis.
- Chapter 7 summarizes the key findings and contributions of the thesis and reflects on the implications of the research and its potential impact on the field.

2. BACKGROUND OVERVIEW

This section encompasses crucial components of digital security infrastructure, spanning from traditional Public Key Infrastructure (PKI) to cutting-edge blockchain technologies. In the realm of PKI, the exploration begins with the examination of Hash Functions (Section 2.1.1.), which plays a pivotal role in creating fixed-size representations of data essential for cryptographic operations. Moving forward, Certificate Authority (CA) (Section 2.1.2.) is introduced, serving as a cornerstone in validating the authenticity of digital entities through the issuance of digital certificates (Section 2.1.3.) . This section delves into the mechanisms of Certificate Revocation Lists (CRL) (Section 2.1.4.) and the Online Certificate Status Protocol (OCSP) (Section 2.1.5.) , crucial elements for maintaining the life cycle of digital certific ates. Time Stamp Section 2.1.6. emerges as a critical tool, providing temporal validation, while Digital Signature: An Overview (Section 2.1.7.) encapsulates the broader context of digital signatures within PKI.

Transitioning into contemporary advancements, the section extends to Blockchain in Section 2.2. and its disruptive influence. Within this domain, Consensus Algorithms in Section 2.2.1. become pivotal in establishing agreement protocols, ensuring the integrity and trustworthiness of distributed ledgers. Smart Contracts in Section 2.2.2. represent self-executing contracts with encoded business logic, pushing the boundaries of automation and autonomy. A comparative analysis between Databases and Blockchain Section 2.2.3. unfolds, shedding light on the distinctive features that set these two technologies apart. The concluding integration of Blockchain and Public Key Infrastructure (PKI): An Integrative Approach (Section 2.2.4.) underscores the evolving landscape, offering a holistic perspective that bridges traditional cryptographic practices with emerging decentralized technologies. This comprehensive background overview sets the stage for a nuanced exploration of the intricate interplay between security, cryptography, and decentralized systems.

2.1. Public Key Infrastructure

In PKI, digital certificates are essential for linking a public key to the entity (such as an individual or organization) that owns it as defined in [69][70]. These certificates are verified using digital signatures from Certificate Authorities (CAs). A digital certificate contains essential components, including the public key itself, personal information about the entity, and additional data required for validating its authenticity. CAs are responsible for signing end-user certificates. In many cases, the CA does not directly issue certificates to end-entities but uses one or more intermediate CAs. At the top of the chain of trust is the trusted root certificate. The trust list is a pre-configured set of certificates that are inherently trusted by a software application, web browser, operating system, or any other system that requires PKI functionality. They act as the starting point of trust in the PKI hierarchy, as they are self-signed certificates which are called as root certificates, representing the highest level of trust in the system. There are various types of certificates used for different purposes in (PKI) such as SSL certificates for securing web communications, electronic seal certificates for document authenticity, code-signing certificates for verifying software authenticity and electronic signature certificates. Moreover, there are emerging application domains for certification such as Multi-Dimensional Certification [71] and Certification of Internet of Things Devices [72]. Each type of certificate may have its own trust list to establish trust in the corresponding CAs. Because of the diverse range of certificates and the need to maintain their validity and trustworthiness over time, managing trust lists can be challenging.

Public Key Infrastructure (PKI) plays a crucial role in information security by integrating cryptographic techniques, key management processes, and adherence to policies to ensure secure electronic communications. PKI operates through a sophisticated yet systematic framework that facilitates the distribution and authentication of public encryption keys, thereby enabling secure data transfers and fostering trust in the digital domain. This section explores the core mechanisms, applications, benefits, and challenges of PKI.

PKI encompasses the frameworks and services that enable secure electronic transactions and communications by utilizing various cryptographic services, such as confidentiality, data

integrity, and identity authentication. It relies on asymmetric cryptography, or public-key cryptography, where each user has a pair of keys: a public key and a private key. The public key is widely distributed for data encryption, while the private key is kept confidential and used for data decryption.

Key components of PKI include a Certificate Authority (CA), Registration Authority (RA), central directory, certificate management system, and certificate policies. The Certificate Authority (CA) forms the foundation of the Public Key Infrastructure (PKI) system, overseeing the issuance and revocation of certificates. The Registration Authority (RA) is tasked with validating user requests for digital certificates. A central directory ensures the secure storage and distribution of public keys, and the certificate management system oversees administrative tasks. Additionally, certificate policies outline the procedures for the CA's management and use of certificates.

PKI is integral to securing various aspects of online data communication and transactions. It is widely employed in e-commerce, online banking, email encryption, secure access to private networks, and digital signatures, among other applications. By establishing secure communication channels between parties over insecure networks, PKI enhances the safety and reliability of online interactions.

Digital signatures, one of the most prevalent applications of PKI, validate the authenticity and integrity of messages, software, or digital documents. They serve as a digital equivalent to handwritten signatures but with much higher security, confirming both the signer's identity and the integrity of the data.

PKI offers several significant benefits, notably enhancing the security of online transactions and communications. Through robust authentication, PKI facilitates the secure exchange of information over insecure networks, thereby boosting user trust and supporting the growth of online services.

Another major advantage of PKI is data integrity. By using cryptographic checksums, PKI ensures that the data received by the recipient is identical to what was sent by the sender,

thus confirming the data's integrity.

Additionally, PKI provides non-repudiation, ensuring that parties involved in a communication cannot deny their participation. This feature is particularly valuable in legal contexts and business transactions.

Despite its numerous advantages, implementing a PKI comes with several challenges. The foremost challenge is the significant investment required in both financial resources and time. Properly establishing a PKI involves an extensive process that includes comprehensive planning, deployment, and management. This process demands the expertise of professionals skilled in cryptography and network security.

Another critical challenge is the careful distribution and management of keys. Private keys must be securely stored and kept inaccessible to unauthorized parties. The loss or theft of a private key can result in severe security breaches.

Trust is also a significant issue in PKI implementation. Users must place their trust in the Certificate Authority (CA), and this trust needs to be meticulously maintained. Any mismanagement or breach within the CA can lead to catastrophic consequences, undermining the entire system's reliability.

In an era where cyber-attacks and data breaches are persistent threats, PKI stands as a vital defense mechanism. By enabling secure communications, maintaining data integrity, and ensuring non-repudiation, PKI bolsters trust in digital interactions. However, the successful implementation and management of PKI require a deep understanding of its components, operations, and potential challenges. Organizations must continuously strive to improve and adapt their PKI systems in response to the evolving cyber threat landscape, thereby safeguarding their operations and ensuring their longevity in the digital realm.

2.1.1. Hash Functions

In the dynamic landscape of modern cryptography, the core principle of hashing is to obfuscate raw information, rendering it irreproducible in its original form. This intricate process involves subjecting data to a mathematical operation through a specialized function known as a hash function, yielding a unique hash value or digest. The irreversible nature of hash functions ensures that the original plaintext cannot be deduced from the digest, even through exhaustive means [73–77].

Hash functions are crucial in converting plaintext into its corresponding hash digest. Designed to be irreversible, they are fundamental in securing digital information. Additionally, hash functions exhibit consistency by always producing the same output for a given input, regardless of how many times the function is applied.

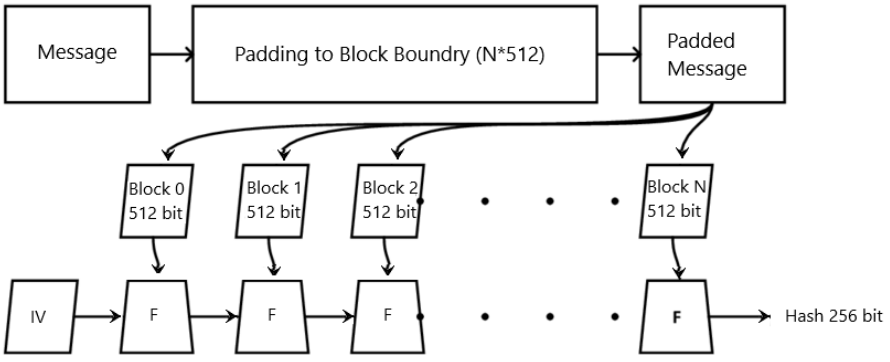

Figure 2.1 SHA-256 Basic Flow

One prominent member of the SHA-2 family, SHA-256, stands out as a stalwart in cryptographic algorithms. Conceived in 2001 through collaboration between the NSA and NIST, SHA-256 aimed to supplant its predecessor, SHA-1, which had become susceptible to brute force attacks [78]. The “256” in its name signifies the fixed size of the hash digest, ensuring uniformity at 256 bits regardless of the original plaintext/cleartext size. SHA 256 flow is simply illustrated in Figure 2.1.

1. The message is padded in multiples of 512 bits. [78]
2. The message is divided into blocks of 512 bits. [78]

3. Initialization vector and first block enter the function. [78]

4. Output of the first function and Block 1 join the second function as input, continuing until the end. [78]

5. This process is repeated 64 times within SHA-256. [78]

Another notable hash algorithm in the cryptographic arena is Keccak-256. Keccak [79], a family of sponge construction hash functions, gained recognition for its role as the cryptographic basis for SHA-3. Keccak-256, specifically, produces a 256-bit hash digest and is notable for its resistance to various cryptographic attacks.

2.1.2. Certificate Authority (CA)

A Certificate Authority (CA) serves as a critical element of the Public Key Infrastructure (PKI). It is a trusted third-party entity responsible for issuing, renewing, and revoking digital certificates. It provides assurance of the authenticity of certificates and therefore plays a crucial role in maintaining trust within the digital ecosystem. By signing digital certificates, the CA essentially vouches for the identity of the certificate holder.

Throughout the thesis, we will use some terms derived from eIDAS [80] which is a framework for providing various trust services and cross-border interoperability across Europe. A Trust Service Provider (TSP) is the provider of a trust service, which in the case of the current study is a Certificate Authority (CA) issuing certificates for entities as a trust service. A Supervisory Body (SB) is the authority of a domain in which one or more TSPs reside, and which are regulated according to certain predefined conditions. SBs have full control over deciding the eligibility of a TSP, and in adding it to or removing it from the trusted list according to its compliance with a related bylaw.

Operating a CA requires robust security mechanisms to prevent malicious activities that could undermine the trust in the certificates it issues. Should a CA's security be

compromised, all certificates issued by it would lose their credibility, leading to widespread distrust and potential chaos in digital communication and transactions.

The CA's responsibilities also extend to maintaining a Certificate Revocation List (CRL) and, in many instances, supporting the Online Certificate Status Protocol (OCSP) for real-time validation of certificates. It's imperative for a CA to accurately track the status of its issued certificates, and in case of a security breach or compromise, immediately revoke those certificates to minimize potential damage.

2.1.3. Digital Certificates

Digital certificates, also known as X.509 certificates, form the bedrock of secure online transactions and communications. They provide a digital passport that authenticates the identity of an individual, a server, or an entity on a network, thereby fostering trust and ensuring secure data transfer.

A digital certificate contains several pieces of information: the name of the subject (the person, system, or entity the certificate identifies), a serial number, expiration dates, a copy of the subject's public key, and the digital signature of the CA [70]. The digital signature of the CA is pivotal, as it signifies that the CA verifies and backs the identity of the subject, attesting to the legitimacy of the certificate.

Digital certificates adhere to the X.509 standard, an ITU-T standard that establishes the format of public key certificates. This standardization facilitates their widespread adoption and use across diverse systems and platforms [70, 81].

Digital certificates are used extensively in various domains such as SSL/TLS for securing web traffic, securing email communications, ensuring secure remote access to servers, and many other situations where data needs to be transmitted securely over networks.

X.509 Certificates: A Comprehensive Examination

In the realm of digital communication and cryptography, X.509 certificates stand as a vital cornerstone. These certificates, as components of the Public Key Infrastructure (PKI), serve as digital passports or identity cards for entities online, helping to establish a web of trust that is integral for secure digital interactions.

Defining X.509 Certificates

X.509 is a standard [70] defined by the International Telecommunication Union (ITU) that outlines the format of public key certificates [70]. These digital certificates are used to associate a public key with an entity (a user or device) in the digital landscape. The information contained in an X.509 certificate provides the assurance that the public key indeed belongs to the named entity, thus contributing to the creation of a trustworthy digital environment.

An X.509 certificate is encoded by Abstract Syntax Notation (ASN.1) [82] and comprises several components, each carrying specific information about the entity or about the certificate itself as sample of X.509 Certificate is illustrated in Figure 2.2:

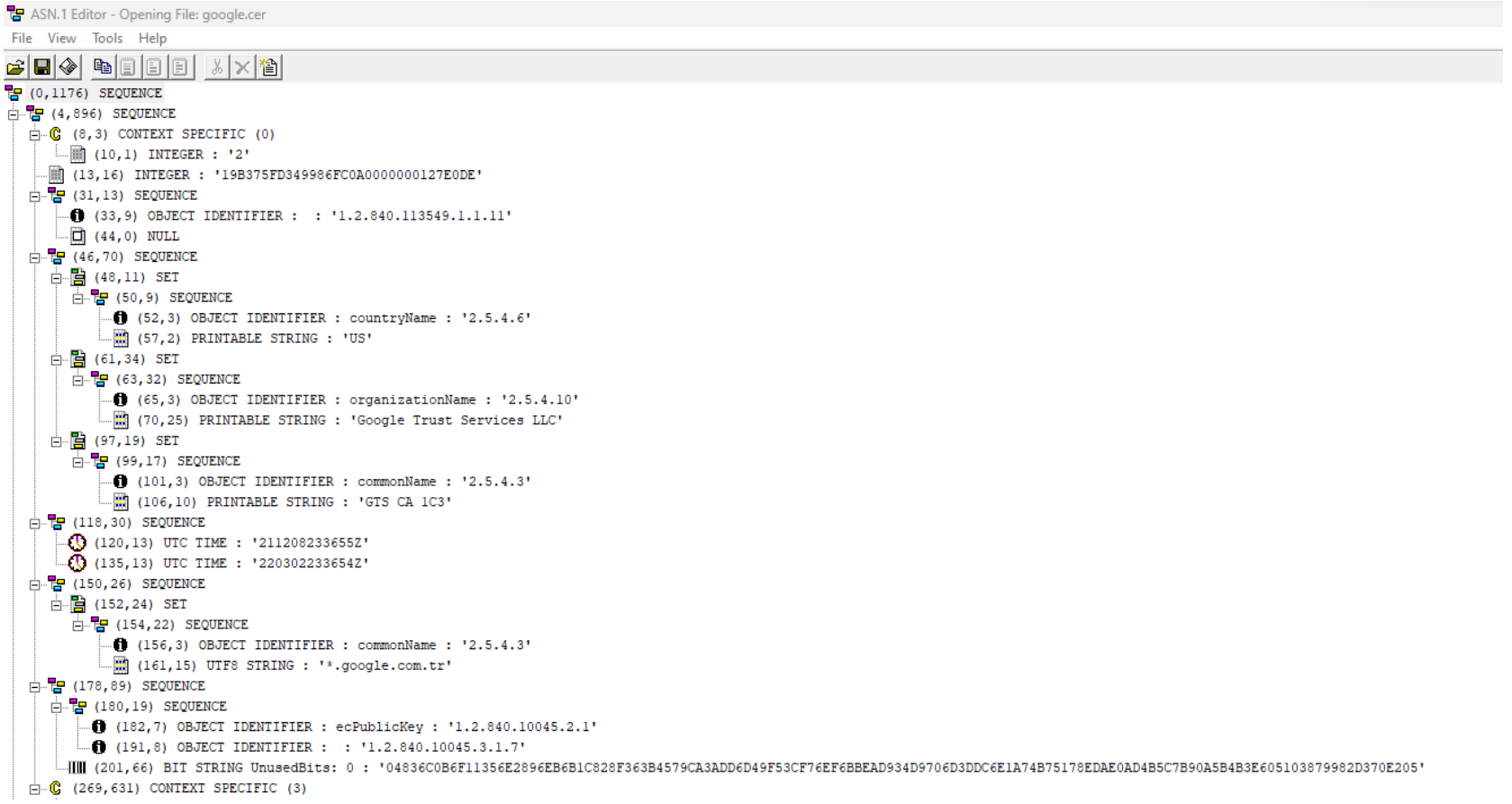

Figure 2.2 A Piece of X.509 Certificate on ASN Viewer

Version: This field indicates the version of the X.509 standard used in creating the certificate. [70]

Serial Number: Each certificate is given a distinctive identifier by the Certificate Authority (CA) with it [70].

Algorithm Information: This attribute gives details about the cryptographic algorithms used for the signature [70].

Issuer: This field specifies the name of the Certificate Authority that issued the certificate. [70]

Validity Period: This attribute specifies the duration for which the certificate is deemed valid, indicating both the 'not before' and 'not after' dates. [70]

Subject: This field contains information about the entity that the certificate is issued to. It can include details such as the common name (CN), which could be a domain name for a website, an organizational unit (OU), the organization (O), the locality (L), the state or province (ST), and the country (C) [70].

Subject Public Key Info: This is the public key of the entity and the algorithm used to generate it.

Extensions: Extensions provide additional information and capabilities beyond the core X.509 standard. They can include usage restrictions for the certificate, URL of the OCSP responder, and information about how to retrieve the CRL.

Signature: The issuer's digital signature is placed here. This signature is the result of the CA's private key signing the hash of the certificate data.

Trust Model and Validation Process

Inherent to the X.509 system is a trust model which is typically hierarchical. At the top of this hierarchy sits a root Certificate Authority (CA), an entity that is universally trusted to issue certificates correctly. Root CAs issue certificates to intermediate CAs, which can then

issue certificates to end entities or to lower-level intermediate CAs as shown in Figure 2.3. This forms a chain of trust, linking each certificate back to the trusted root CA.

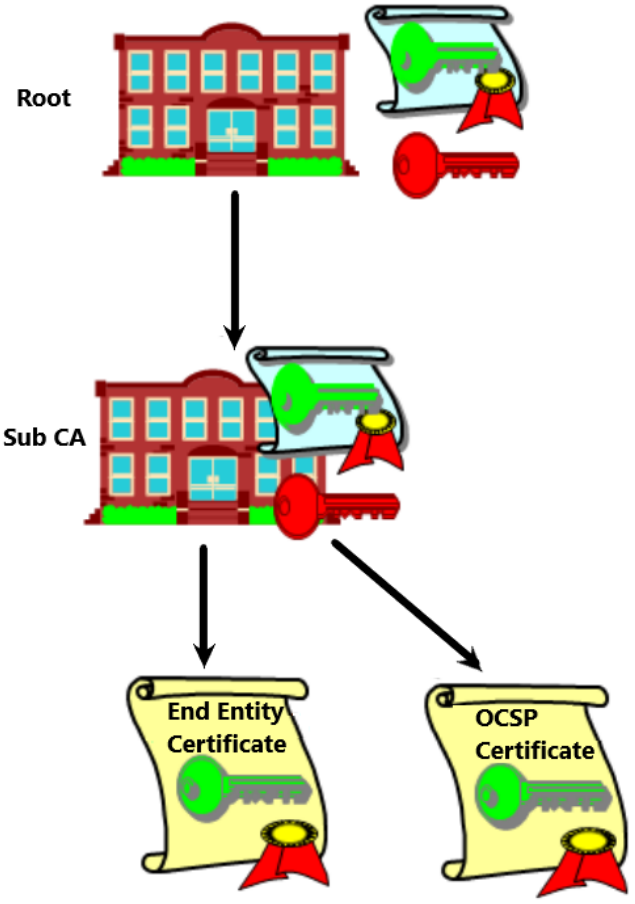

Figure 2.3 A Sample CA Hierarchy

When an X.509 certificate is presented, the receiving party can check its validity by performing a series of checks [70]:

Expiration Check: Confirm that the current date falls within the designated validity period of the certificate [70].

Signature Verification: Validate that the certificate's signature can be accurately decrypted using the public key of the issuer, ensuring it aligns with the hashed certificate data [70].

Revocation Status: Investigate whether the certificate is listed in the Certificate Revocation List (CRL) or utilize the Online Certificate Status Protocol (OCSP) to determine its revocation status [70].

Chain of Trust Validation: Scrutinize the entire certificate chain back to a trusted root Certificate Authority (CA). Each certificate in the chain must successfully pass checks for expiration, signature, and revocation.

X.509 certificates are fundamental to securing digital communications and transactions. They authenticate entities, facilitate encrypted connections, and support non-repudiation in digital transactions. However, the management and validation of X.509 certificates also pose challenges. These include ensuring the correct and timely revocation of compromised certificates, maintaining the security of CAs, and dealing with the potential computational and network overheads of certificate validation. [70]

2.1.4. Certificate Revocation List

While the issuance of digital certificates is a critical aspect of a CA's responsibilities, the management of those certificates, including revocation, is equally important. The Certificate Revocation List (CRL) is a mechanism that supports this responsibility [70].

A CRL is a compilation of digital certificates that the issuing Certificate Authority (CA) has invalidated before their intended expiration date, as depicted in Figure 2.4 and Figure 2.5. This revocation could be due to several reasons, such as the private key associated with a certificate being compromised, the CA itself being compromised, or the end of the relationship between the certificate's subject and the entity it represents.

CRLs are periodically updated and published in a public repository accessible to anyone who wishes to control the revocation status of digital certificates. However, this poses its own set of challenges. The primary challenge is the time lapse between CRL updates, which could potentially lead to situations where a recently revoked certificate is still viewed as valid by relying parties [70].

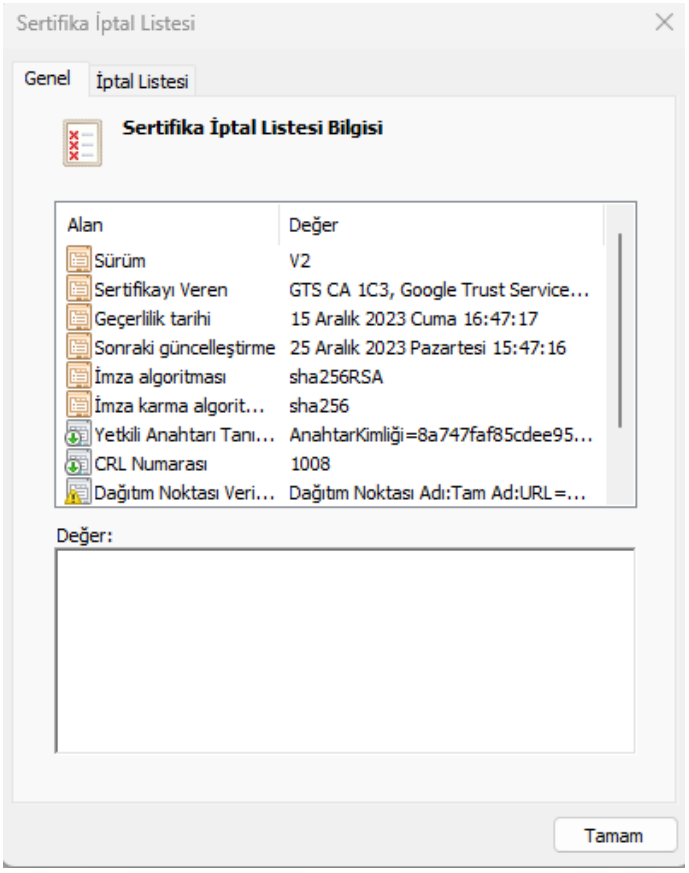

Figure 2.4 CRL View

2.1.5. Online Certificate Status Protocol

The Online Certificate Status Protocol (OCSP) is an Internet protocol that is utilized to verify the status of a digital certificate. Designed as an improvement over the Certificate Revocation List (CRL) method, OCSP provides real-time status checks, reducing the latency inherent in CRLs and offering more timely detection of compromised certificates [83].

The OCSP was developed by the Internet Engineering Task Force (IETF) and is described in detail in the RFC 6960 standard [83]. Its primary function is to convey information about the revocation status of a digital certificate, directly influencing the reliability of transactions or communications reliant on that certificate.

To overcome the limitations of CRLs, the OCSP was developed. It is an protocol for obtaining the revocation status of a X509 certificates in real-time [83]. Instead of

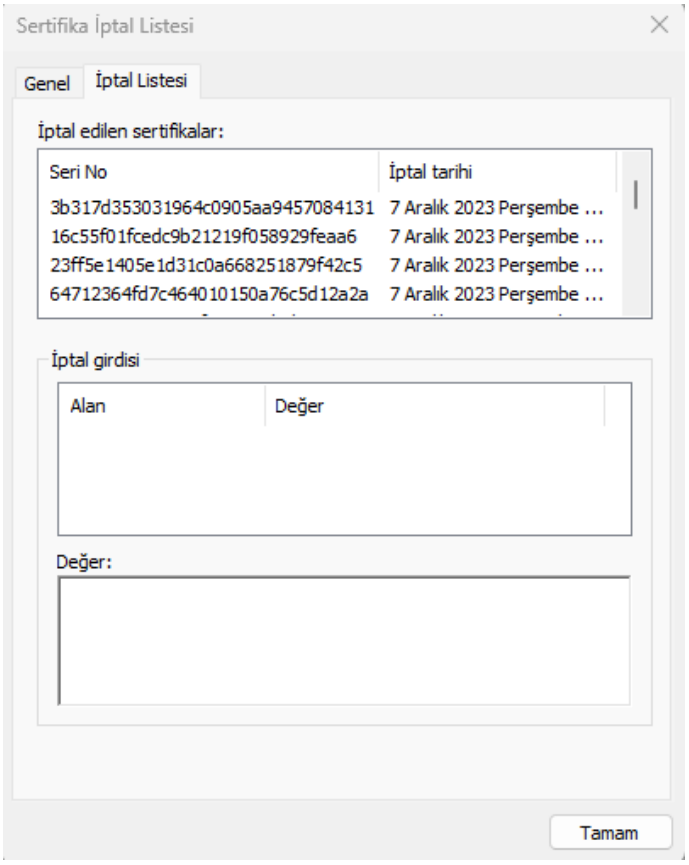

Figure 2.5 CRL View Revoked Certificates

downloading a possibly large CRL, a client can send a query to an OCSP responder (typically maintained by the CA) for the status of a single certificate. The OCSP responder then returns a response indicating the certificate's status - good, revoked, or unknown [83].

OCSP presents a significant improvement over the use of CRLs due to its ability to provide real-time feedback [83]. This improvement, however, does not come without its own challenges. The most critical challenge is the increased load on the OCSP servers, especially in large-scale environments with numerous certificates to check. Hence, OCSP must be implemented strategically to balance between efficiency and server load. Furthermore, privacy concerns can arise as the CA can theoretically track which sites a user is visiting by the OCSP requests it receives.

OCSP Functioning

The OCSP system consists of two main components: the OCSP client and the OCSP responder. The client is typically integrated into the device or application of the party that wants to verify the status of a digital certificate, such as a web browser. The OCSP responder is usually managed by a Certificate Authority (CA) or a trusted partner [83].

When a client needs to check the status of a certificate, the flow is started as shown in Figure 2.6.

1. Client gets OCSP address from Certificate's Authorithy Information Access attribute
2. Client sends OCSP request
3. OCSP server genarates OCSP Response and sign it with its private key.
4. OCSP server sends OCSP Response back to client

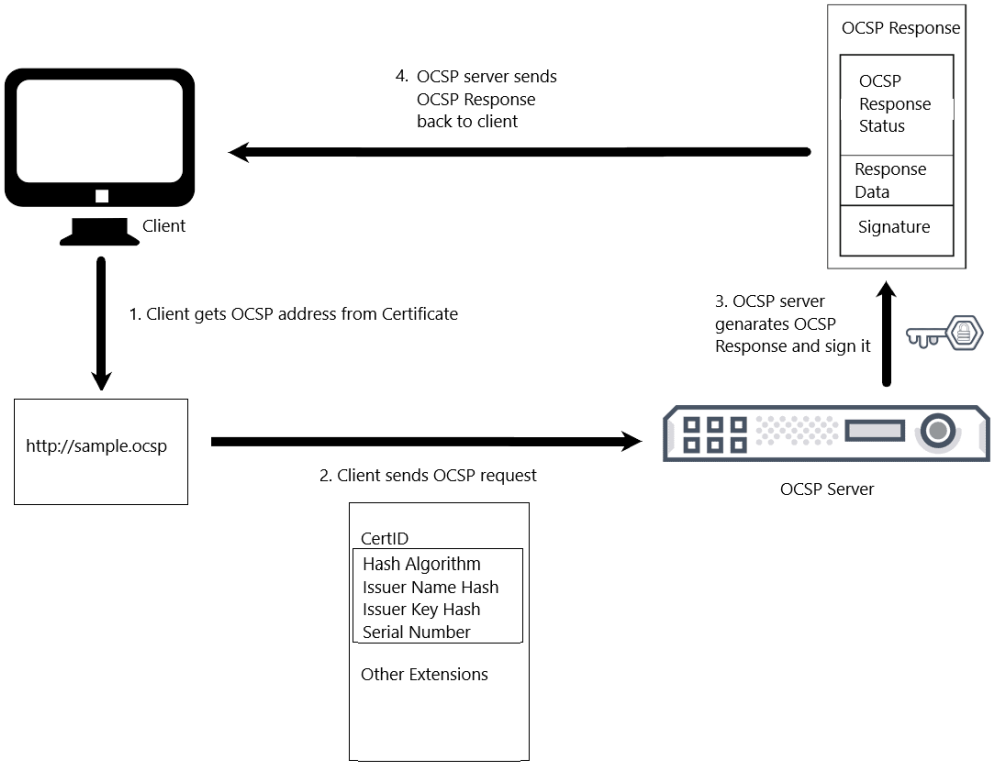

Figure 2.6 OCSP Simple Flow

The 'good' status means that the certificate is still valid. The 'revoked' status indicates that the certificate is no longer trusted and should not be used. The 'unknown' status means the responder doesn't have any information about the certificate's status.

The OCSP response also contains a timestamp indicating when the status was last updated, and it is digitally signed by the OCSP responder to ensure its authenticity and integrity.

Advantages of OCSP

Compared to CRLs, OCSP offers several benefits. First, it provides near real-time certificate status checking, which is a significant advantage over CRLs, which might only be updated periodically. This capability can be critical in limiting the potential misuse of a compromised certificate.

Second, OCSP can be more efficient than CRLs. Instead of downloading a potentially large CRL, a client only needs to send a small request and receive a small response for each certificate it wants to verify.

Challenges and Considerations

Despite its advantages, the use of OCSP also presents some challenges. For instance, the availability and reliability of OCSP responders are critical. If a responder becomes unreachable or is slow to reply, it can degrade the performance of applications and services relying on it for certificate status checks.

Privacy is another concern with OCSP. The OCSP requests sent from a client to a responder could potentially allow the responder (or anyone observing the network traffic) to track which servers a client is connecting to, as the request contains the certificate's serial number.

To mitigate these challenges, various optimizations and extensions to the OCSP protocol have been developed. One such solution is OCSP stapling, also known as the TLS Certificate Status Request extension. This method allows the server (e.g., a web server) to obtain a timely OCSP response for its own certificate and send ("staple") it to clients during the initial TLS handshake. This approach reduces the number of OCSP requests that clients need to

send, improves privacy, and eliminates potential delays caused by clients having to contact OCSP responders directly.

OCSP Revocation Nocheck

The OCSP Revocation Nocheck is an extension to the OCSP that allows for certain optimizations in certificate status checking processes. Before delving into the specifics of the Nocheck extension, it is essential to understand the context in which it operates, i.e., the realm of certificate revocation and status checking.

Certificates, once issued by a Certificate Authority (CA), can be revoked for a variety of reasons, such as compromise of the certificate's private key or cessation of the certified operations. To provide a mechanism for entities to verify the current status of a certificate, CAs publish Certificate Revocation Lists (CRLs) and also, in most cases, operate an OCSP responder. Entities can query the OCSP responder to check if a specific certificate is still valid or has been revoked.

However, a concern arises when we consider the certificates of the OCSP responder itself. The entity must verify the OCSP responder's certificate's validity, which in turn could involve checking the status of another OCSP responder, leading to a potentially infinite loop. The situation gets even more complicated when an entity must validate multiple certificates, each requiring a separate OCSP request.

Herein lies the role of the OCSP Revocation Nocheck extension. This extension, when included in an OCSP responder's certificate, signifies that the entity need not check the revocation status of this certificate. It is essentially a trust token, indicating that the certificate can be trusted for the purpose of OCSP response signing, without needing a revocation check. This extension streamlines the certificate status checking process and can help reduce network traffic and latency by eliminating unnecessary OCSP requests.

However, it is worth noting that the use of the Nocheck extension must be approached cautiously. Since the Nocheck extension exempts a certificate from revocation checking, it is crucial to ensure that this certificate is appropriately protected and monitored. The private

key corresponding to this certificate must be securely stored and used only for signing OCSP responses. Any potential compromise of the key must lead to immediate revocation of the certificate and issuance of a new certificate with a new key.

In summary, the OCSP Revocation Nocheck extension is a useful tool for optimizing certificate status checking processes. It helps reduce the complexity and resource consumption involved in checking the validity of certificates, particularly in cases involving multiple or nested checks. However, its usage must be balanced with appropriate security measures to ensure the integrity and trustworthiness of the overall certificate status checking process.

2.1.6. Time Stamp

Time-Stamping Services and Protocols: An Extensive Study

In the domain of digital security, time-stamping is an invaluable service. Time-stamping refers to the process of securely recording the creation time of a digital data item (like a document, transaction, or electronic signature) to provide proof that it existed at a particular point in time.

A time-stamp [84] is a data structure that contains a representation of a particular point in time and a representation of the digital data item, typically in the form of a hash. This structure is then digitally signed by a trusted third party known as a Time Stamping Authority (TSA) to certify its accuracy.

A time-stamp has several key components:

1. *Time*: This is the time value that represents the point in time when the time-stamp was created. The accuracy and precision of this value depend on the TSA's time source.

2. *Data Representation:* Typically, this is the hash [73–77] of the digital data item being time-stamped. This approach allows any size data item to be time-stamped while preserving the confidentiality of the original data.

3. *Signature:* The TSA's digital signature on the time and data representation ensures the integrity and authenticity of the time-stamp.

Time-Stamping Protocol

The most widely recognized time-stamping protocol is defined in the IETF's RFC 3161 [84], "Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)". This protocol defines how a client can request a time-stamp from a TS Server and how the TS Server responds to such a request.

The protocol works as shown in the Figure 2.7 follows:

1. The client first computes the hash of the digital data item to be time-stamped.

2. The client sends a TimeStampReq (Time-Stamp Request) to the TS [84]. This request contains the hash and may also include additional parameters such as the desired accuracy.

3. The TS generates a time-stamp, which includes the current time, the hash from the request, and possibly additional information. The TS then signs this time-stamp with its private key.

4. The TS sends a TimeStampResp (Time-Stamp Response) back to the client. This response includes the signed time-stamp and possibly additional information about the status of the request.

Importance and Applications of Time-Stamps

Time-stamps serve several important functions in digital communications and transactions:

Proof: Time-stamps can be used to prove that a digital data item (like an electronic signature) existed at a particular time, supporting non-repudiation of transactions or events.

Data Integrity: By time-stamping a hash of digital data, one can prove that the data has not been altered since it was time-stamped.

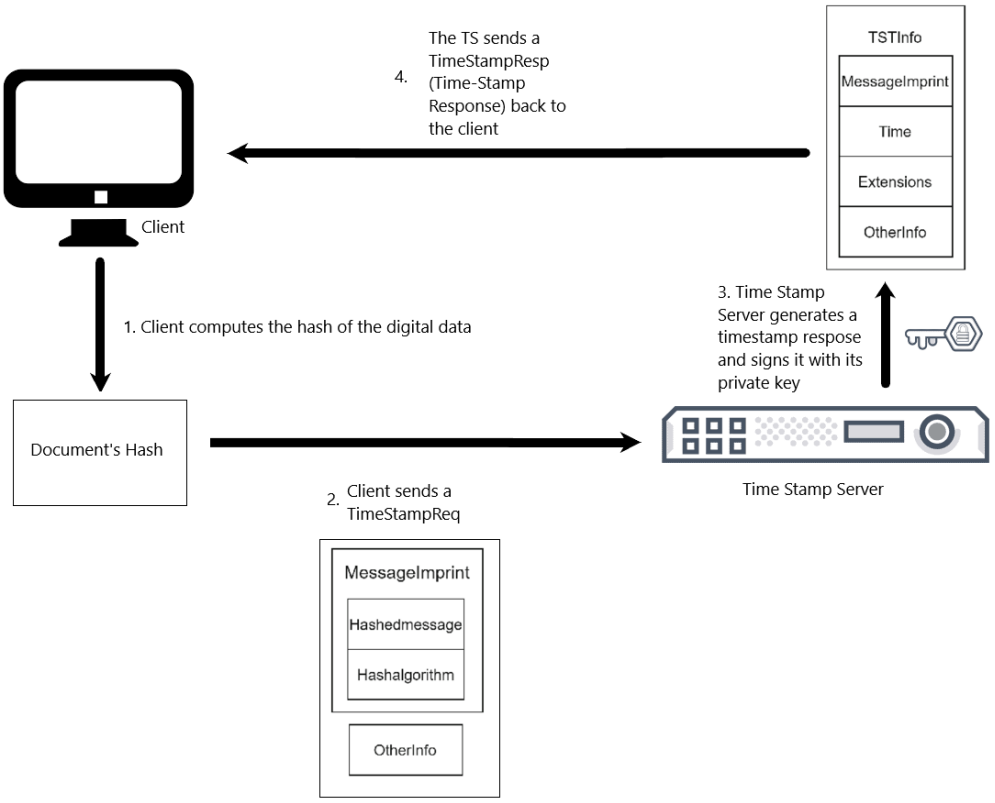

Figure 2.7 Time Stamping Flow

Document Authenticity: Time-stamps can be used to demonstrate the authenticity of digital documents or other data items, as any change in the document after it was time-stamped will cause the hash to change.

Long-Term Validity of Signatures: Time-stamps are an essential component of long-term signature formats such as XAdES, CAdES, and PAdES. By time-stamping a signature and then periodically time-stamping the time-stamps, one can maintain evidence of the signature's validity even as the cryptographic technology evolves.

Despite its various advantages, time-stamping services must be deployed carefully, considering the trustworthiness of the TSA, the security of the time-stamping system, and the legal and regulatory requirements of the specific application.

2.1.7. Digital Signature: An Overview

In an increasingly more digitalized world, the warranty of authenticity, integrity, and non-repudiation in digital transactions is paramount. electronic signatures, a cornerstone of Public Key Infrastructure (PKI), facilitate these assurances. Rooted in cryptographic concept and serving as a electronic analog to handwritten signatures, they provide an necessary tool in organising trust within the digital surroundings.

A digital signature is a mathematical method hired to confirm the genuineness and integrity of a message, software program, or digital record. It involves growing a unique signature for the signer, based totally on a hash of the document and the signer's personal key in a two-step system of hashing and encryption. when this digital signature is effectively proven using the signer's public key, it confirms that the document has not been altered since it turned into signed and verifies the identification of the signer.

Electronic signatures generally encompass three algorithms: key generation, signing, and signature verification.

Key era: This set of rules creates a pair of keys - a private key that stays confidential to its owner, and a corresponding public key this is publicly available. these keys are mathematically linked, and at the same time as the private key's used to create the electronic signature, the general public key's used to affirm it.

Signing: This set of rules produces a signature upon receiving the message and the personal key. The procedure includes growing a hash (digest) of the message the use of a selected hash function. The personal key then encrypts this hash, developing a unique signature for that precise aggregate of the message and the personal key.

Signature Verification: This algorithm checks the authenticity of the signature upon receiving the message, its signature, and the signer's public key. It makes use of the public key to decrypt the signature returned into the hash and then re-computes the hash of the received

message. If the decrypted hash and the newly computed hash healthy, it verifies the integrity of the message and the authenticity of the signer.

Electronic signatures provide severa benefits in electronic communication and transactions, encompassing authentication, integrity, and non-repudiation.

Authentication: Electronic signatures authenticate the identity of the celebration worried. whilst the signature originates from the private key of the signer, it authenticates that the signer is certainly the originator of the report or message, thereby mitigating the danger of impersonation or identification fraud.

Integrity: By means of growing a completely unique signature based totally on the record and the private key, electronic signatures make sure the integrity of the signed record. Any alteration, but minor, made to the document after signing will bring about a one-of-a-kind hash throughout verification, signaling that the file has been tampered with.

Non-Repudiation: Electronic signatures offer a assure that the signer can't convincingly deny the authenticity of their signature, establishing non-repudiation. This aspect is crucial in felony and economic transactions wherein proof of participation is vital.

Despite the security and believe benefits provided with the aid of digital signatures, sure challenges need to be addressed. The secure garage of the non-public secret's of extreme significance. If the non-public key is compromised or misplaced, it may cause extreme outcomes like identity theft or lack of ability to get admission to one's own digital resources.

Another concern is the scalability and efficiency of digital signature algorithms. As the number of users in a system grows, so too does the number of keys, which can lead to complex and time-consuming key management. In addition, the computational intensity of cryptographic operations can be a limiting factor in systems with limited processing power.

Moreover, legal and regulatory aspects of digital signatures can vary widely between jurisdictions. While digital signatures have legal standing in many regions, the lack of universally accepted standards and regulations can pose challenges, particularly in cross-border transactions.

As we continue to progress further into the digital era, the importance of trust and security in online transactions and communications cannot be overstated. Digital signatures provide a robust mechanism for confirming the authenticity and integrity of digital communications, thereby fostering trust and security within the digital ecosystem. However, the effective implementation and management of digital signatures require continuous improvements and adaptations to combat emerging threats and challenges. As a result, digital signatures remain a dynamic and vital research area in the field of digital security and cryptography.

CAdES (CMS Advanced Electronic Signatures)

CAdES, an acronym for CMS Advanced Electronic Signatures, is a set of extensions to the original Cryptographic Message Syntax (CMS) [85] standard, enhancing it for advanced electronic signatures. The CMS standard is widely used for digitally-signed or encrypted messages in Public Key Infrastructure (PKI) [69] and Secure/Multipurpose Internet Mail Extensions (S/MIME) for email encryption [70].

CAdES is designed to cater to various legal and audit requirements that could not be satisfied by CMS signatures. It extends the CMS signature format to provide additional signed and unsigned properties, such as signature timestamps, certificate reference attributes, and a mechanism for including the complete validation data (i.e., certificates and revocation status information) needed to validate the signature.

CAdES ensures that digital signatures remain reliable for long periods, even if the signer's certificate expires or is revoked, or the cryptographic algorithm becomes obsolete. CAdES is defined by the European Telecommunications Standards Institute (ETSI) in the standard ETSI TS 101 733 [50].

Signed and Unsigned Attributes

In the context of a CAdES signature, attributes refer to additional pieces of information that can be associated with a signature.

Signed attributes are protected by the signature, meaning that if they are modified, the signature validation will fail. These can include:

*Content type: [50]*The type of the signed data.

Message Digest: [50] The hash of the signed data.

Signing time: [50] The time at which the signer claims to have created the signature.

Signing certificate: [50] A reference to the certificate that should be used to validate the signature. Unsigned attributes are not protected by the signature, meaning they can be modified or added after the signature is created. This allows additional information to be associated with the signature over time, such as evidence of validity. These can include:

Signature time-stamps: [50] Time-stamp tokens (TSTs) obtained from a Time Stamping Authority (TSA) to prove that the signature existed at a particular time. Complete certificate references: Information required to retrieve the certificates needed to validate the signature. Revocation values: Information about the revocation status of the certificates. Signature Types

Various signature types or levels are defined under the CAdES standard to cater to different requirements. Here are the details of some of the key signature types:

Signature types in electronic signatures are classified according to the authenticity lifetime of the signature. The most commonly used signatures, namely *Basic Electronic Signature (BES), Electronic Signature with Time (ES-T), Extended Long Electronic Signature (ESX Long), and Archival Electronic Signature (ESA)* [50] are explained below:

Basic Electronic Signature (BES) signature is the most simple signature type. There is no trusted time information that guarantees the date of signature creation. The validity of BES signature is restricted with the lifetime of the end entity certificate. The signature cannot be verified after the certificate has revoked or the expired. So, time stamp is very important to expand life time of signature. Time information in the BES signature may be added as a signing time attribute which is not trusted. The cryptographic signature value in the BES signature is calculated by concatenation then hashing of *Signed Attributes* field as seen in Figure 2.8 [50].

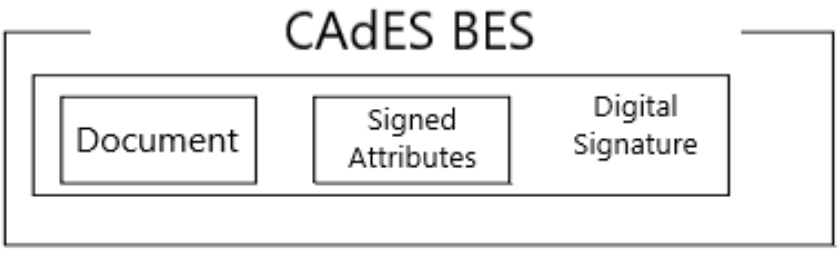

Figure 2.8 BES

The mandatory signed attributes for BES signature are as follows:

- Content Type : [50] The type of the signed content.
- Message Imprint : [50] The hash value of the content to be signed.
- Enhanced Security Services - Signing Certificate V2 [50] : References the certificate of signer and limits the set of certificates to be used in validation of signature. The *issuerSerial* field present in *ESSCertID* shall match the *issuerAndSerialNumber* in *SignerIdentifier* field of the *SignerInfo*. In addition, the SHA-1 hash of the certificate shall match the certHash from ESSCertID [85].

Electronic Signature with Time (ES-T) [50] consists of the time-stamp information that signifies the date when the signature was legally created. The format of BES signature is shown in Figure 2.9. The signature creation time is protected with a time-stamp, if the certificate and signature are valid at the point of timestamp, the signature is also valid even the certificate is revoked or expired after the creation. It is not enough to have a time-stamp for long-term validation, it is a need to have all relevant revocation data and issuer certificates are required. For this reason, the ES-X LONG signature is recommended [50].

Extended Long Electronic Signature (ESX Long) [50] consists of the ES-T signature, the root and subordinate certificate values/references of the certificate authority and the revocation data/references (CRL/OCSP responses) for signature verification in order to enable signature validation for a long period of time as shown in Figure 2.10 [50]. In

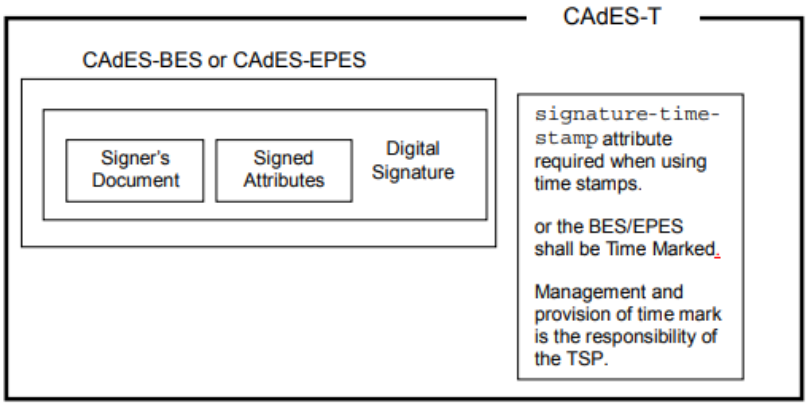

Figure 2.9 EST

signature validation, all certificate and revocation data which are added to the signature are used so connecting to any external system may not be an issue.

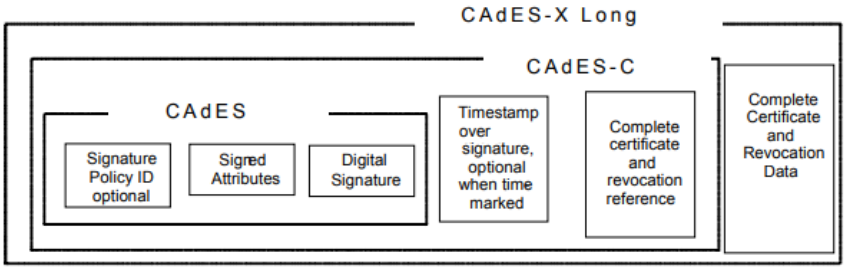

Figure 2.10 ESX Long

Archival Electronic Signature (ESA) [50] is constructed on ESX Long signature in order to increase life time of signature than the validity period of the related root, subordinate, OCSP and time-stamp certificates as seen in Figure 2.11 [50]. Archiving process should be done before the CA's certificate expiration/revocation or announcement soon after used algorithms are in danger. Archival is a continuous process before last used time-stamp in archiving is expired it need to be repeated with a new timestamp [50].

Signature Sizes Issues for Long Term Signatures

In the realm of CAdES (CMS Advanced Electronic Signatures) [50], the significance of signatures extends beyond the cryptographic assurance of data integrity. This explanation

delves into the implications of including complete certificate values and revocation information within CAdES signatures.

Electronic signatures, including those following the CAdES standard, often encompass the entire certificate chain. This entails the inclusion of not only the signer's certificate but also the intermediate and root certificates forming the chain up to the trusted root certificate. This comprehensive approach ensures that anyone validating the signature has access to the full chain of trust, thus establishing the authenticity of the signer's identity. Integral to CAdES signatures is the incorporation of revocation information. This information provides details about the current validity status of certificates within the signature. Specifically, it addresses whether the signer's certificate or any intermediate certificates in the chain have been revoked. This inclusion is paramount for assessing the real-time trustworthiness of the signature.

The inclusion of complete certificate values and revocation information contributes to the expansion of CAdES signatures. This phenomenon, often termed "signature size overhead," warrants consideration due to its impact on transmission time, storage requirements, and computational resources during signature validation. As such, careful deliberation is necessary when balancing the necessity of comprehensive information with potential implications for signature size.

The inclusion of complete certificate values and revocation information assumes heightened importance when considering the long-term validation of signatures. Certificates may be revoked over time, and having this information embedded within the signature ensures that validation remains robust, even in scenarios where the original certificate authority is inaccessible or has undergone changes in status.

Each signature type caters to different use cases and provides a different level of assurance. While CAdES-BES [50] and CAdES-EPES [50] might be sufficient for simple use cases, CAdES-X-Long [50] and CAdES-A [50] are better suited for situations where the signature might need to be validated many years in the future.

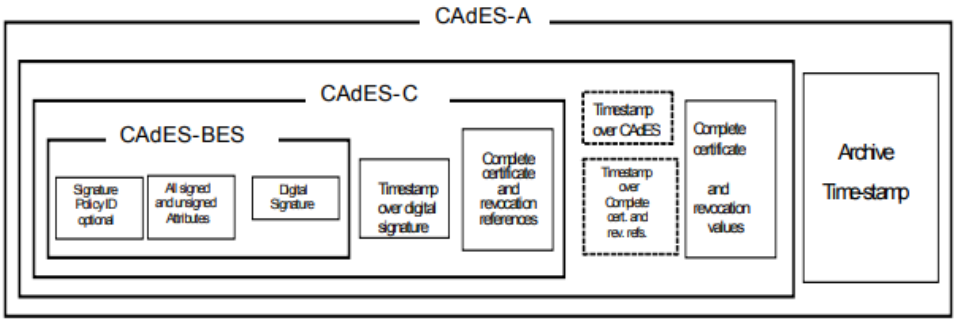

Figure 2.11 ESAv2

Multiple Signatures for Electronic Signatures

Electronic signatures play a pivotal role in modern transactions, offering efficient and secure authorization methods. In this chapter, we explore the concept of multiple signatures, covering parallel and serial signing approaches, their applications, advantages, and key considerations.

In the use of electronic signatures, a content (document, bank instruction, etc.) may be signed not only by a single signer but also by multiple signers. In such cases, multiple signing, comprising parallel signing or serial signing, is utilized. Multiple signing is defined in the CAdES [50], XAdES [52], and PAdES [53] standards.

Parallel Signatures: Parallel signatures involve simultaneous signing by multiple parties, streamlining transactions for speed, flexibility, scalability, and allowing multiple independent signatures over the same data as shown in Figure 2.12.

In CAdES [50], each signer signs a hash value consisting of the concatenation of the message digest, content type, and ESS-Signing-Certificate. This process ensures a secure and verifiable parallel signing approach, contributing to the overall integrity and authenticity of digital transactions.

Choosing between parallel, serial, or multiple signatures depends on specific workflow requirements, transaction nature, and user preferences. Implementing multiple signing processes contributes to efficient, secure, and streamlined digital transactions.

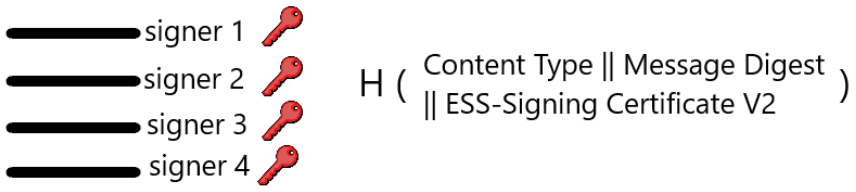

Figure 2.12 Parallel Signature

Serial Signatures: Serial signatures follow a sequential signing process, ensuring a controlled and ordered approach, creating a clear audit trail, and providing flexibility for workflows requiring specific sequences as shown in Fig. 2.13.

Serial signatures in CAdES involve a sequential signing process where each signer signs the cryptographic signature value of the previous signer's signature value. This ensures a controlled and ordered approach, creating a clear audit trail, and providing flexibility for workflows requiring specific sequences.

This sequential signing method adds an additional layer of verification, as each signature is directly linked to and dependent on the cryptographic signature value of the preceding signer. This approach enhances the integrity and traceability of digital transactions, making it suitable for workflows with strict sequential requirements.

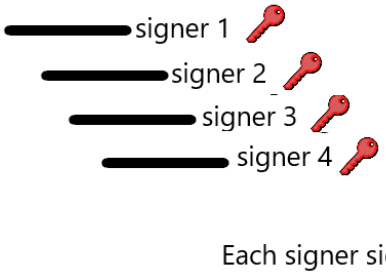

Figure 2.13 Serial Signature

XAdES (XML Advanced Electronic Signatures)

XAdES stands for XML Advanced Electronic Signatures. It's a set of extensions to the XML-DSig recommendation by the World Wide Web Consortium (W3C) [52], making it suitable for advanced electronic signatures. Like CAdES, XAdES also ensures long-term validation of digital signatures and caters to a wide range of legal and audit requirements.

XAdES is beneficial because XML is a universal standard for data interchange [52], and XML-DSig [86] enables the signature of multiple types of resources described in XML. XAdES extends XML-DSig to provide advanced electronic signatures capable of satisfying legal requirements. It also introduces additional signed and unsigned properties for advanced signatures.

XAdES supports multiple forms, including XAdES-BES (Basic Electronic Signature) [52], XAdES-EPES (Explicit Policy Electronic Signature) [52], XAdES-T (with time-stamp) [52], and more complex forms like XAdES-C (Complete) [52], XAdES-X (Extended), XAdES-XL (Extended Long-Term), and XAdES-A (Archival) [52]. The ETSI standard ETSI TS 101 903 defines XAdES [52].

PAdES (PDF Advanced Electronic Signatures)

Digital signatures play an essential role in affirming the authenticity and integrity of digital documents. As we increasingly transition towards paperless systems, the necessity for a robust, legally sound, and universally acknowledged digital signature standard has become paramount. PAdES, or PDF Advanced Electronic Signatures [53], is a standard that addresses this need, especially concerning the ubiquitous PDF documents.

PAdES is a set of extensions for the PDF specification that enable it to support advanced electronic signatures. This standard, devised by the European Telecommunications Standards Institute (ETSI), extends the ISO 32000-1 PDF specification to provide a digital signing format that is compliant with the eIDAS regulation – the European Union legislation that establishes standards for electronic identification and trust services. The primary objective behind PAdES is to facilitate the creation of signatures that are not just legally

binding, but are also capable of remaining verifiable over the long term as technologies evolve and cryptographic algorithms become obsolete.

PAdES Signature Formats

PAdES defines several levels of electronic signatures, each providing different features. These are referred to as PAdES B-Level, PAdES T-Level, PAdES LT-Level, and PAdES LTA-Level.

- **PAdES B-Level (Basic Level) [53]**: This level offers the minimal requirements for an advanced electronic signature. It ensures the integrity of the document and validates the signer's identity.
- **PAdES T-Level (Time-stamped Level) [53]**: In addition to the B-Level features, this level incorporates a trusted time stamp on the signature, indicating that the signature was applied at a particular point in time.
- **PAdES LT-Level (Long-Term Level) [53]**: LT-Level incorporates the features of the T-Level and additionally provides a way to access certain certificate-related information needed for validation, such as the certificate's status (revocation information) at the time of signing.
- **PAdES LTA-Level (Long-Term with Archival Level) [53]**: This level further extends the LT-Level by adding an archival timestamp. This timestamp aids in ensuring the long-term validity of the signature, even when the certificate used for the original signature has expired or the hashing algorithm used has been broken.

Significance of PAdES

PAdES is a significant advancement in digital document signing for several reasons:

1. **Long-Term Validation (LTV) [53]**: PAdES supports LTV, making it possible to verify a document's signature long after the technology used to create the signature has

become obsolete. This feature is vital for documents that need to be archived and accessed over extended periods.

2. **Legal Compliance**: PAdES is compliant with the eIDAS regulation, which provides a legal framework for electronic signatures within the EU. This compliance means that a PAdES signature has the same legal status as a handwritten signature in the European Union and is acknowledged in international transactions.

3. **Interoperability**: As an extension to the ISO-standardized PDF format, PAdES signatures inherently support interoperability. Documents signed with PAdES can be viewed and verified using a wide range of PDF viewers, making PAdES suitable for cross-border and cross-organization transactions.

4. **Versatility**: PAdES supports various signature types, including visible signatures, invisible signatures, and certified signatures, thereby enabling users to select the signature type that best suits their use case.

PAdES signifies a considerable stride in the realm of digital signatures, specifically tailored for PDF documents. By combining legal recognition, long-term validity, and extensive interoperability, it provides a robust framework for the digital signatures of the future. As the digital landscape continues to evolve, it is expected that PAdES will play a significant role in shaping digital signature standards.

XAdES (XML Advanced Electronic Signatures) [52]

XAdES (XML Advanced Electronic Signatures) [52] is a set of extensions to XML-DSig (XML Digital Signature) [86], a recommendation by the World Wide Web Consortium (W3C) and the Internet Engineering Task Force (IETF) for creating digital signatures in the XML data format. Developed by the European Telecommunications Standards Institute (ETSI), XAdES enriches the XML-DSig [86] standard to meet the advanced electronic signature requirements defined in the European Union's eIDAS regulation [80].

The adoption of XAdES provides several distinct advantages. Firstly, it allows for the creation of legally binding signatures. Secondly, it caters to the need for long-term signature

validation, ensuring that the signature remains verifiable as technology progresses and cryptographic algorithms become outdated. Lastly, by utilizing the XML format, XAdES signatures inherently support data interoperability across different systems and platforms.

XAdES identifies multiple levels of electronic signatures, extending the capabilities of XML-DSig and offering varying degrees of security and assurance:

- **XAdES-BES (Basic Electronic Signature) [52]**: This level equates to XML-DSig, supplemented with specific signed attributes such as the signing certificate, and represents the minimal requirements for an advanced electronic signature.
- **XAdES-EPES (Explicit Policy Electronic Signature) [52]**: This level extends XAdES-BES by associating the signature with an explicit policy, defined by the signer.
- **XAdES-T (Time-stamped) [52]**: This level extends XAdES-EPES with the addition of a timestamp to indicate when the signature was created.
- **XAdES-C (Complete) [52]**: This level extends XAdES-T by incorporating references to the validation data required for the signature.
- **XAdES-X (Extended) [52]**: This level adds a second timestamp to the references incorporated in XAdES-C.
- **XAdES-XL (Extended Long-Term) [52]**: This level extends XAdES-X by including the complete validation data required for the signature.
- **XAdES-A (Archival) [52]**: This level is designed for long-term archiving and extends XAdES-XL with periodic timestamping to protect against future cryptographic attacks.

XAdES is a comprehensive and robust digital signature standard that provides for the creation, verification, and preservation of legally binding signatures in the XML format. Its multi-level design offers flexibility, allowing entities to choose the level of security that best suits their specific use cases. Furthermore, XAdES's compatibility with eIDAS and its ability to ensure long-term signature validation positions it as an integral part of the evolving digital signature landscape.

2.1.8. Threshold Signature Scheme

The Threshold Signature Scheme (TSS) represents a pivotal advancement in the realm of cryptographic protocols, addressing critical issues related to security, privacy, and decentralization. At its core, TSS divides the responsibility of signing a document or transaction among multiple parties, requiring only a subset of them (the threshold) to collaboratively generate a valid signature. This scheme enhances security by distributing trust and making it significantly more challenging for attackers to compromise the signing key.

TSS is defined by two main parameters: *(n)* and *(t)*. Here, *(n)* represents the total number of parties involved in the scheme, while *(t)* denotes the minimum number of parties required to produce a valid signature. The scheme ensures that any group of *(t)* or more parties can collaboratively sign a document, whereas any group smaller than *(t)* cannot generate a valid signature, thus enforcing both security and flexibility in digital signature generation.

The applications of Threshold Signature Schemes are vast and varied, spanning multiple domains where security and distributed trust are paramount. In blockchain technology, TSS facilitates secure and decentralized control of assets, enabling cryptocurrency wallets and platforms to distribute signing authority among multiple stakeholders. This not only mitigates the risk of single points of failure but also aligns with the decentralized ethos of blockchain networks.

Moreover, TSS is instrumental in enhancing the security of multi-factor authentication systems, secure voting systems, and distributed ledger technologies. By requiring a subset of participants to authenticate a transaction or vote, TSS adds an additional layer of security and resilience against hacking attempts and unauthorized acces

One of the hallmark advantages of a Threshold Signature Scheme is its contribution to enhancing security. By distributing the signing capability, TSS significantly reduces the risk of key compromise. Even if an attacker manages to obtain a few of the signing keys, without meeting the threshold, these keys are essentially useless. Furthermore, TSS promotes fault

tolerance in cryptographic operations, ensuring that the loss or compromise of a subset of participants does not halt or significantly disrupt the system's functionality.

In addition to security, TSS enhances privacy. Since the signature is generated collaboratively without reconstructing the complete signing key at any point, individual keyholders' privacy is maintained, and the key itself remains protected against exposure.

2.2. Blockchain

Blockchain is an immutable, distributed ledger of transactions whereby transactions reside in so-referred to as blocks, and ledgers are distributed across peer-to-peer networks. these networks generally rely upon Merkle trees, which have non-leaf nodes that are categorized with the hash of all in their toddler nodes. hence, it lets in participants to create unique, concise, and quick verifiable proof. The Merkle timber develop logarithmically in terms of the variety of its leaves. every blocks include block headers, previous header hash, merkle root and every leaf node is a hash of transactional data as visible within the figure 2.14.

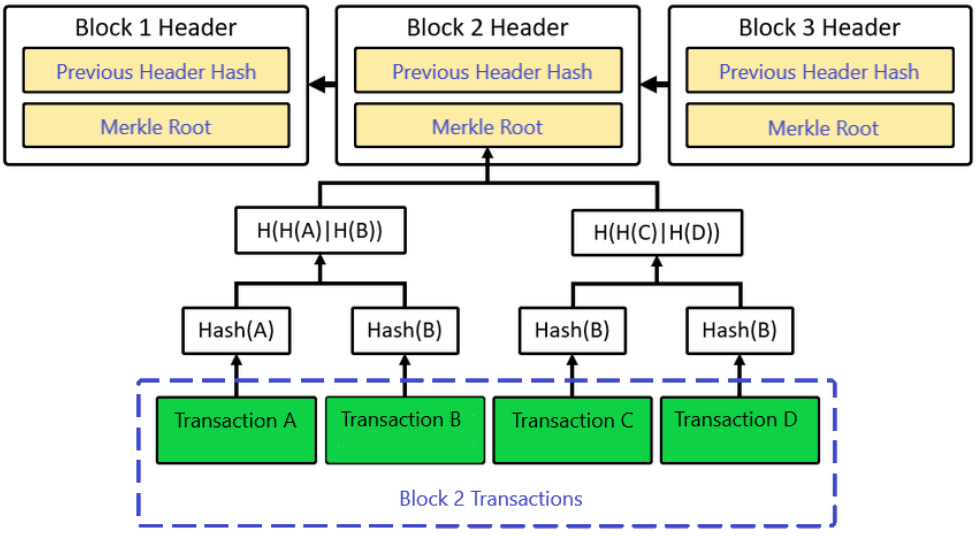

Figure 2.14 Merkle Tree

New blocks, which include new transactions, are appended to the blockchain by a stochastic process called mining. The mining process commonly depends on Proof of Work (PoW) [87]

or Proof of Stake (PoS) [88] methods. The PoW method is a consensus mechanism that requires participants to make an effort to resolve a random puzzle to ensure that nobody deceives the system. However, PoW methods require a significant amount of computational power, and as such they consume a considerable amount of energy. Since the cost of this mechanism increases proportionally to the number of transactions, the PoS system is proposed whereby a participant is chosen to add the latest transaction stack to the blockchain and they receive an amount of crypto money as a reward in return. The selection of participants is conducted based on a level of probability, where a higher stake increases the chance of a participant being selected [88].

A decentralized blockchain network is comprised of nodes that represent mining entities such as servers or computers. They are responsible for verifying and maintaining the public ledger of transactions on a blockchain network. Typically, there are three kinds of nodes in a network, each with varying levels of responsibility: light nodes, full nodes, and archive nodes. A light node is the most basic type of computing device used to support a blockchain network. In PoW systems, miners are the light nodes; whereas in PoS systems, staking wallets are the light nodes. A light node maintains the header chain, requests the remainder of the data, and verifies data validity against the state roots in the block headers. A full node has the responsibilities of a light node, and additionally, it keeps the full blockchain data. Archive nodes store everything kept in the full node and build a historical state archive [89].

2.2.1. Consensus Algorithms

Consensus algorithms play a pivotal role in distributed systems by ensuring agreement among nodes in a network on a single, consistent state of the system. These algorithms are particularly crucial in blockchain networks, where multiple nodes need to agree on the validity of transactions and the state of the ledger. In this section, we explore two prominent consensus algorithms: Proof of Work (PoW) [87] and Proof of Stake (PoS) [88].

Proof of Work (PoW) [87]

Proof of Work [87] is one of the earliest and most widely known consensus algorithms. It operates on the principle of requiring participants, often referred to as miners, to solve computationally intensive mathematical puzzles in order to validate and add new blocks to the blockchain. The first miner to successfully solve the puzzle is granted the right to add the next block and is rewarded with newly minted cryptocurrency.

Mechanism Puzzle Solving: Miners compete to find a nonce that, when combined with the block's data, produces a hash value below a certain target threshold.

Difficulty Adjustment: The network adjusts the difficulty of the puzzle periodically to maintain a consistent block generation time.

Consensus: Once a miner finds a valid solution, they broadcast it to the network. Other nodes verify the solution, and if accepted, the block is added to the blockchain.

Advantages Security: PoW is highly secure due to the computational effort required to solve puzzles.

Decentralization: PoW networks are often characterized by a decentralized distribution of mining power.

Challenges Energy Consumption: PoW has been criticized for its high energy consumption, as miners compete by performing vast amounts of computational work.

51% Attacks: In theory, a malicious actor with more than 51% of the network's mining power could manipulate the blockchain.

Proof of Stake (PoS) [88]

Proof of Stake [88] is an alternative consensus algorithm designed to address some of the drawbacks of PoW, particularly its energy inefficiency. In PoS, validators are chosen to create new blocks based on the amount of cryptocurrency they hold and are willing to "stake" as collateral.

Mechanism Staking: Participants lock up a certain amount of cryptocurrency as collateral to become eligible for block validation.

Block Creation: Validators are chosen to create new blocks based on factors such as the amount of cryptocurrency staked, time held, or a combination of these.

Consensus: Validators propose and validate new blocks, and the consensus is achieved based on the weight of their stakes.

Advantages Energy Efficiency: PoS is often considered more environmentally friendly compared to PoW, as it does not rely on computational puzzles.

Security: PoS can be secure, especially with mechanisms in place to penalize malicious validators.

Challenges Initial Distribution: Critics argue that PoS systems may favor participants with more significant amounts of cryptocurrency, potentially leading to centralization.

Nothing-at-Stake Problem: Validators may have an incentive to support multiple, potentially conflicting, blockchain histories.

In conclusion, both PoW and PoS have their strengths and weaknesses, and the choice between them often depends on the specific goals and requirements of a blockchain network. As the field continues to evolve, hybrid consensus mechanisms and new alternatives are also emerging to address the limitations of existing approaches.

2.2.2. Smart Contracts

Smart contracts (SCs) are computer programs that can reliably and consistently perform transactions and agreements between anonymous parties. They can trigger subsequent actions in a workflow when certain conditions are met without a central authority, legal system, or external enforcement mechanism.

The most popular blockchain utilizing SCs is Ethereum. A SC can be deployed on a blockchain with a gas fee determined due to its size. The gas fee is a fee paid to miners to remunerate the computational power required to process and validate transactions. The immutability of blockchain secures smart contracts from tampering so that, once a SC is deployed, it cannot be modified. Users interact with a SC via transactions with the compiled code of a smart contract through the contract's address. Such transactions might change the state of the contract and receive/send coins from/to another account. Also, a SC can invoke other SCs.

Ethereum tokens are defined on top of the Ethereum blockchain as digital assets. One of the most known Ethereum tokens is named ERC-20 [68], published as a technical standard for smart contracts on the Ethereum blockchain. In our design, we created a stake-reward-punishment mechanism based on ERC-20.

2.2.3. Databases vs Blockchain

Before integrating an application to a blockchain, it is important to determine whether a blockchain is necessary, and which type of blockchain to choose. Since, the conventional PKI approaches are based on database technologies, in this section, we compare blockchain with such traditional approaches in in terms of different requirements of PKI [90]:

Authority: Blockchain is mainly decentralized, but private blockchains may also benefit from centralized approaches. Even though some kinds of distributed database technologies exist, databases are generally managed by an administrator and primarily designed as centralized.

Transparency: While blockchain provides transparency, only the administrators decide who can access data to in databases. Hence, communities seek for transparent architectures like Certificate Transparency (CT) lately. However, such solutions are susceptible to attacks such as split-word attacks.

Architecture: While databases depend on the client-server architecture, blockchain utilizes distributed ledgers and every node participates in the blockchain. Distributed ledger architecture empowers PKI to work independently and removes any need for centralized control. Hence, distributed ledger networks are more advantageous from the privacy point of view.

Data Handling: Blockchain supports only read-write operations. On the other hand, databases support create, update, delete and read-write operations. In PKI, TSPs must guarantee that some of the data such as the revocation list will not be deleted. Moreover, blockchain can provide data persistency by storing copies of data in full nodes.

Integrity: Merkle tree is the primary component of blockchain that utilizes cryptographic hash functions. Every block stores data in this tree, consisting of the digest value of child nodes and the parent node's header consisting of a combination of children's hashes. This process continues up to the final node, which is named as the root node. The root node provides a hash value for the whole tree, and provides data integrity. On the other hand, malicious actors can alter data stored in databases. Even though databases have some security measures such as audit logs, they are still prone to modification. TSP, which operates PKI, must preserve and guarantee data integration of certificates and certificate revocation lists. Blockchain achieves that and provides integrity due to its very nature, when an absolute trust in TSP exists.

Cost: While blockchain is a costly approach due to being hard to implement, databases have been using for a long time and easy to implement.

Performance: Blockchain has verification and consensus methods. Nodes actively participate in the process, and the mining process takes time. From that perspective, databases are faster than blockchain.

Table 2.1 Comparison of Databases and Blockchain in PKI

	Database	**Blockchain**
Authority	Centralized	Decentralized
Transparency	Admin-controlled access	Transparency
Architecture	Client-server	Distributed ledger
Data Handling	CRUD operations	Read-write operations
Integrity	Susceptible to modification	Ensured through Merkle trees
Cost	Cost-effective	Costly implementation
Performance	Faster	Slower

2.2.4. Blockchain and PKI: An Integrative Approach

Blockchain technology and Public Key Infrastructure (PKI) are two separate but related technologies, each designed to promote secure digital transactions. While PKI enables the secure transmission of data over potentially unsecure networks, blockchain provides a mechanism for recording transactions in a secure and verifiable way.

Before examining the relationship between blockchain and PKI, it's important to understand what each technology involves:

PKI encompasses a range of roles, policies, and procedures that are essential for the creation, management, distribution, utilization, storage, and revocation of digital certificates. It is employed to safeguard the transmission of data across networks by utilizing two cryptographic keys: a public key and a private key. This technology forms the foundation of numerous secure communication protocols utilized on the internet today.

Blockchain is a type of distributed ledger that maintains a continually growing list of records, called blocks, which are linked and secured using cryptographic principles. Each block typically includes a cryptographic hash of the previous block, a timestamp, and information about the transactions that occurred. One of the most significant characteristics of blockchain is its decentralization, which implies that no single entity holds authority over the entire chain.

The relationship between blockchain and PKI arises from their shared objective of creating trust in digital transactions. In the context of blockchain, this is often about ensuring that all participants in the network agree on the history of transactions. In the context of PKI, it's about ensuring that the parties involved in a digital transaction are who they say they are.

Blockchain can be seen as a form of PKI. It uses public key cryptography to validate transactions and create new blocks. A user's public key serves as their address in the blockchain, while their private key is used to sign transactions. Furthermore, blockchain technology can be used to enhance traditional PKI systems in several ways:

Decentralization: One of the main weaknesses of traditional PKI systems is the reliance on centralized Certificate Authorities (CAs). These entities are targets for hackers and their compromise can have serious implications for the security of the PKI system. By utilizing a decentralized model, blockchain can mitigate this risk, as there is no single point of failure.

Transparency and Auditability: All transactions on a blockchain are visible to all participants and cannot be altered, which promotes transparency and accountability.

Revocation and Expiry: In a traditional PKI system, revocation and expiry of certificates can be challenging to manage. A blockchain-based system could provide a more efficient and reliable way of handling certificate revocation and expiration.

Automation: Blockchain smart contracts could potentially automate the issuance and revocation of certificates, reducing the need for manual intervention and making the process more efficient.

Cost-effectiveness: Blockchain could potentially offer a more cost-effective way of managing a PKI system, by reducing the need for centralized authorities and manual processes.

When blockchain technology and PKI are combined, we get a hybrid model that significantly elevates the security landscape by incorporating the strengths of both systems and alleviating their individual weaknesses.

Reinventing Trust Management: At its core, blockchain is a trustless system that operates without the need for a central authority. This facet is revolutionary, particularly in PKI, where trust is of paramount importance but its management can often be complex.

In the traditional PKI model, trust is centrally managed by Certificate Authorities (CAs). However, these entities have become the weak links in the chain due to the concentration of power and the potential for single points of failure. Several high-profile breaches have occurred when CAs were compromised, leading to the issuance of fraudulent certificates.

With blockchain, trust is not centrally managed, but rather distributed across the nodes participating in the network. This eliminates the risk associated with centralized trust and creates a system that is inherently more resilient and robust against attacks. In a blockchain-based PKI, each certificate can be independently verified by any node in the network, ensuring transparency and preventing unauthorized issuance of certificates.

Improved Certificate Lifecycle Management: Certificate lifecycle management is another area where blockchain can bring significant improvements to traditional PKI systems. The process of issuing, renewing, and revoking certificates can be cumbersome and inefficient in a centralized system. Moreover, the certificate revocation process has been a longstanding issue in traditional PKI, often leading to the continued acceptance of expired or revoked certificates.

Recording certificates on a blockchain allows for more efficient management of their entire lifecycle. Once a certificate is added to the blockchain, its status can be updated (e.g., issued, renewed, or revoked) and these updates are instantly visible to all participants. This ensures that all nodes in the network have an up-to-date view of the status of all certificates, enhancing the effectiveness of certificate revocation.

Enduring Auditability and Non-Repudiation: Blockchain's immutable nature further benefits PKI systems by providing enduring auditability. Adding each certificate to the blockchain creates a permanent and immutable record, which is highly valuable for audit and compliance purposes.

Furthermore, blockchain's immutability supports non-repudiation, a critical element in digital transactions. Non-repudiation ensures a party involved in a transaction cannot deny their actions. With each transaction being permanently recorded on the blockchain, the evidence supporting non-repudiation becomes indisputable.

Challenges and Future Directions: Despite the compelling advantages, the integration of blockchain with PKI is not without challenges. Blockchain's public nature can potentially conflict with privacy requirements, especially with regulations like GDPR in the European Union. Scalability is another concern as the number of certificates could become so large that it becomes impractical to store and manage them on a blockchain.

The convergence of blockchain and PKI opens new avenues for research and innovation. A hybrid model could introduce novel ways to establish trust, manage identities, and secure digital transactions. As we look forward to the future, integrating blockchain with PKI could potentially reshape the landscape of digital security, heralding a new era of decentralized trust and security.

In conclusion, the integration of blockchain and PKI offers the potential for a more secure, efficient, and accountable system for managing digital transactions. However, challenges remain, including the need to ensure the scalability and confidentiality of blockchain-PKI systems, and to address regulatory and legal considerations. As the fields of blockchain and PKI continue to evolve, further research and development will be needed to realize their full potential.

3. RELATED WORK

In the literature, it is seen that blockchain has been widely used in different fields recently, and remarkable results have been obtained [91–94]. However, in the field of PKI, which has an important place in the digital world, it is seen that blockchain studies are few, and new studies are needed. These studies and the findings obtained from these studies are given in detail in the subsequent sections.

3.1. A Blockchain-based PKI Management Framework [1]

Yakubov et al. [1] developed a new blockchain based PKI which allows issuance, validation, and revocation of digital certificates. In their design, each Certificate Authority (CA) has an associated smart contract (SC) that contains the CA certificate, a digest of all certificates issued by that CA, and their revocation statuses. They introduced a new X.509 extension, incorporating the SC address into the certificate to establish a connection between certificates and SCs. However, the study lacked a supervisory mechanism for SCs, allowing attackers to potentially deploy SCs on the blockchain to issue root certificates and create fraudulent end-user certificates.

3.2. Certledger: A New PKI Model with Certificate Transparency based on Blockchain [2]

M. Kubilay et al. [2] propose a Public Key Infrastructure (PKI) architecture referred to as CertLedger. This structure is designed to validate, keep, and revoke SSL certificate at the same time as managing relied on certificate Authority (CA) certificate inside an open blockchain. The method targets to enhance transparency within the certificate issuance and revocation procedure, thereby preventing man-in-the-center attacks.

Key features of CertLedger:

Transparency: CertLedger enhances transparency in the certificate lifecycle, addressing potential MITM attacks by providing a more transparent process for certificate issuance and revocation.

Blockchain Storage: CertLedger stores the entire X.509 certificate within the blockchain transaction. This approach ensures that the entire certificate, including its ASN.1 encoded structure, is recorded on the blockchain.

However, there are certain limitations and considerations associated with CertLedger:

Costly Storage and Decoding: Storing and decoding an entire X.509 certificate on a smart contract can be a costly process due to the complex ASN.1 encoded structure and file size of X.509 certificates.

Limited Applicability: The CertLedger mechanism is specifically designed for SSL certificates and website URLs. As a result, it may not be applicable to other types of certificates and might not be easily adaptable to new projects with different certificate requirements.

In summary, while CertLedger provides transparency and security benefits in SSL certificate management, the potential challenges of storing and decoding entire X.509 certificates, as well as its limited applicability to specific use cases, should be considered in evaluating its suitability for broader and more diverse certificate management scenarios.

3.3. SCPKI: a Smart Contract-based PKI and Identity System [3]

Basssam et al. [3] introduces a smart contract-based Public Key Infrastructure (PKI) and identity system known as SCPKI, which operates on a Web of Trust model. The system utilizes smart contracts to enhance security and ensure the detection of fraudulent attempts. In the defined trust model, participants have the ability to add their attributes to the blockchain, and other participants can acknowledge trust in them by sending new transactions.

Web of Trust Model: SCPKI operates based on a Web of Trust model, allowing participants to add their attributes to the blockchain. Other participants can then acknowledge trust in them through new transactions.

Smart Contracts for Security: The use of smart contracts enhances security and fraud detection within the PKI and identity system.

However, there are certain challenges and considerations associated with SCPKI:

Cost of Transactions: Executing transactions and changing states within smart contracts have associated costs. An incentive mechanism is required to encourage participants to contribute to the system, as the cost may act as a barrier.

Susceptibility to Attacks: SCPKI is noted to be susceptible to attacks due to the lack of a punishment mechanism. Without a means to penalize malicious or fraudulent behavior, the system may be vulnerable to exploitation.

In summary, SCPKI introduces a novel approach to PKI and identity systems by leveraging smart contracts and a Web of Trust model. However, addressing challenges related to the cost of transactions and susceptibility to attacks will be crucial for ensuring the robustness and effectiveness of the proposed system.

3.4. ETHERST: Ethereum-based Public Key Infrastructure Identity Management with a Reward-and-punishment Mechanism [4]

Chong [4] et al. presents an Ethereum-based PKI Identity management system named ETHERST, which is built on the Web of Trust model similar to SCPKI. ETHERST incorporates a new stake-reward-punishment mechanism to enhance its functionality. However, certain aspects and potential threats have been identified:

Web of Trust Model: ETHERST utilizes a Web of Trust model, akin to SCPKI, allowing participants to establish trust through interactions recorded on the blockchain.

Stake-Reward-Punishment Mechanism: ETHERST introduces a new stake-reward-punishment mechanism, which likely involves participants staking cryptocurrency to vouch for their trustworthiness.

Insufficient Discussion of Threats: The paper highlights that threats against the validity of the design are not sufficiently discussed, particularly in scenarios where attackers may dominate the voting process. A comprehensive analysis of potential threats is crucial for evaluating the robustness of the system.

Lack of Fraud Reporting Mechanism: ETHERST is noted to lack a fraud reporting mechanism for recovery. In the absence of a mechanism for reporting and addressing fraudulent activities, the system may face challenges in dealing with malicious actors.

In summary, while ETHERST builds upon the Web of Trust model and introduces a stake-reward-punishment mechanism, the identified concerns regarding the discussion of threats and the absence of a fraud reporting mechanism highlight areas that need further attention. Addressing these concerns is essential for ensuring the effectiveness and security of the proposed PKI Identity management system.

3.5. Blockchain-based Public Key Infrastructure Certificate Management [5]

Garba et al. propose a blockchain-based Public Key Infrastructure (BB-PKI) model aimed at preventing impersonation attacks arising from compromised Registration Authorities (RAs) [5]. The proposed model aims to enhance transparency in the registration process by involving RAs as intermediary nodes between users and Certificate Authorities (CAs) for reviewing and approving certificate requests. Key points in the BB-PKI model include:

Transparent Registration Process: The BB-PKI model incorporates RAs as intermediaries in the registration process, providing a transparent mechanism for reviewing and approving certificate requests.

Blockchain Maintenance Manager (BMM): The introduction of a Blockchain Maintenance Manager (BMM) is designed to oversee the activities of RAs and trusted CAs within the blockchain.

However, there are certain limitations and considerations associated with BB-PKI:

Lack of Trusted List Management: The model lacks a mechanism to manage trusted lists, which are essential for maintaining and updating trusted entities within the PKI system. A comprehensive method for managing and updating trusted lists is crucial for system integrity.

Undefined CA Inclusion and Exclusion Processes: BB-PKI does not define processes for the inclusion and exclusion of CAs. The absence of clear procedures for adding or removing CAs can pose challenges in terms of governance and overall system management.

In summary, while BB-PKI addresses certain security concerns related to compromised RAs in the PKI ecosystem, the lack of mechanisms for managing trusted lists and undefined processes for CA inclusion and exclusion are areas that need further consideration for the completeness and effectiveness of the proposed model.

3.6. An Efficient PKI Architecture based on PBFT Through Dynamic Threshold Signatures [6]

Kubilay at al. propose, KORGAN, detailed in [6], leverages a permissioned blockchain with a Practical Byzantine Fault Tolerance (PBFT) consensus mechanism. The study primarily aims to enhance certificate validation during TLS negotiation. A key innovation is that end-users do not need to join the blockchain network. Instead, they only need access to public keys to validate the dynamic threshold signatures of the blockchain's blocks.

Permissioned Blockchain: KORGAN operates on a permissioned blockchain, which typically restricts participation to a known set of nodes or participants.

The consensus mechanism used in KORGAN is PBFT. PBFT is designed to handle Byzantine faults in a distributed system, providing a high level of fault tolerance.

Certificate Validation Optimization: The primary focus of KORGAN is on optimizing the process of certificate validation during TLS negotiation.

End-Users Involvement: One notable feature is that end-users are not required to join the blockchain network. They only need access to public keys for the validation of dynamic threshold signatures.

It's important to note that while the study emphasizes optimizing certificate validation during TLS negotiation, the broader implications of KORGAN in terms of blockchain-based certificate management and the specifics of its permissioned blockchain model are not detailed in the provided summary.

In summary, KORGAN presents a system with a permissioned blockchain and PBFT consensus, focusing on efficient certificate validation during TLS negotiation and allowing end-users to participate without joining the blockchain network directly.

3.7. A Semidecentralized PKI System based on Public Blockchains with Automatic Indemnification Mechanism [7]

A semi-decentralized Public Key Infrastructure (PKI) system based on public blockchains, aiming to prevent single points of failure is introduced in [7]. The proposed architecture defines four key actors: web owner, Certificate Authority (CA), web user, and smart contract (SC). However, certain limitations and potential issues have been identified:

Semi-Decentralized PKI: The architecture is designed to be semi-decentralized, leveraging public blockchains to prevent single points of failure.

Actors Defined: The system defines four main actors—web owner, CA, web user, and smart contract (SC)—to play specific roles in the PKI system.

Optimal Use of TP-Merkle Trees: The proposed system optimally utilizes TP-Merkle trees for its operation.

Lack of Supervisory Mechanism: The design of the system lacks a supervisory mechanism. The absence of oversight can create vulnerabilities, and the paper does not address potential solutions for monitoring and supervising the system's operation.

Assumption of CA Deployment of Smart Contracts (SCs): The proposed system assumes that all CAs will deploy smart contracts themselves. This assumption introduces a potential vulnerability, as an attacker could create a fake CA smart contract to issue fraudulent end-user certificates. In the event of a man-in-the-middle attack, such fake certificates may not be easily detectable.

Focus on SSL Certificates: The design of the system is tailored specifically for SSL certificates. It does not provide a solution for other types of certificates, limiting its applicability to a broader range of use cases.

In summary, while the semi-decentralized PKI system proposed in [7] leverages public blockchains and TP-Merkle trees, it faces challenges related to the lack of a supervisory mechanism, potential vulnerabilities in the assumption of CA deployment of smart contracts, and a limited scope focusing on SSL certificates. These aspects would need careful consideration and potential enhancements for a more comprehensive and secure PKI solution.

3.8. SABRES - a Proof of Concept for Enhanced Cloud Qualified Electronic Signatures [8]

A proof of concept for enhanced cloud qualified electronic signatures (SABRES) is proposed in [8], aiming to improve trust in the creation of Qualified Electronic Signatures in cloud environments. This approach introduces a new blockchain solution for logging and verifying access to private keys. However, there are certain limitations and considerations to be aware of:

Enhanced Cloud Qualified Electronic Signatures: SABRES is designed to enhance trust in the creation process of Qualified Electronic Signatures in cloud signing, utilizing a blockchain solution for logging and verifying accesses to private keys.

SSL-Specific Focus: The proposal emphasizes that existing Public Key Infrastructure (PKI) solutions are SSL-specific, suggesting a broader application for Qualified Electronic Signatures beyond SSL.

Limited Certificate Coverage: SABRES does not suggest a design covering all types of certificates. The focus is primarily on Qualified Electronic Signatures and their creation in the cloud, potentially limiting its applicability to a specific use case.

Focus on Cryptographic Signatures in the Cloud: The main focus of SABRES is on creating cryptographic signatures in the cloud. It does not address the creation details of Advanced Electronic Signatures (AdES), which are essential for certain applications and compliance requirements.

Web-Based Signature Application: SABRES encourages signers to use a web-based application for signatures. This may limit its integration with potential new applications that may seek to interact with the smart contract in different ways.

Interoperability Challenges: The absence of advanced electronic signature operations within the smart contract may lead to potential applications creating non-compliant AdES. Interoperability could be a challenge as a result.

Lack of Solution for Local Signing, Validation, and Preservation: SABRES does not provide a solution for local signing, signature validation, and preservation operations, which are critical components of a comprehensive electronic signature system.

In summary, while SABRES presents a proof of concept for enhancing trust in cloud-based Qualified Electronic Signatures, it has limitations such as a specific focus on cryptographic signatures in the cloud, potential interoperability challenges, and the lack of coverage for various types of certificates and signature operations. These considerations are essential for evaluating the broader applicability and compliance of the proposed solution.

3.9. AKI: Accountability in Key Infrastructure [9]

Kim et al. propose AKI [9] (Accountability in Key Infrastructure), a novel Public Key Infrastructure (PKI) architecture designed to mitigate the concentration of trust within Certificate Authorities (CAs). Unlike traditional PKIs where CAs wield unilateral authority, AKI decentralizes accountability by involving multiple entities across all defined operations. This includes the introduction of new roles such as Certification Agency (CA), Integrity Log Server (ILS) Operators (ILSO), and Validators.

AKI operates under the assumption that its trusted entities (CAs, ILSs, Validators) do not collude, a premise that could be exploited by a determined adversary. Vulnerabilities in AKI include scenarios where a compromised CA and ILS could collaborate to issue fraudulent certificates, facilitating a potential split-world attack. Notably, AKI lacks mechanisms to detect or mitigate such attacks. Furthermore, AKI is susceptible to misuse where an adversary possessing a compromised domain private key could illicitly request certificate revocations without additional verification.

3.10. ARPKI (Advanced Accountability in Public Key Infrastructure) [10]

Basin et al. propose ARPKI [10] (Advanced Accountability in Public Key Infrastructure) as an advancement of AKI, aimed at enhancing security against adversaries capable of compromising up to $(n-1)$ trusted entities. In ARPKI, the issuance of an ARPKI certificate, called ARCert, necessitates the involvement of at least two Certificate Authorities (CAs) and one Integrity Logging Service (ILS). It is important to note that ARPKI remains susceptible to a split-world attack if the collaborating entities involved in issuing an ARPKI certificate collude. Similar to AKI, ARPKI lacks a mechanism to detect such attacks. Furthermore, the dependency on an ILS for synchronization across other ILSs introduces a potential single point of failure within the ARPKI system.

3.11. NameCoin [11]

NameCoin [11], a fork of Bitcoin, serves as a decentralized Domain Name System (DNS) for addresses ending in ".bit". Its primary purpose is to provide a decentralized and censorship-resistant alternative to traditional DNS. In NameCoin, a unique feature allows for the addition of a self-signed Transport Layer Security (TLS) certificate to DNS addresses as supplementary information.

This means that, alongside the conventional DNS functionality, NameCoin enables domain owners to attach their own TLS certificates to their domains. During the TLS handshake process, TLS clients can then authenticate the domain using this self-signed certificate. This approach enhances the security of connections to "bit" addresses by allowing clients to verify the authenticity of the domain through the attached TLS certificate, even in a decentralized and blockchain-based system like NameCoin.

3.12. PB-PKI (Privacy-By-Policy Public Key Infrastructure) [12]

PB-PKI [12] (Privacy-By-Policy Public Key Infrastructure) is an adaptation of CertCoin with a focus on privacy. The primary distinction lies in avoiding the direct linking of identity with the associated public key. This approach aims to prevent the tracking of identities' actions based on their use of public keys. Such privacy-conscious features are particularly relevant in use cases like ubiquitous computing, the Internet of Things (IoT), and vehicular networks.

While the privacy-aware design of PB-PKI is compelling for specific scenarios, it poses challenges when applied to identity management of domains and their certificate lifecycle. This is primarily due to a conflict with the transparency requirement. In the realm of domain identity management, transparency is often crucial for ensuring accountability and trust. The contradiction arises because the separation of identity and public key, while enhancing privacy, may hinder the transparency needed for effective domain and certificate management.

While PB-PKI offers valuable privacy enhancements, its trade-off with transparency makes it less suitable for certain applications where visibility into the connection between identity and public key is essential, such as in domain and certificate management.

3.13. CertChain: Enhancing Certificate Audit on TLS Connections through Blockchain [13]

CertChain, as proposed in [13], introduces an architecture for public auditing of TLS connections using blockchain technology. The framework innovates with a novel data structure named CertOper, integrated into blockchain transactions to manage certificate-related operations including registration, update, and revocation. CertChain also implements a dual counting bloom filter to enhance the accuracy of revocation checks.

3.13.1. Key Aspects of CertChain

- **CertOper Data Structure:** CertChain integrates CertOper into blockchain transactions, enabling functionalities such as certificate registration, updates, and revocations.
- **Dual Counting Bloom Filter:** To improve security by reducing false positives in revocation checks, CertChain utilizes a dual counting bloom filter.
- **Consensus Protocol:** CertChain adopts a consensus protocol inspired by Ouroboros, where the dependability rank of a Certificate Authority (CA) or bookkeeper determines leader selection.

Challenges and Vulnerabilities

Despite its innovative features, CertChain faces several challenges:

- **MITM-like Attacks:** CertChain is susceptible to Man-in-the-Middle (MITM)-like attacks in scenarios involving dishonest bookkeepers. Although bookkeepers may

maintain the blockchain honestly, they can provide incorrect responses to TLS clients during certificate validation. This could lead clients to accept invalid certificates, exposing connections to potential MITM attacks.

- **Privacy Concerns:** CertChain does not adequately address the privacy of TLS clients. Bookkeepers have the capability to track clients' visiting histories, posing privacy risks in environments where confidentiality is crucial.

CertChain introduces significant advancements in the auditing and management of TLS certificates using blockchain, its susceptibility to MITM-like attacks and privacy vulnerabilities necessitates further enhancements to ensure robust security and privacy protections.

3.14. LRS PKI: A Novel Blockchain-based PKI Framework using Linkable Ring Signatures [14]

Liang et al. propse LRS PKI to address the challenges inherent in the Certificate Authority (CA) of the Public Key Infrastructure (PKI) system, an innovative blockchain-based PKI solution called LRS PKI is proposed in Computer Networks [14]. This system leverages linkable ring signatures to enhance security. By utilizing blockchain technology, it ensures meticulous recording of certificate operations, while the InterPlanetary File System (IPFS) is utilized for certificate storage, thereby separating the blockchain layer from the storage layer effectively.

In enhancing the privacy of the issuing CA and preventing potential targeted attacks, the authors introduce the novel concept of a Ring CA. This innovative approach protects the confidentiality of the issuing CA for specific certificates, thereby thwarting attackers from exploiting vulnerabilities.

Furthermore, to ensure the integrity of CAs and prevent malicious activities, their methodology integrates a verification step during certificate validation. This entails

scrutinizing the consistency of the issuing CA, thereby elevating the overall security posture of LRS PKI.

The security analysis and experimental findings affirm the resilience, efficiency, and practicality of LRS PKI. In future endeavors, they aim to conduct comprehensive applicability experiments on a larger scale, deploying LRS PKI across multiple nodes in diverse regions. This will offer a more precise evaluation of its performance in real-world scenarios [14].

3.15. PoliCert: Secure and Flexible TLS Certificate Management [15]

Szalachowski et al. [15] proposes Policert which operates as a public log-based program that facilitates the management, issuance, and enforcement of certificate policies. Multisignature certificates and subject certificate policies are logged on a public log server. However, this approach lacks established mechanisms to detect and control errant log behavior.

3.16. Khan et al. [16]

Khan et al. [16] proposes Accountable and Transparent TLS Certificate Management which explore two different attacks for Policert and eliminates these attacks by introducing an improved revocation system and monitoring mechanism. Khan et al.

Khan et al. [95] also proposes a secure and accountable TLS certificate management (SCM). In SCM, CA-signed domain certificates are stored in log servers which is conducted on the blockchain platform. Moreover SCM decreases the storage cost of blockchain dramatically.

Khan et al. [96] also propose a log-based PKI called as Attack-Resilient TLS Certificate Transparency. ARCT eliminates impersonation attacks on registration process of certificate-issuance by collaborative certificate-issuance mechanism. In addition, it provides an revocation mechanism. However they don't provide a audit or issuance mechanisms for CA certificates.

3.17. A Multi-Party Contract Signing Solution Based on Blockchain [17]

Josep-Lluis Ferrer-Gomila et al. introduce a blockchain-based protocol [17] designed for multi-party contract signing, particularly significant in electronic commerce and dynamic data marketplaces. Unlike conventional approaches, this protocol eliminates the necessity for a trusted third party (TTP) by ensuring fairness in contract signing—each party either receives evidence of the signing or none at all, thereby preventing discrepancies among honest signatories. The protocol meets essential requirements such as fairness, timeliness, non-repudiation of origin and receipt, and also supports confidentiality. To mitigate blockchain-related costs, its usage is restricted to exceptional cases and is initiated by only one party involved, demonstrating manageable financial implications.

3.18. Elliptic Curve Threshold Signature Scheme for Blockchain [18]

Huifang Yu et al. [18] proposed an Elliptic Curve Threshold Signature Scheme (ECTSS) for Blockchain aimed at reducing gas consumption caused by transmitting off-chain data, which can lead to network congestion. They introduce a blockchain-specific variant known as BC-ECTSS to address multiple approval issues. Through a rigorous analysis in the random oracle model, they illustrate that BC-ECTSS achieves existential unforgeability against adaptive chosen-message attacks, ensures anonymity, and offers low computational costs.

3.19. General Discussion

To sum up, in the realm of blockchain-based Public Key Infrastructure (PKI), diverse approaches have been explored to enhance certificate management and security. Yakubov et al [1]. present a framework leveraging smart contracts for individual Certificate Authorities (CAs), introducing a new X.509 extension for improved linkage between certificates and smart contracts. While innovative, the absence of a supervisory system for smart contracts

raises security concerns. M. Kubilay et al. contribute CertLedger [2], emphasizing transparency in certificate issuance and storing entire X.509 certificates in blockchain transactions. However, challenges arise in terms of costly storage and limited applicability.

Example of a decentralized approach, Bassam et al. propose SCPKI [3], a system operating on a Web of Trust model via smart contracts, addressing security but facing challenges in transaction costs and susceptibility to attacks. Chong introduces ETHERST [4], an Ethereum-based PKI with a Web of Trust model and a stake-reward-punishment mechanism, yet concerns linger about threat discussions and fraud reporting.

Garba's BB-PKI [5] model aims to prevent impersonation attacks, emphasizing transparent registration processes, though lacking trusted list management. KORGAN [6] focuses on efficient certificate validation during TLS negotiation using a permissioned blockchain with PBFT. Gwan [7] introduces a semi-decentralized PKI system based on public blockchains, emphasizing the prevention of single points of failure, but concerns persist regarding the absence of a supervisory mechanism. SABRES targets enhanced cloud Qualified Electronic Signatures but exhibits limitations in certificate coverage and a focus on cryptographic signatures. AKI [9] proposes accountability in PKI by involving multiple entities in operations, introducing Certification Agency (CA), Integrity Log Server (ILS) Operators, and Validators. Despite its innovative approach, challenges related to compromise risks and certificate revocation weaknesses have been identified.

Our proposed approach, SemiDec-PKI contributes a novel and semi-decentralized approach to Public Key Infrastructure (PKI) leveraging blockchain technology. One of its key innovations is the support for any type of certificate, replacing the traditional X.509 certificate through a redefinition within Smart Contracts (SCs). This redefinition significantly reduces the complexity associated with storing, parsing, and decoding X.509 certificates, enhancing the overall efficiency of the PKI system.

Moreover, SemiDec-PKI addresses critical security concerns by incorporating mechanisms to detect misissuance and eliminate single point of failure for issuing any level of certificates. It introduces a robust reward-punishment mechanism, utilizing Ethereum Request for

Comments (ERC20) tokens. This mechanism plays a crucial role in preventing compromise of Trusted Service Providers (TSPs) and supervisory bodies, thus ensuring the integrity of the PKI system.

Last but not the least, the adaptability of SemiDec-PKI is a notable feature, as it can be seamlessly extended to accommodate new services for different types of certificates. Specifically, it opens avenues for providing qualified validation and preservation services for electronic signatures and seals, demonstrating its flexibility in meeting evolving requirements.

4. SEMIDEC-PKI

In today's interconnected and digitalized world, ensuring secure communication and protecting sensitive information has become a critical priority. Public Key Infrastructure (PKI) serves as a fundamental framework that guarantees the integrity, confidentiality, and authenticity of electronic communications and transactions. By harnessing the power of asymmetric encryption and digital certificates, PKI establishes an infrastructure for building trust, verifying identities, and securing sensitive data across diverse networks. Certificate Authorities (CAs) play a pivotal role in the PKI ecosystem by issuing and managing digital certificates. These trusted third-party entities verify the identity of certificate subjects and digitally sign their certificates, vouching for their authenticity.

4.1. PKI ISSUES

While CAs play a vital role in the system, they expose PKI to potential vulnerabilities and incidents where faulty certificates might be issued. A number of high-profile incidents have highlighted the risks associated with faulty certificate issuance [97]. For example, Comodo and DigiNotar were compromised, and attackers managed to issue fraudulent certificates for popular websites, including Google, Yahoo, and Skype [33][98]. Symantec mistakenly issued certificates that included a domain name, violating CA industry policies and procedures in 2015 [34]. Furthermore, CAs (e.g., Lets Encrypt) are taken down by research in practice to show their weak security [97, 99, 100]. A rigorous vetting process and adherence to industry standards by CAs are crucial to preventing such unauthorized certificate issuances [100].

A rigorous vetting process as well as adherence to industry standards by CAs are crucial to preventing such unauthorized certificate issuances. As shown in the incidents above, CAs constitute a *Single Point of Failure (SPoF)* in the certificate issuance mechanism. Therefore, CAs should provide guarantee regarding the accuracy of the information included in the certificate or the integrity of their infrastructure. CAs offers financial compensation or

remedies in situations where the CA's negligence or failure to meet industry standards has caused harm to a subject/entity. As part of the current PKI model, *punishment issues* are handled by insurance companies, and entities must prove that they have been victimized in court.

Another issue is the *management of trust lists*. In PKI, a certificate chain is a list of certificates that usually starts with an end-entity certificate followed by one or more CA certificates. CAs at the top of the hierarchy are called root CAs. They can issue certificates directly for users, or delegate some of the task to intermediate CAs. Root CAs are self-signed and pre-installed in the OSs, browsers, and applications. Besides the trust lists of SSL certificates, applications could have their own trust lists of CAs. For example, Adobe has a trust list for electronic signature [35], Java has a trust list for code signing certificates. Since CA needs to contact every application, browser, and OS, so it is hard to manage trust lists for each software.

In PKI, CAs have a crucial role in tasks like identity validation and application assessment. However, their involvement also introduces the risk of human errors. To enhance security and mitigate potential risks, it becomes essential to implement a crosscheck mechanism. This process should involve personnel from diverse authorities to maximize its effectiveness. Currently, the conventional certificate life-cycle lacks this protective measure. Hence, impersonation attacks that impersonate legitimate entities and fraudulently obtain certificates could occur in this registration step. Therefore, trust and security of the infrastructure could be compromised due to these *certificate registration issues*.

The last issue called *the revocation monopoly issues* is related to certificate revocation, which is the process of declaring a previously issued digital certificate as invalid or no longer trustworthy before its expiration date. Certificate owners or CAs can initiate the revocation process for the certificate to the responsible CA. In an emergency, it can be problematic if the CA cannot be reached immediately, such as during non-working hours.

In order to address these issues, we propose a new semi-decentralized PKI architecture called SemiDec-PKI that supports any type of digital signature. SemiDec-PKI provides

all services required for the management of certificates such as revocation, validation, dissemination, monitoring, and auditing. The proposed approach is based on blockchain due to its suitable characteristics for the problem such as decentralization, immutability, transparency and security. Blockchain is simply described as a distributed, decentralized, and immutable ledger composed of transaction records, and serves as a foundation for various protocols such as Bitcoin, Ethereum. Smart contracts (SCs), which are self-executing programs stored on a blockchain that are executed when certain conditions are met, are another important concept introduced with Ethereum. SCs help to automate the execution of agreements between parties so that all parties can trust the outcome without the need for any third-party involvement or time loss. They also have the capability to automate workflows by initiating a sequence of actions. Our PKI architecture is built upon SCs that effectively manages certificate issuance, revocation, validation, and fraud mechanisms. For instance, the smart contract code meticulously verifies all certificate fields, ensuring the validity of the certificate structure during the certificate issuance step, and all parties could check the status of certificates via SCs in the validation step.

The proposed approach constructs a hierarchical structure comprising a Trust List (TL), which is under the management and governance of Supervisory Bodies (SB). SBs are predefined in smart contracts (SC) and eases the *management of trust lists*. SemiDec-PKI is based on a voting scheme that combines the Web of Trust with a centralized approach in order to prevent *SPoF*. The voters are selected by using a stake reward-punishment mechanism. While SBs vote for CA certificates, CAs vote for end-user certificates. The voting system also increases robustness of the system against impersonation attacks resulting from *certification registration issues*. Moreover, the proposed approach enables users to revoke their certificates independently, eliminating the necessity for a centralized revocation authority. This empowers users with increased autonomy and effectively addresses *revocation monopoly issues*. By using SCs, we increase the security controls in each step in certificate life cycle. Last but not least, we propose a robust commercial model based on a stake reward and punishment mechanism which incentivizes certain behaviors while penalizing undesirable actions. Ethereum Request for Comments 20 (ERC-20) tokens [68]

are digital assets created, managed, and exchanged on the Ethereum blockchain, play a crucial and integral role in this mechanism. CAs and end-users participate by locking up a specific amount of cryptocurrency tokens as collateral, referred to as their stake, which serves as assurance and is subsequently distributed among voters. Potential incidents and fraud complaints are promptly addressed by penalizing the responsible CAs through the loss of their tokens, allowing for a resolution of *punishment issues* without the need for lengthy court procedures.

To sum up, SemiDec-PKI approach presents a decentralized PKI architecture that effectively addresses the challenges inherent in traditional PKI systes. While various other proposals based on blockchain technology are available in the literature [1–7] that aim to improve certificate life-cycle management for CAs and mitigate known security threats, SemiDec-PKI distinguishes itself from other studies by providing a comprehensive solution that extends beyond SSL certificates to encompass any type of certificates. SSL certificates, due to their online and publicly accessible nature, allow for the monitoring of the number of active certificates in use. However, the statistics about qualified certificates for e-signature or code signing, which plays a critical role in various applications, remains relatively scarce a global scale. Only a limited number of countries, such as Turkey, with its 80 million-strong population, have conducted research to compile statistics on these certificates. For example, in Turkey, there are approximately 6.9 million qualified digital certificates and 900,000 mobile signature certificates dedicated to e-signatures [101, 102]. This underscores the urgent need for secure and efficient PKI systems that can extend their benefits to various certificate types. Moreover, while certificates conform to the X509 standard and are encoded with ASN.1 in the traditional architecture and the other proposals, SemiDec-PKI redefines the certificate structure and ensures the integrity and validity of certificates by using SCs and hence prevents the issuance of malformed certificates. Of particular significance, the Semi-Dec PKI stands out by effectively resolving the persistent problem of single points of failure with its pioneering voting scheme. This approach provides robust protection against vulnerabilities linked to certificate issuance, thus fortifying the PKI system's resilience and security.

A well-managed Public Key Infrastructure (PKI) should adhere to various technical and legal documents [103]. The trustworthiness of a Certificate Authority (CA) is determined by its compliance with these requirements. However, the lack of a standardized, globally accepted method for managing certificate issuers can lead to specific challenges and concerns. These issues can be summarized as follows:

A well-managed Public Key Infrastructure (PKI) should adhere to various technical and legal documents [103]. The trustworthiness of a Certificate Authority (CA) is determined by its compliance with these requirements. However, the lack of a standardized, globally accepted method for managing certificate issuers can lead to specific challenges and concerns. The current issues and corresponding solutions of SemiDec-PKI are summarized as follows:

1. Management of trust lists: The European Union (EU) enacted a regulation for the management of trust lists and, the CA/Browser Forum (CA/B Forum) [49] established industry standards and guidelines for the issuance and management of digital certificates in the SSL framework [104, 105]. Some browsers or operating system vendors run certificate inclusion programs to decide on trust lists under the supervision of CA/Browser Forum [106], hence there is no common global trust list for CAs issuing certificates. It is therefore hard to determine which trust list is valid for which certificate type from the application's point of view.

Solution: Semidec-PKI contains a chain of trust within itself. With the trust chain it contains and the ability to verify certificates via smart contracts, applications do not need to manage a separate trusted list. The system allows for expansion by welcoming participants from new countries into the SBs, and it provides a voting process where existing Supervisor Bodies can vote for new SBs.

2. Single Point of Failure (SPoF): Even if applications handle the management of trust lists, CAs which have absolute trust in the infrastructure may still make mistakes, and this may result in the *Single Point of Failures*. Attackers may still issue end-user certificates and exploit the system by compromising CAs [16].

Solution: eIDAS (electronic IDentification, Authentication and trust Services) [80] defines procedures from defining supervisory bodies to the issuance of end-user certificates. A supervisory body audits CAs through accredited conformity assessment bodies (CABs) and reviews related audit reports in order to decide if the CA is eligible to issue certificates. Therefore, one way to manage a worldwide trust list of CAs would be to specify SBs, CABs, and related technical and legislative requirements for CAs to be designated as trusted. The current study, SemiDec-PKI, proposes a system that aligns with the regulatory framework provided by eIDAS while harmonizing centralized and web of trust-based approaches. It achieves this by employing blockchain technology to deliver ease of access and transparent management of a globally trusted list. The system incorporates a voting mechanism to eliminate *Single Point of Failure* in scenarios where a Certificate Authority (CA) may be compromised or deceived.

3. Certificate registration: In conventional operation, certificates are issued only after registration authorities assess end-user application forms. Once approved and issued, an end-user certificate remains valid until revoked or expired. Registration authorities handle identity validation and application assessment, making them susceptible to human errors. Therefore, cross-checks are critical protection mechanisms, especially when involving checkers from different authorities to enhance effectiveness. However, the conventional certificate lifecycle does not currently employ such a mechanism.

Solution: SemiDec-PKI includes more than one CA in the control mechanism and prevents fraud documents and impersonation attacks related to domain ownership by providing cross check with voting dynamics.

4. Revocation monopoly: The lifecycle of a certificate spans from its creation to the end of its validity. In the conventional PKI architecture, CAs are responsible for services in a certificate lifecycle, namely registering, verifying, creating, and validating certificates. CAs can also revoke certificates in the case of an issue being reported. Although certificate owners and CAs can initiate the revocation process for the certificate, the revocation process

itself can only be conducted by CAs. Therefore, if CAs are not reachable outside of normal working hours, this may cause problems in case of emergency.

Solution: SemiDec-PKI resolves this issue by actively involving multiple CAs in the certificate revocation process. Voters in this design consist of Supervisor Bodies and CAs. The revocation of a CA is dependent on the votes of Supervisor Bodies.

5. Punishment: In the context of traditional PKI, Certificate Authorities (CA) engaged in fraudulent activities or failed to meet security standards can face a wide range of consequences. One notable consequence is the potential removal of the CA's root certificate from the trust lists maintained by browsers and operating systems. Additionally, legal actions might ensue against the errant CA, leading to fines or penalties for various infractions, including negligence, breach of contract, and involvement in fraudulent activities. However, it's important to note that these punitive measures occur external to the PKI framework and fall under the jurisdiction of legal authorities. Despite their effectiveness, these actions are not inherently integrated into the PKI system itself. As PKI develops, there is a growing desire to strengthen its capacity for identifying such breaches within its framework. This would lead to quicker detection of misconduct and more effective resolution for those impacted.

Solution: SemiDec-PKI also incorporates a punishment mechanism. In the event of an attempted manipulation by an attacker, SemiDec-PKI responds by imposing appropriate penalties, as determined by the voting results. The attacker faces the risk of losing ERC tokens due to their actions, serving as a powerful deterrent against malicious behavior.

6. Monitoring and logging: Although it is possible to track SSL certificates of end-users created by CAs, this does not apply to other types of certificates. Moreover, both in traditional PKI and the Certificate Transparency Project, *split-world attacks* [2] are an issue resulting from the difference between the validation process of SSL/TLS certificates by web browsers and their logging procedures on servers. In this attack, attackers acquire fraudulent certificates from a Certificate Authority (CA) and manipulate the logging mechanism

Table 4.1 Issues and Corresponding Solutions Proposed by SemiDec-PKI

Issues	**Corresponding Solutions in SemiDec-PKI**
Management of trust lists	trust chain and smart contract-based certificate validation.
Single Point of Failure (SPoF)	voting scheme.
Certificate registration	cross check via multiple CAs'.
Revocation monopoly	revocation that can be done by users in smart contract.
Punishment	disincentive with ERC Tokens.
Monitoring and logging	audit and fraud mechanisms.
Split-world attacks	via blockchain.

to either incompletely log or entirely omit the fraudulent certificates from Certificate Transparency (CT) logs.

Solution: SemiDec-PKI incorporates multiple certificate authorities (CAs) in its control mechanism to prevent fraudulent documents and impersonation attacks concerning domain ownership through cross-checks with voting dynamics. It employs a hierarchical structure for auditing and cross-checking the entire PKI ecosystem. To eliminate *Split-world attacks* SemiDec-PKI uses a blockchain-based approach without requiring more than one log server.

All the issues and the corresponding solution proposed by SemiDec-PKI are summarized in Table 4.1.

To address these issues, SemiDec-PKI introduces a novel approach by combining centralized and Web-of-trust paradigms, utilizing smart contracts to establish a robust Public Key Infrastructure (PKI). The system incorporates a voting mechanism to eliminate *Single Point of Failure* in scenarios where a Certificate Authority (CA) may be compromised or deceived. Additionally, it employs a hierarchical structure that facilitates auditing and cross-checking of the entire PKI ecosystem and enables *Monitoring and Logging*.

Unlike conventional PKI operations, SemiDec-PKI resolves the issue of CA *Revocation monopoly* by actively involving multiple CAs in the certificate issuance process. Voters in this design consist of Supervisor Bodies and CAs. The issuance of a CA is dependent on the votes of Supervisor Bodies. SemiDec-PKI provides *Management of trust lists* and additionally, the system allows for expansion by welcoming participants from new countries

into the SBs, and it provides a voting process where existing Supervisor Bodies can vote for new SBs. This collaborative approach enhances security and resilience. Furthermore, implementing smart contracts in SemiDec-PKI ensures the seamless validation, issuance, and revocation of certificates, reducing the risk of erroneous transactions in *Certificate Registration* and supporting the system's overall integrity. By embracing these innovative features, SemiDec-PKI presents a promising advancement in the field of Public Key Infrastructures, contributing to a more secure and efficient framework. SemiDec-PKI also incorporates a *Punishment Mechanism* as a pivotal component of its security infrastructure. In the event of an attempted manipulation by an attacker, the system responds by imposing appropriate penalties, as determined by the voting results. In consequence, the malevolent actor stands to lose ERC tokens as a consequence of their illicit actions. This punishment mechanism serves as a powerful deterrent against malicious behavior, underpinning the system's resilience and integrity in the face of potential threats.

SemiDec-PKI, redefines the following mechanisms based on blockchain: (1) certificate issuance, (2) certificate revocation, and (3) audit and fraud reporting. The first mechanism defines the steps of certificate issuance for supervisory bodies, CAs, and for end-users. Certificates are validated by supervisory bodies and CAs using a smart contract-based operation reinforced with a stake-based reward-punishment-based mechanism. In the second mechanism, the proposed SemiDec-PKI solution defines a revocation mechanism that can not only be triggered by issuers as expected, but also by certificate owners. Lastly, the SemiDec-PKI system is monitored by auditors who are willing participants seeking rewards, act as full nodes voluntarily, and continuously verify system activity. Whenever auditors identify an issue, they promptly report it and receive ERC-20 tokens for their efforts.

In SemiDec-PKI, a smart contract is defined with the state variables listed in Table 4.2. The requirements of these variables are denoted as "sh" for shall, "c" for conditional, and choice for "ch." Considering the size of each variable and assuming String is 32 bytes, the size of a certificate is approximately 256 bytes, which is half of the average X509 certificate size (512 bytes) given in CertLedger [7]. All mechanisms utilize these variables, as given in detail below.

Table 4.2 SemiDec-PKI variables and requirements

State Variable	Explantion	Type	Req.
certificateId	serial number of the certificate	uint64	sh
includeTransactionKey	transaction key inclusion status	bool	c
subject	identity information	string	sh
certificateType	type of the certificate	uint8	sh
publicKey	ECC256 compressed public key	bytes32+1	ch
issuerCertId	certificate Id of the issuer certificate	uint64	sh
expirationDate	expiration date of the certificate	uint32	sh
ownerAddress	Ethereum address of the owner	bytes20	ch
issuerAddress	Ethereum address of the issuer	bytes20	sh
audit	IPFS address of audit/registration documents	bytes32	c
cryptographicSignature	cryptographic signature of the certificate	$2\times$ bytes32	c
positiveVotersCount	stands for a trust measure	uint8	sh
negativeVotersCount	stands for a trust measure	uint8	sh
positiveVoters	addresses of positive voters	bytes32	sh
negativeVoters	addresses of negative voters	bytes32	sh
revocationStatus	status of the certificate	bool	sh
waitingTime	penalty for improper voting	uint32	c
X509Certificate	IPFS address of X509	bytes32	c

4.2. Certificate Issuance

An authoritative Certificate Authority (CA) is responsible for issuing certificates containing a public key and the identity of the owner. Certificate issuance signifies the CA's validation that the public key within the certificate corresponds to the entity, whether an individual, organization, server, or other entity, as specified in the certificate. In the context of traditional PKI, the issuance of inaccurate certificates can arise due to errors during the verification and certificate generation processes. These vulnerabilities expose the system to potential attacks. If attackers succeed in acquiring the CA's authority keys, they can generate additional certificates, undermining the system's security. The reliance on a single CA for generating certificates also brings about risks of Single Points of Failure (SPoF), as the entire system's reliability depends on the trust vested in that CA. In order to address these challenges, the implementation of a decentralized monitoring and auditing system becomes imperative. This system should ensure that even if an attacker gains control over a supervisory body or a CA,

the overall system's availability remains unaffected. Additionally, it should prevent attackers from issuing unauthorized certificates to end-users or CAs.

The proposed approach, SemiDec-PKI presents a viable approach to meet these requirements through a voting scheme based on a stake-based reward-punishment mechanism. By incentivizing stakeholders to actively participate in the voting process and penalizing malicious actions, the system ensures a collaborative and secure environment. It significantly mitigates the risks associated with SPoF, and with participant of more CAs reduce certificate registration issues.

In the proposed approach, Certificates Authorities (CAs) and supervisory bodies (SBs) serve as authorized issuers during the certificate issuance phase. When a CA is issued, it can initiate issuing end-user certificates. Initially, a smart contract (SC) defines the set of supervisory bodies, who have the authority to issue a new CA or supervisory certificates. Certificate issuance operates on a voting mechanism, where both SBs and CAs play pivotal roles. SBs partake in voting to suggest the inclusion of new SBs or the issuance of new CAs. CAs cast their votes to endorse the issuance of end-user certificates. When the number of votes surpasses a predetermined threshold, the proposed addition or issuance moves forward.

The voting threshold is of utmost significance within this process, as it establishes the minimum number of votes required to attain consensus among stakeholders and validate the action. Each voting transaction accrues gas fees, and conducting numerous voting sessions might result in reduced overall cost-effectiveness and system efficiency. Therefore, the threshold should consider the trade-offs between augmenting security through increased cross-checks and maintaining the practicality and effectiveness of the system. Please note that the threshold value in smart contracts can be modified through majority consensus among relevant voters.

The voting mechanism within SemiDec-PKI integrates a stake-based reward-punishment system to uphold the network's integrity and security. All voters, including supervisory bodies (SBs), Certificate Authorities (CAs), and end-users, are involved in the voting process by staking tokens. Hence the proposed approach encourages honest and responsible

participation, as individuals have a vested interest in the stability of the system. If a voter or end-user commits any malicious or erroneous act, the consensus mechanism enforces punishments through agreement of network participants.

The certificate issuance steps within the SemiDec-PKI system are depicted in Fig. 4.1 and can be summarized as follows:

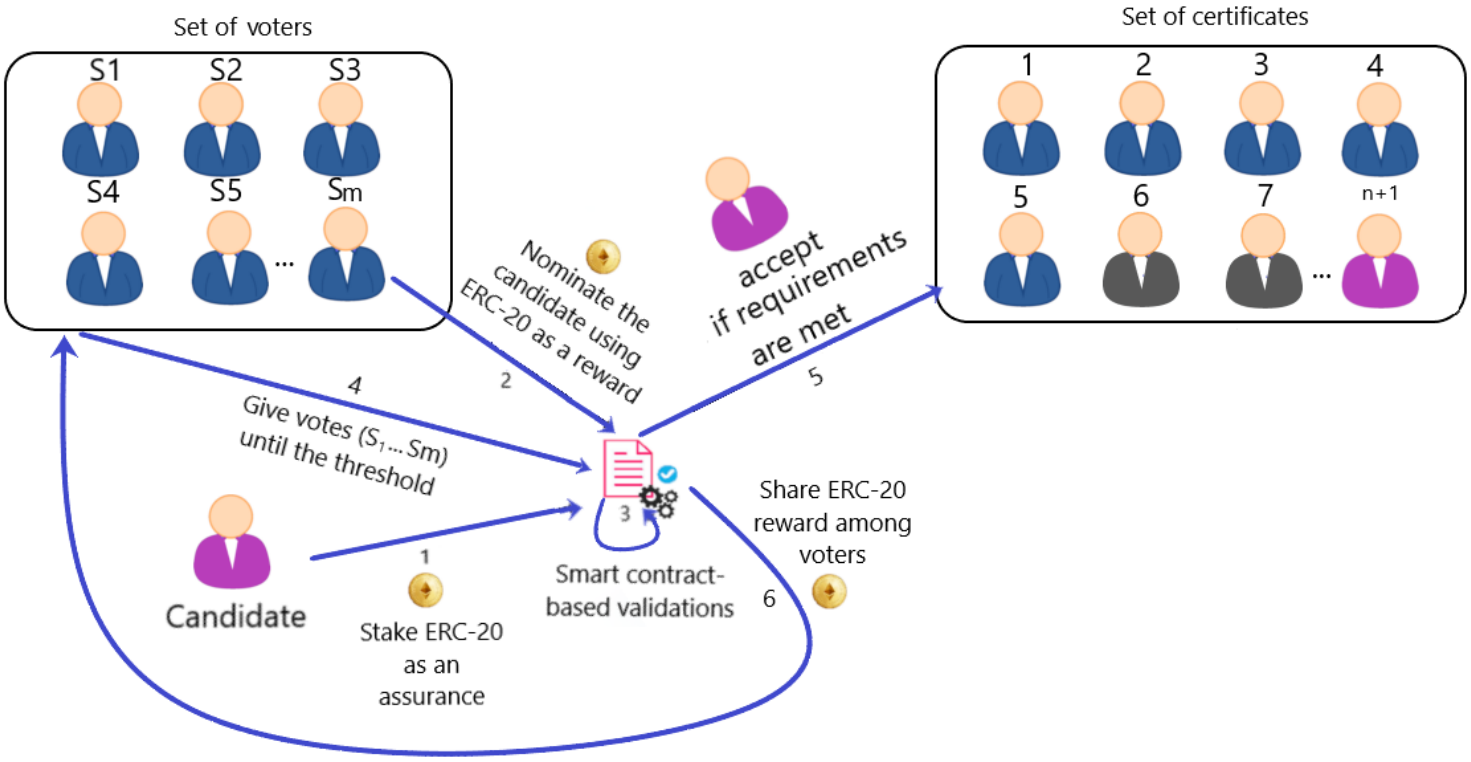

Figure 4.1 Certificate Issuance Mechanism

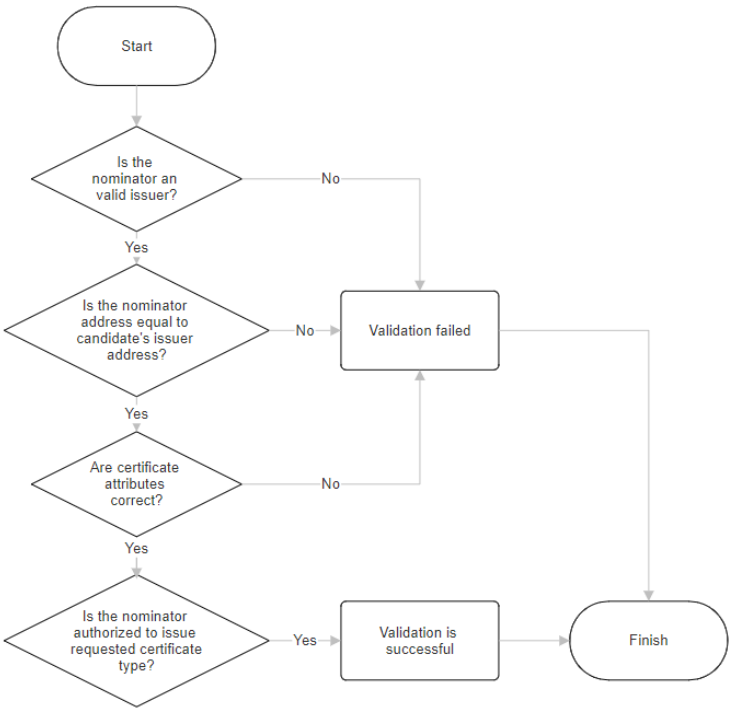

Figure 4.2 SC-based Validation Steps on Certificate Issuance

(1) Where a candidate is an issuer, it needs to stake ERC-20 tokens as a blocked safety deposit for the purpose of assurance. (2) An issuer sends a transaction using ERC-20 tokens as a reward in order to add a new certificate candidate, which in turn triggers the smart contract-based validation. (3) To start the voting process, the following SC-based validation requirements need to be met in the SC as shown in Fig. 4.2. If these requirements are not met, the SC-based validation fails, and the token is subsequently refunded. However, if the requirements are met, the voting process proceeds according to the following requirements defined in the SC. (4) If the SC-based validation is confirmed, voting for the acceptance of this new certificate begins. If the candidate is an issuer certificate then SBs can vote; however, if it is an end-user certificate then it is the CAs that can vote.

The requirements of the voting mechanism are specified as follows: Voters must have valid certificates, and have an adequate stake. The waiting time (penalty) must expire of voters who are punished for previously registering improper votes. If a *cryptographicSignature* is given in the certificate, its verification must be conducted by the participants during

voting. Voters need to assess documents such as the registration documents and audit reports uploaded by the issuer. Only a single vote can be given for each certificate candidate. Each voting operation has a cost, with the *voting threshold* value used to prevent excessive cost expenditure. As such, voting is discontinued if positive or negative votes counts reach the voting threshold defined in the SC. (5) Only candidates that successfully pass the voting mechanism will be issued. Even though an attacker may compromise a supervisory body, it cannot forge a trusted supervisory body or a CA, which prevents the *Single Point of Failure*. (6) Reward tokens are shared among the winning voters.

4.3. Certificate Revocation

Certificate revocation is a critical process that invalidates a digital certificate for various reasons, including compromise of the associated private key, loss or deletion of the certificate, changes in entity information, discontinuation of use, or suspicion of misuse. Revocation of a certificate means invalidating the certificate before its expiration date. In the traditional infrastructure, end-users or CAs can initiate the revocation process. However, the process itself can only be conducted by only CAs. CAs provide certificate revocation information through mechanisms like the Certificate Revocation List (CRL) and the Online Certificate Status Protocol (OCSP).

In contrast, the proposed SemiDec-PKI introduces a novel approach to the revocation process. In this system, the responsibility for certificate revocation lies with the certificate owners or issuers, who can initiate the revocation process via smart contracts (SC) as depicted in Fig. 4.3. This decentralized revocation process empowers certificate owners to take prompt action and eliminate the need for a *revocation monopoly*, enhancing the efficiency and responsiveness of the PKI infrastructure.

4.3.1. Revocation triggered by the issuer:

When the certificate issuer initiates the revocation mechanism in the proposed SemiDec-PKI, some validations take place as shown in Fig. 4.4.

Algorithm 1: Certificate Issuance Mechanism

Data: Candidate, ERC-20 tokens
Result: Issued Certificates, Distributed Rewards

Issuer Staking: ;
    for *each candidate* **do**
        Where a candidate is an issuer, stake ERC-20 tokens as a blocked safety deposit for assurance. ;
Add New Certificate Candidate: ;
    for *each issuer* **do**
        Issuer sends a transaction with ERC-20 tokens as a reward to add a new certificate candidate, triggering smart contract-based validation. ;
Smart Contract Validation: ;
    for *each candidate* **do**
        Start voting process with SC-based validation requirements. ;
        if *IssuerAddressInCAList(Candidate)* ***and*** *IssuerAddressMatches(Candidate)* **then**
            if *IssuerEligibility(Candidate)* ***and*** *MandatoryFieldsCompleted(Candidate)* **then**
                Proceed with the voting process. ;
            else
                SC-based validation fails, refund tokens. ;
        else
            SC-based validation fails, refund tokens. ;
Voting Mechanism: ;
    for *each candidate* **do**
        If SC-based validation is confirmed, start voting for a new certificate. ;
        Voters must have valid certificates and adequate stake. ;
        Waiting time must expire for voters punished for improper votes. ;
        if *CryptographicSignaturePresent(Candidate)* **then**
            Verify cryptographic signature during voting. ;
        Voters assess documents like registration documents and audit reports. ;
        Only a single vote can be given for each certificate candidate. ;
        Voting operation has a cost; voting threshold prevents excessive costs. ;
        Discontinue voting if positive or negative votes reach the threshold. ;
Issuance of Successful Candidates: ;
    for *each successful candidate* **do**
        Only candidates passing the voting mechanism are issued. ;
        Attacker cannot forge a trusted SP or a CA, preventing the SPoF ;
Reward Distribution: ;
    for *each winning voter* **do**
        Reward tokens are shared among the winning voters. ;

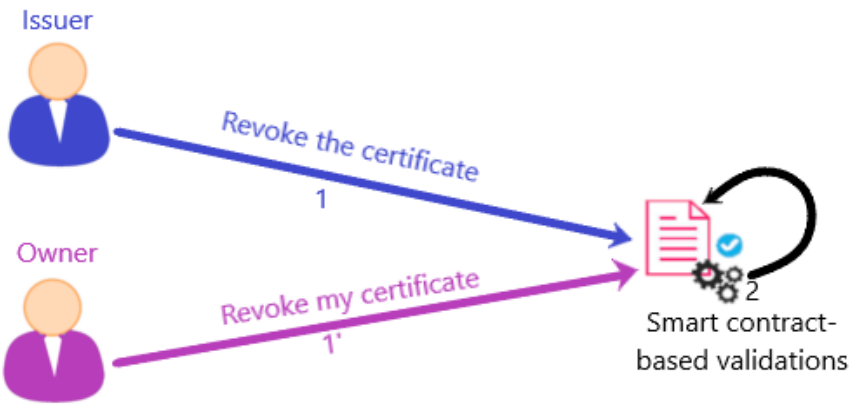

Figure 4.3 Certificate Revocation Mechanism

4.3.2. Revocation triggered by the owner:

If the certificate to be revoked includes the owner's Ethereum address, then the owner can revoke it by sending a transaction. Again, the sender's address must match the *ownerAddress* of the certificate. Then, the certificate will be revoked if the transaction is valid. Please note that supervisory bodies are unlikely to be revoked. Since the initial supervisory bodies are defined in a smart contract, they do not have an issuer. However, if a supervisory body is compromised and labeled as malicious based on the voting mechanism carried out by other supervisory bodies as given in the audit and fraud reporting mechanism, then their certificates are revoked. This decentralized revocation process empowers certificate owners to take prompt action in response to security concerns and ensures the overall integrity and trustworthiness of the SemiDec-PKI system. Moreover, it adds a layer of resilience and agility to the PKI system, enhancing its adaptability to changing security requirements and potential threats by eliminating *revocation monopoly*.

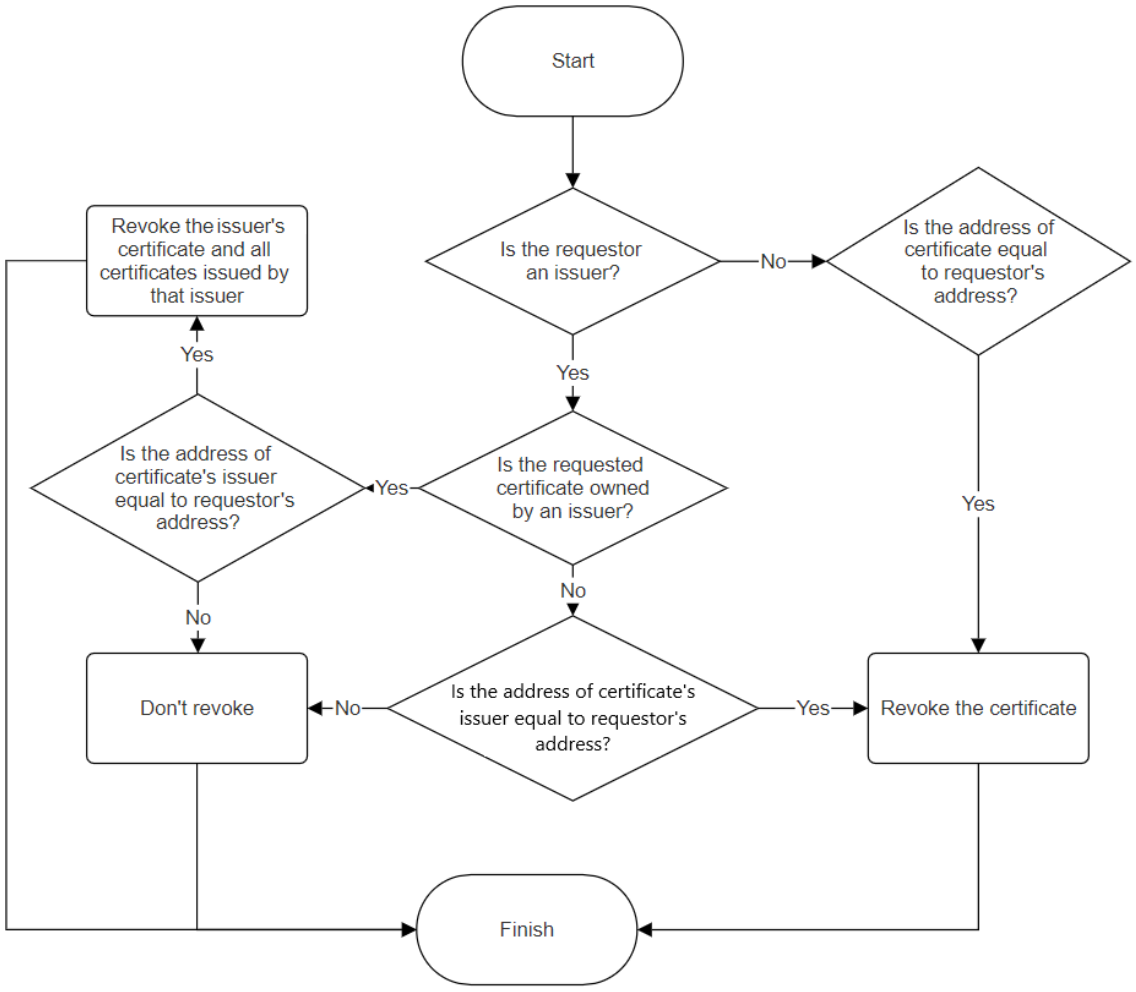

Figure 4.4 SC-based Validation Steps on Certificate

4.4. Audit and Fraud Reporting

Any participant who volunteers to become a full node can be an auditor. In the fraud reporting mechanism, for the inclusion of a candidate certificate, a new vote is arranged, and the previous voters are blocked from the current voting by being added to a temporary ban list. When an auditor detects a forgery regarding certificates in the system, the steps shown in Figure 4.5 are applied in the following order: In step 1, the auditor uploads evidence to IPFS. In step 2, the auditor sends a transaction for fraud reporting using ERC-20 as an assurance. In step 3, all those who previously voted for the reported certificate are banned recursively. In step 4, voters assess the evidence. In step 5, eligible authorities vote for the fraud report

until the voting threshold has been reached. In step 6, the certificate's status is updated according to the election result. In the final step, when the voting concludes as fraud, the *revocationStatus* of the certificate is updated, and the auditor and winner voters punish the previous voters and the issuer by getting their tokens and applying them with an incremental waiting time penalty. If the voting fails to conclude fraud, the winner voters earn ERC-20 by receiving the auditor's assurance tokens.

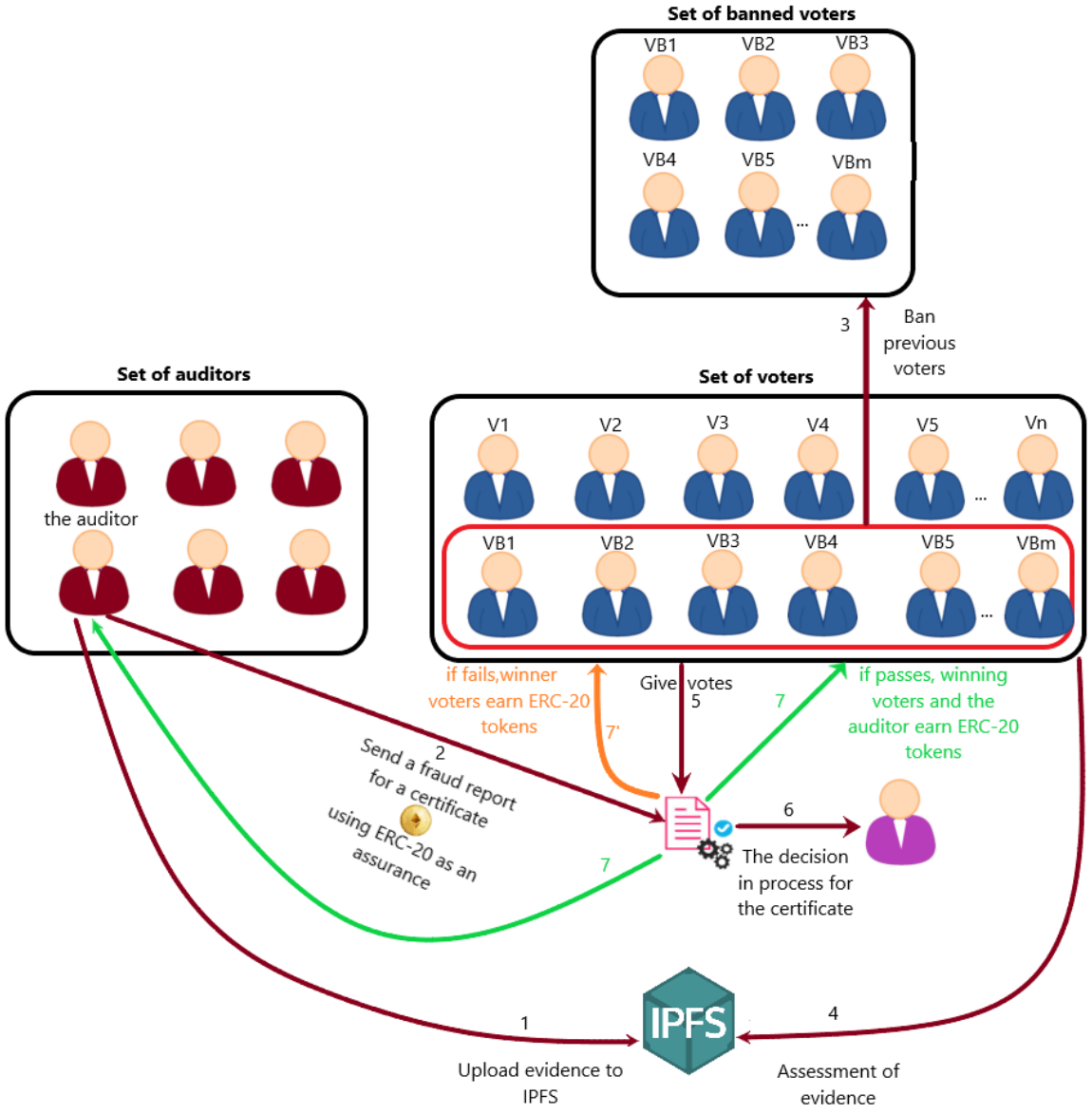

Figure 4.5 Fraud Detection Mechanism

As previously stated, voting is concluded when either the positive or negative votes reach a certain threshold. In theory, numerous attackers can dominate the voting for a transaction because of this threshold-based approach. However, auditors who can send a fraud transaction by a stake can prevent such cases. Once the transaction is sent, a chain of events is triggered. Firstly, SC blocks voters in the previous voting and starts new voting. Validators vote, and the reporter and new voters receive the ERC-20 tokens by punishing attackers where fraud is proven. However, where fraud is not proven, the tokens staked by the auditor are shared among the validators.

Algorithm 2: Fraud Reporting Mechanism

Data: Candidate Certificate, Evidence, ERC-20 tokens
Result: Revocation of Fraudulent Certificates, Punishment of Participants

Auditor's Fraud Reporting:
    Any participant who volunteers to become a full node can be an auditor.
    For inclusion of a candidate certificate, arrange a new vote and block previous voters from current voting by adding them to a temporary ban list.
Fraud Detection Steps: foreach *reported certificate* **do**
    Step 1: Auditor uploads evidence to IPFS.
    Step 2: Auditor sends a transaction for fraud reporting using ERC-20 as assurance.
    Step 3: Ban all previous voters recursively for the reported certificate.
    Step 4: Voters assess the evidence.
    Step 5: Eligible authorities vote for the fraud report until the voting threshold is reached.
    Step 6: Update certificate's status based on the election result.
    Final Step: if *voting concludes as fraud* **then**
        Update *revocationStatus* of the certificate.
        Auditor and winner voters punish previous voters and the issuer by taking their tokens and applying an incremental waiting time penalty.
    else
        Winner voters earn ERC-20 by receiving the auditor's assurance tokens.

In summary, the proposed approach, SemiDec-PKI, effectively tackles various infrastructure challenges, including *trust list management*, *Single Point of Failure (SPoF) prevention*, *certificate registration*, *revocation monopoly*, *punishment*, *monitoring and logging*. Trust Lists are efficiently managed and governed by predefined Supervisory Bodies (SBs) within smart contracts (SC), streamlining the *trust list management* process. The SemiDec-PKI architecture, which combines Web of Trust and centralized approaches, ensures resilience

against *SPoF* by involving multiple CAs in the certificate issuance process. The introduced voting scheme significantly enhances the system's robustness, thwarting impersonation attacks arising from *certification registration* issues. Furthermore, this approach eliminates the need for centralized revocation authorities, empowering end-users to independently revoke their certificates and eradicating the *revocation monopoly* issues. A stake reward and *punishment* mechanism promotes active participation among end-users and voters by incentivizing positive contributions and disincentivizing malicious or erroneous behavior. This mechanism ensures a collaborative and secure environment within the system. Furthermore, SemiDec-PKI utilizes a hierarchical structure that simplifies auditing and cross-verification of the entire PKI ecosystem, allowing for *monitoring and logging*. To comprehensively evaluate the proposed approach, subsequent sections delve into more detailed analyses of its security, usability and performance aspects.

4.5. Security, Usability and Performance

In this section, we discuss the applicability of the designed system on the basis of security and performance.

4.5.1. Security

Threats to the proposed design are investigated under three groups: Threats against the infrastructure; cryptographic threats; blockchain-based threats.

Threats against the infrastructure: A semi-decentralized structure is established in SemiDec-PKI, and the proposed voting mechanism eliminates single points of failure. Even if the most potent entities, supervisory bodies, are compromised, attackers cannot issue a CA certificate, supervisory certificate, or an end-user certificate.

In SemiDec-PKI, the validation steps are secured by the proposed stake-based reward-punishment mechanism, and a voting threshold value is defined to limit the number of transactions for each voting. Since voters are composed of supervisory bodies and CAs, they

are unlikely to be compromised. However, in an extraordinary scenario in which the number of attackers (n) is bigger than or equal to the threshold (m) (i.e. $n \geq m$), attackers could potentially manipulate the first voting. Of course, an honest auditor would submit a fraud report for the transaction in this scenario, so a new voting would be triggered. Therefore, m attackers that took part in the previous voting would be blocked; hence the remaining $n - m$ attackers would still be able to participate in the new voting. In the case that the sum of $n-m$ is still may be greater than m ($n - m \geq m$), attackers could still dominate the voting, and a part of the auditor's stake would be given to the attackers. These steps are therefore repeated for n/m times, so that the attackers would lose in any voting process where dominated by benign nodes. As a result, the tokens of benign validators would be refunded, whilst the attackers would lose a considerable amount of tokens, and the attackers' certificates could be revoked due to the fraud reporting mechanism.

When a Certificate Authority (CA) validates an end-user's application, it casts a vote. During this process, CAs can be susceptible to impersonation attacks. For an attacker A to succeed, it must deceive a number of CAs greater than a defined threshold T. Suppose the system includes a total of n CAs, and attacker A has a probability p_r of deceiving each CA. The number of deceived CAs, denoted as Z, follows a probability distribution $P(n, p)$. Consequently, the probability that Z is at least T can be expressed as:

$$P_r(Z \geq T) = \sum_{x=T}^{n} \binom{n}{x} p^x (1-p)^{n-x} \tag{1}$$

The correlation between P and n, T is illustrated in Figure 4.6. It is given that for any n, it can be selected a proper T in order to minimize the probability. Even p_r is picked as 0.5, it is hard to achieve an impersonation attack and deceive a CA in real life. In each voting process, the participation of multiple Certificate Authorities (CAs) enhances the security of the certificate registration process by facilitating increased cross-checking. This multi-party involvement ensures a higher level of scrutiny and validation, thereby reducing the chances of fraudulent or erroneous certificate issuance. However, it is essential to consider that every voting transaction incurs gas costs within the smart contract (SC) environment. As

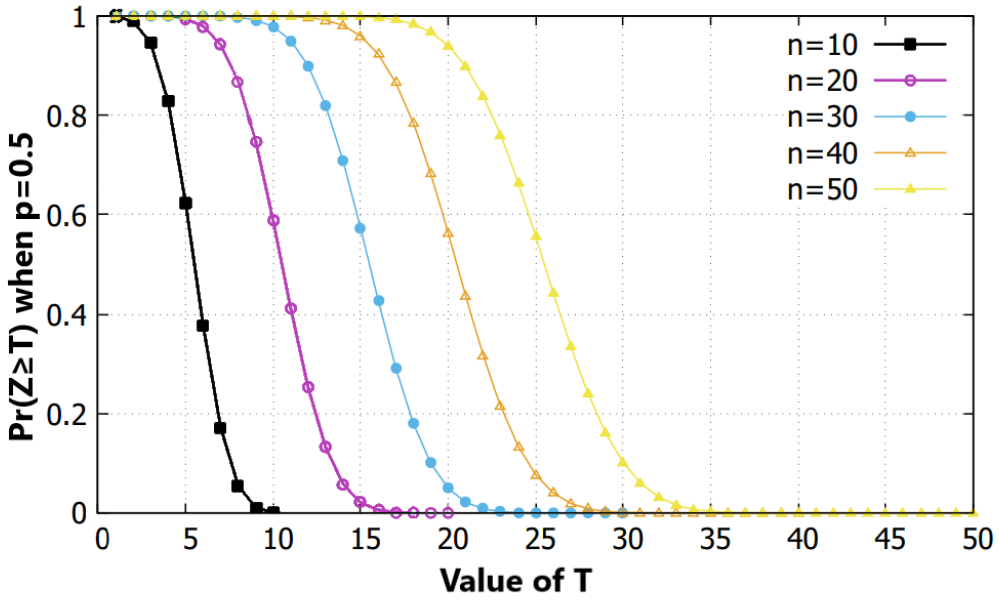

Figure 4.6 The probability of deceiving T CAs; assuming an impersonation attack success rate is 50%. $p = 0.5$

a consequence, conducting multiple voting sessions may lead to an overall decrease in the cost-effectiveness of the system, affecting its efficiency within the SC framework. Striking a balance between the enhanced security achieved through increased cross-checking and the associated gas costs is critical to optimizing the SemiDec-PKI system's performance and ensuring its practicality and sustainability within the blockchain-based ecosystem. So we initially picking five CAs (T=5) in order to increase security and reduce assuming risk from 0.5 to $0.5^5(0,03125)$. However, to maintain flexibility and optimize efficiency, the voting threshold value needs to be adjustable. Thus, we implement a smart contract feature that allows initiating a voting process for changing the threshold value. When this voting is triggered, the approval of at least $(n/2) + 1$ voters is required to approve the threshold change. By empowering the stakeholders to modify the threshold value, the SemiDec-PKI system ensures adaptability to changing security requirements while maintaining a practical balance between security and efficiency.

Cryptographic threats: Thanks to the immutability characteristic of blockchain, all transactions and certificates are protected in SemiDec-PKI, as supervisory bodies and CAs

use not only transaction keys but also private keys for signing. Thus, an attacker would need to take possession of both keys in order to render harm to the system. Please note that SemiDec-PKI uses ECC256 keys and allows rekeying.

Blockchain-based threats: The proposed infrastructure is built on Ethereum, which is still hypothetically vulnerable to threats in which attackers compromise 51% of the system. If a group of attackers controls more than 50% of the mining hash rate or processing power, they could block new transactions from being confirmed, thereby stopping payments among users [107]. While this threat is still proof-of-stake, attackers would need to outlay considerable amount of money in order to take control of 51% of the Ethereum (ETH). In addition, such actions would trigger a downturn in the value of Ethereum, which would directly conflict with the attackers' motivation.

4.5.2. Usability

In this study, we present a novel approach that does not rely on the ASN.1 format, which is essential for representing data within X.509 certificates. These certificates are easily understood when used in common systems like web browsers. However, things get more complicated when we enter the realm of blockchain-based smart contracts [108, 109]. Our approach focuses on efficiently storing specific X.509 attributes as variables within the smart contract structure. Our challenge here is to provide a smooth transition to a smart contract-friendly certificate structure without sacrificing the familiarity and reliability associated with the X.509 format. Therefore SemiDec-PKI eables the attachment of an IPFS address for the standard X.509 certificate alongside the new blockchain-based certificates for backward compatibility. User could choose between these two formats based on their specific requirements and preferences during the transition.

4.5.3. Performance

End-users are light nodes, and voters and auditors are defined as full nodes in the proposed infrastructure. The PoW algorithm is utilized in the current Ethereum system. However, the increased number of transactions (Txn) and the subsequent rise in Ethereum prices would lead to an increase in the total price for transactions and deploying SCs. Therefore, this consensus algorithm is planned to be changed to Proof of Stake (PoS) in Ethereum 2.0. Hence, it will be able to support more transactions and the use of SCs. Moreover, it is expected to decrease energy consumption.

Here, the storage cost of the proposed infrastructure based on blockchain is calculated as in [2]. There are about 3.64×10^8 registered domain names (as of 2021 [30]) and 46 million websites using SSL certificates [110]. Therefore, in a worst-case scenario, let us consider that half of all registered domains are assumed to use SSL certificates [111]; the storage cost is calculated as follows, based on the following assumptions:

- The total number of all other certificate types is equal to 3.64×10^8.
- Redefined certificates in this proposal would be approximately 256 bytes, or half of the average X509 certificate size (512 bytes) given in CertLedger [2].
- SSL certificate maximum lifetime period is defined as approximately 1 year (398 days) in a recent CAB Forum Ballot [49]. For the sake of simplicity, the worst case scenario is adopted for all other certificate types, and the certificate lifetime is defined as 1 year.
- PoS is used as the consensus algorithm in Ethereum. In PoS, the average block time, which is the time it takes to generate a new block, is 12-14 seconds [89].
- Adding new certificates is expected to cover the majority of transactions (Txn) in SemiDec-PKI, so the costs of all other transactions are ignored here.
- We also assume that certificates are issued homogeneously throughout the year, and the block time is 12 seconds [2].

Hence, five blocks are generated in 60 seconds. Then, the total number of blocks generated in 1 year would be 26,280,000 ($365 \times 24 \times 60 \times 5$). The number of transactions (Txn) is equal to the total number of certificates (3.64×10^8) / generated blocks in a year, as in CertLedger [2] shown in the following:

$$Number\ of\ Txn := \frac{Number\ of\ Certificates}{Annually\ Generated\ Blocks} \tag{2}$$

The size of a block (BS) is given in Equation 3.

$$\begin{aligned} BS :=& Txn\ Size \times Number\ of\ Transactions \\ &+ Header \end{aligned} \tag{3}$$

$$Txn\ Size := Message + Signature \tag{4}$$

$$Message := PKSender + Receiver + Data \tag{5}$$

where PKSender, Receiver, Data, and Signature correspond to the size of the sender's public key (64 B), the receiver's address (20 B), the certificate (256 B), and the size of the signature (64 B) in a transaction. Hence, the transaction size becomes 404 B. The header size of a block is fixed at 508 B [112]. Hence, the average size of a block becomes 6,103 B. The total size of the blockchain are calculated as in CertLedger [2]:

$$\begin{aligned} Blockchain\ Size :=& Annually\ Generated\ Blocks \\ &\times BS \end{aligned} \tag{6}$$

Hence, the blockchain size of a full node is approximately 150 GB, as given in Equation 5. Since the cost of 1 GB of disk storage is about 0.02 USD [113], the combined cost for all certificate transactions would be approximately 3 USD per annum. On the other hand, light nodes do not store the entire block, just the header (508 bytes per block). As such, the total blockchain size of a light node per year would be approximately 640 MB ($26,280,00 \times 508$).

In the literature, certificates are logged to the blockchain and sent to the client in X509 format. In this study, the log in the blockchain was used without sending a separate certificate in the X509 standard. While the X509 certificate size is 512 B on average [2], our certificates are approximately 256 B.

Certificate issuance time is not measurable because registration authorities checks the appliance documents and gives votes according to validation result. But in the certificate transparency which in usage on SSL when a certificate is submitted to a log successfully, the server sends a Signed Certificate Timestamp (SCT) as proof and promise to add the certificate in the Merkle Tree within a fixed amount of time known as the Maximum Merge Delay (MMD) [70]. MMD is usually 24 hours [114]. While the voting mechanism ensures cross-checking, CA croos-check durations are expected not to cause performance issues if they are reasonable.

4.6. Comparison with Related Studies

The comparison criteria are selected according to cover the most critical shortcomings of PKI and mainly taken from Certledger [2]. In addition to them, new criteria regarding security and operation are added as shown in Table 4.3: *Punishment Mechanism*, *Commercial Model*, *Enables Crosscheck*, *Support Any Certificate*.

Certificate validation: Certificate revocation is an essential process that does not rely on third-party validation, as the certificate itself contains information about its expiration date and revocation status. In SemiDec-PKI, it does not rely on third-party certificate validation, as the certificate itself contains information about its expiration date and revocation status. As in Certledger [2], KORGAN [6], and Hwang et al. [7], clients only need to verify proofs. Yakubov et al. [1] utilizes smart contracts or web services. SCPKI [3] and ETHERST [4] depend on the Web of Trust. SCPKI [3] redefines the certificate as *Attribute* and validates its *Signature* and *Revocation* as a validation step. ETHERST [4] contributes SCPKI and adds *trustorCount* attribute to *Signature*.

Log proofs: In Certificate Transparency, end-users or CAs make certificate-related logs into servers managed by different centers. There may be differences and synchronization issues in these log servers that tried to be corrected with the gossip protocol [84]. In SemiDec-PKI, all data, including the certificate itself, is kept on the blockchain. Therefore, a single log is copied multiple times and distributed in the system. Thus, inconsistent logs related to a certificate are prevented from being found on different servers. SemiDec-PKI provides proof of existence and revocation status for all certificates as in Certledger [2], KORGAN [6], Yakubov et al. [1], BBPKI [5], Hwang et al. [7], SCPKI [3], and ETHERST [4].

Auditing – Monitoring: The Certificate Transparency project reveals the concept of monitoring and auditing certificates. Third-parties audit certificate logs on different, partially independent log servers. This way, it is ensured that the logs on different log servers are consistent. Although Certledger [2], Yakubov et al. [1], and BBPKI [5] do not need an audit for consistency, they do not offer an additional audit mechanism before the certificates (CA and end-user) are produced and valid. A CA may generate a certificate that should not have been generated in the first place, and this generation cannot be revealed until a complaint. In this situation, end-users are expected to notice that CAs have been attacked or their certificates are issued by mistake. Likewise, these logs must be audited by third parties. In Hwang et al. [7], the end-user is also included in the certificate generation in order to solve this issue. On the other hand, Yakubov et al. [1] introduces an audit mechanism, and KORGAN [6] further builds upon this concept by incorporating a threshold signature mechanism, making audits verifiable for anyone possessing the public key of block signers. These solutions, however, necessitate external monitoring for their effectiveness.

While, Yakubov et al. [1] and KORGAN [6] requires an external monitoring, SemiDec-PKI does not due to its own audit and monitoring mechanism.

Assurance- Punishment - Commercial model: CAs require to take out insurance in conventional PKI for the damages resulted from faulty certificates. In such cases, end-users and CAs must compromise or the matter goes to court. Certledger [2] proposes a fraud reporting mechanism that only suggests that penalties such as financial or total

prohibition may be imposed. ETHERST [4] utilizes a reward-punishment mechanism to encourage commercial adaptation of blockchain-based PKI. In the web of trust mechanism, untrusted nodes are punished with PKITokens. Yakubov et al. [1], SCPKI [3], KORGAN [6], BBPKI [5], and Hwang et al. [7] do not provide a punishment mechanism. In SemiDec-PKI, this insurance is guaranteed by initial assurance tokens. Complaints are created, and the result of the complaint is concluded with a reward-punishment system. Certledger [2], ETHERST [4], and SCPKI [3] also provides a commercial model for incentive shareholders. While SCPKI [3] and ETHERST [4] utilizes ERC tokens, SemiDec-PKI provides a commercial model that includes an assurance mechanism in addition to the punishment-reward mechanism.

Single point of failure - Crosscheck : In traditional PKI, Yakubov et al. [1], Certledger [2], KORGAN [6], BBPKI [5], and Hwang et al. [7], the certificate is activated after the certificate is issued, and faulty situations are handled only afterwards. In addition, Hwang et al. [7] includes end-users in the system in order to increase controls on certificate issuance process. However, in cases where end-users could be less conscious, it is possible to generate problematic certificates. SemiDec-PKI provides a cross-check mechanism for each certificate that includes other CAs to the certificate issuance. Thanks to this mechanism, a certificate is not accepted and will only be activated with the review of at least two CAs. Therefore, even if an attacker captures a CA, he cannot generate end-user certificates without the permission of other CAs due to the voting mechanism. Even if an attacker compromises the supervisory body, the most powerful actor in the system, it can not issue a certificate without going through the voting mechanism. Thus, single-point-of-failure is prevented. When a CA's certificate is issued, Certledger [2] asks for the approval of more than one of the management board members. However, when one of CAs is compromised, it cannot prevent attackers from issuing certificates.

Certificate management: Certificates conform to the X509 standard and are encoded with ASN.1, which is a formal notation used for describing data transmitted by telecommunications protocols. In existing PKI models, certificates are produced by the CA and sent to end-users. Although Certledger [2], KORGAN [6], BBPKI [5], Hwang et al. [7],

and Yakubov et al. [1] proposed new approaches, they still use the same X509 certificate standard. However Certledger [2] stated that encoding and decoding the certificate based on ASN.1 is problematic for smart contracts and decoding is not implemented. ASN.1 decoding is a challenging process for SCs and SemiDec-PKI suggest a new solution. SemiDec-PKI supports logging all certificate-related fields to the blockchain structure instead of using the certificate's X509 and ASN.1 notations. It utilizes these logs instead of X509 certificates. Thus, CAs can generate certificates only after filling in all certificate-related fields that are checked and approved by other CAs. Therefore, SemiDec-PKI does not need to follow the X509 standard. This approach is similar to the definition of *attribute* in SCPKI [3] and ETHERST [4], which replaces a new certificate.

Trust List Management: In conventional PKI, certificates are signed by the issuer while issued. While the certificate is being validated, a validation process occurs, starting from the end-user certificate to the root certificates of the issuers. For the verification, all issuers must have root certificates on the side that validates the certificate. The store where the issuers have root certificates is called a trust list. Clients do not have to store trusted keys or certificate logs during certificate verification on the client side in Certledger [2], BBPKI [5], and Yakubov et al. [1]. This way, certificate validation can be performed without a client-side trusted root store. KORGAN [6] also replaces and eliminates the client-side conventional trust list, but end-users must store the blockchain signing keys' public key. Additionally, the concept of a management board that responsible for adding root certificates is discussed in Certledger [2]. However it may require to increase the number of management board members for its world-wide application. SemiDec-PKI provides an internal trust list and extendable supervisory mechanism to increase supervisor bodies. Thus, it will be possible to dynamically include supervisory bodies of countries that join the system. SCPKI [3] and ETHERST [4] adopts a Web of Trust-based approach and are therefore exempt from trusted root or supervisory body concepts.

Users' self revocation: Revocation of the certificate is an essential issue in PKI. In the current PKI scheme, end-users contact CAs and perform certificate revocation via CAs. Certledger [2], KORGAN [6], and BBPKI [5] support user self-certificate revocation. In

Hwang et al. [7], users can send a change status request to Certificate Authority and perform the cancellation over the CA. Since there is no authority in SCPKI [3] and ETHERST [4], revocation can be done by only end-users. SemiDec-PKI also supports user self-certificate revocation.

Table 4.3 Comparison of security and certificate management

	SCPKI Al-Bassam 2017 [3]	Yakubov et al. 2018 [1]	Certledger Kubilay et al. 2019 [2]	KORGAN Kubilay et al. 2020 [6]	BBPKI Garba et al. 2020 [5]	Hwang et al. 2020 [7]	ETHERST Koa et al. 2021 [4]	LRS_PKI Liang et al. 2023 [14]	SemiDec-PKI
External Dependency During Certificate Validation	No	Yes	No	No	Yes	No	No	No	No
Existence of Logs with Different Content	No	No	No	No	No	No	No	No	No
Necessity of External Audit	Yes	No	No	No	No	Yes	Yes	No	No
Necessity of External Monitor	Yes	Yes	No	No	No	No	Yes	No	No
Assurance Mechanism	No	No	No	No	No	No	No	No	Yes
Punishment Mechanism	No	No	No	No	No	No	Yes	No	Yes
Commercial Model	No	No	Yes	No	No	No	Yes	No	Yes
Prevent Single Point of Failure	No	No	Partly[a]	Partly[a]	Partly[b]	No	Partly[c]	Yes	Yes
Enables Cross-check for Certificates	No[d]	No	No	No	Yes	No	Partly[c]	Yes	Yes
Trust List Management	No[d]	Yes[e]	Partly[g]	Partly[h]	Partly[g]	Partly[g]	No[d]	Yes	Yes
Storing Certificate	Yes	Yes	Yes	Yes	No	No	No	Yes	Yes
Support Any Certificate	Partly[i]	No[f]	No[f]	No[f]	No[f]	No[f]	Partly[i]	No[f]	Yes
Require X509 Certificate	No	Yes	Yes	Yes	Yes	Yes	No	Yes	No
Users' self revocation	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes
Architecture	Web of Trust	Centralized	Semi-decentralized	Semi-decentralized	Semi-decentralized	Semi-decentralized	Web of Trust	Semi-decentralized	Semi-decentralized

[a] If CA is compromised the attacker can issue a certificate
[b] Not for Supervisory
[c] Supports Voting Scheme on WoT
[d] Because of WoT Characteristics
[e] Only SSL Certificates but attackers can upload their smart contracts.
[f] Supports only SSL
[g] Adding supervisory bodies is not supported
[h] End-users need to store the validation keys of block signers
[i] Focus on identity management rather than certificates.

5. A NEW BLOCKCHAIN-BASED LONG TERM ELECTRONIC SIGNATURE FORMAT: BlAdES

In this section, a new signature format called BlAdES (Blockchain-based Long-Term Electronic Signature) specifically designed for long-term validity is introduced.

5.1. Design of a new signature format (BlAdES)

Electronic signatures represent a widely adopted and crucial application of digital certificates. Within the framework of eIDAS (Electronic Identification, Authentication, and Trust Services), advanced signature formats have emerged as key standards, each addressing specific requirements to ensure interoperability and address security concerns. Notable formats include CAdES [50], XAdES [52], PAdES [53], and JAdES [54], each contributing to the establishment of a secure and standardized electronic signature ecosystem.

Advanced Signature Formats in eIDAS The eIDAS regulation has spurred the development and adoption of advanced signature formats, each with distinct features and use cases:

CAdES (CMS Advanced Electronic Signatures) [50]: CAdES provides a comprehensive framework for advanced electronic signatures, specifying methods for creation, preservation, and validation [55–58]. It has been a foundational element in ensuring the security and reliability of electronic signatures.

XAdES (XML Advanced Electronic Signatures): Tailored for XML-based documents, XAdES extends the capabilities of electronic signatures to meet the specific needs of XML data structures. Similar to CAdES, it defines methods for creation, preservation, and validation, contributing to a broader scope of interoperability.

PAdES (PDF Advanced Electronic Signatures): Focused on PDF documents, PAdES establishes standards for creating, preserving, and validating electronic signatures within the

context of PDF files. This specialization is crucial for ensuring the integrity and authenticity of electronic signatures in the context of PDF-based workflows.

JAdES (JSON Advanced Electronic Signatures): Designed to accommodate electronic signatures in JSON based documents, JAdES addresses the unique requirements of JSON-based environments. Its specifications align with the broader eIDAS objectives, ensuring a consistent approach to signature creation, preservation, and validation.

Despite the clear definitions of methods for signature creation, preservation, and validation in each format, achieving interoperability among applications remains a significant challenge. Different implementations of these standards, as evidenced by ETSI plug tests [59], create hurdles for seamless interaction between diverse applications.

In response to the existing challenges in interoperability, the current study introduces a novel format known as Blockchain-Based Advanced Electronic Signature (BlAdES). This innovative format builds upon the foundations laid by previous studies, incorporating the essential methods for signature creation, preservation, and validation. What sets BlAdES apart is its utilization of Smart Contracts (SCs) to manage the signature infrastructure. By leveraging blockchain technology, BlAdES offers a groundbreaking solution that minimizes the complexities associated with diverse application interactions.

BlAdES not only supports the established methods found in CAdES, XAdES, PAdES, and JAdES but also takes a giant leap toward enhancing interoperability. The use of Smart Contracts as part of the signature infrastructure ensures a standardized and secure environment for electronic signatures, irrespective of the application used. This contribution significantly reduces the friction in cross-application interactions, addressing a longstanding challenge in the field of electronic signatures.

The main contributions of the proposed BlAdES are summarized as follows:

- BlAdES eliminates distinct timestamping processes with a slight delay caused by Ethereum block delays, hence timestamping a signature is no longer an issue.

Table 5.1 BlAdES Variables

Variable	Details	Type
signatureId	Id number of signature	uint
certificateId	Id number of signer's certificate	uint
messageDigest	sha256 hash value of document	string
isContentConfidential	Is to be signed document private	bool
data	IPFS address of content	byte32
signaturePolicy	0 = private key, 1 = transaction key	uint8
cryptographicSignature	IPFS address of the cryptographic signature	bytes32
signerAddress	Ethereum address of the signer	bytes20
parentSignatureId	for serial signatures; parentSignatureId	uint
extraMessageDigest	for confidential documents; Keccak-256 digest	byte32

- The signature submission process allows the signer and the submitter to be different entities, and the signer's Ethereum addresses can remain anonymous.
- BlAdES provides signature creation operations based on conventional private keys or transaction keys by utilizing SCs.
- BlAdES can preserve signatures for long-term validation with or without content, by not forcing signers to append documents to blockchain for the purposes of ensuring confidentiality.
- BlAdES includes signature validation via a reward-punishment mechanism using ERC-20 tokens [68] in order to eliminate issues related to interoperability or single point of failure.
- BlAdES addresses the challenges associated with large signature sizes. BlAdES achieves this through the elimination of the necessity of certificate values or revocation values. This reduction in signature size is a notable improvement, streamlining processes without compromising the integrity of the validation.

  Moreover, BlAdES introduces several innovative features:

5.2. Signature Creation

Our design provides two signature creation mechanisms. The first one uses traditional signing keys (may be held in Qualified Signature Creation Devices), whilst the second one utilizes transaction keys to create a BlAdES structure. BlAdES provides an alternative mechanism for confidential documents. Due to not including to-be-signed content in the Interplanetary File System (IPFS), it is prone to future manipulation of the content without a hash change by a second-preimage attack, hence it is essential to ensure that the document remains protected. Therefore, the sender uses an extra message digest procedure and adds a Keccak-256 hash value.

5.2.1. Creating signature with conventional private key:

As seen in Figure 5.2, (1) Signer starts off the signing process via the Signature Creation Application Service Component (SCASC), which can be any application that communicates with the Signature Creation Device (SCDev). (2) Certificate is validated on SemiDec-PKI by SCASC. (3) Data to be signed value (DTBS) is calculated by hashing (SHA256) a concatenation of *messageDigest* and *certificateId* H (*messageDigest* —— *certId*) and SCASC connects to SCDev. (4) DTBS is signed by the private key held in SCDev. (5) Signer obtains the cryptographic signature value. (6) Optionally, content is uploaded to IPFS. (7) Option to upload document to IPFS, or not. If content is uploaded, signer can send a transaction with the IPFS address of the content, or message digest, extra message digest, *certificateId*, and cryptographic signature. Please note that the system also supports serial signature, which has a different flow such that: DTBS consists of the parent's cryptographic signature (as a message digest) and *certificateId*, whilst IPFS content or the extra message digest are not used. Differently, in (7) the signer sends transactions which include: *parent signature's Id*, *certId*, message digest, and cryptographic signature. (8) Certificate is validated (using the revocation status and positive voters' count). (9) Signature-based validations are executed on the smart contract: If it is a serial signature (i.e. it has a valid *parentSignatureId*), that is *messageDigest* needs to be a hash of the parent

signature's *cryptographicSignature*. If it is not a serial signature there are two options: either the content is confidential, or it is not confidential and still uploaded to IPFS. If the *content* is uploaded to IPFS, it is checked whether the value in IPFS is the same as the message digest. If the signer does not upload the *content* to IPFS, it is impossible to match the hash of the *content* and *messageDigest*. As stated above, in order to empower the hash of *content's* cryptographic resistance and to protect *content* from manipulation, the *extraMessageDigest* should be provided. Hence, if the *content* is confidential, it is checked whether the *extraMessageDigest* value is provided and the size *extraMessageDigest* value is controlled.

The cryptographic signature is verified with the public key. Data to be signed value (DTBS) is calculated by hashing (SHA256) a concatenation of *messageDigest* and *certificateId* H (*messageDigest* —— *certId*) and cryptographic signature is verified with the public key. If all these validations are successful, the signature is accepted/logged.

5.2.2. Creating signature with transaction key:

As can be seen in the first step of Figure 5.1, the signer optionally uploads the *content* to IPFS or calculates the *extraMessageDigest* value with Keccak-256. In the second step, the signer sends a transaction for signing using inputs of: *messageDigest*, *certificateId*, and *signatureProfile*. In the third step, smart contract-based validations are run, which utilize SemiDec-PKI for certificate validation: If the content is attached to IPFS, the hash is compared with the IPFS address; if not, the existence of the extra message digest is controlled. This confirms that the address initiating the transaction is the same as the certificate address. It also validates that the cryptographic signature value is empty. If these validations are passed, the BlAdES is created successfully. In the case of a serial signature, IPFS content or the extra message digest are not needed, but the parent's signature status is required.

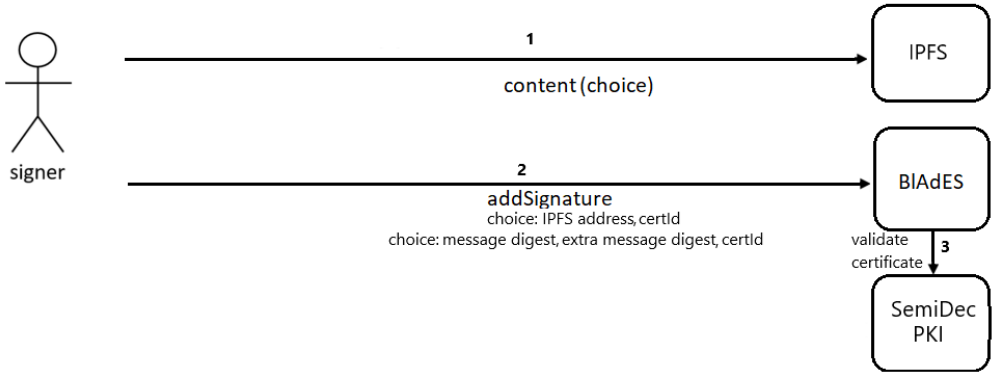

Figure 5.1 Signing process with transaction key

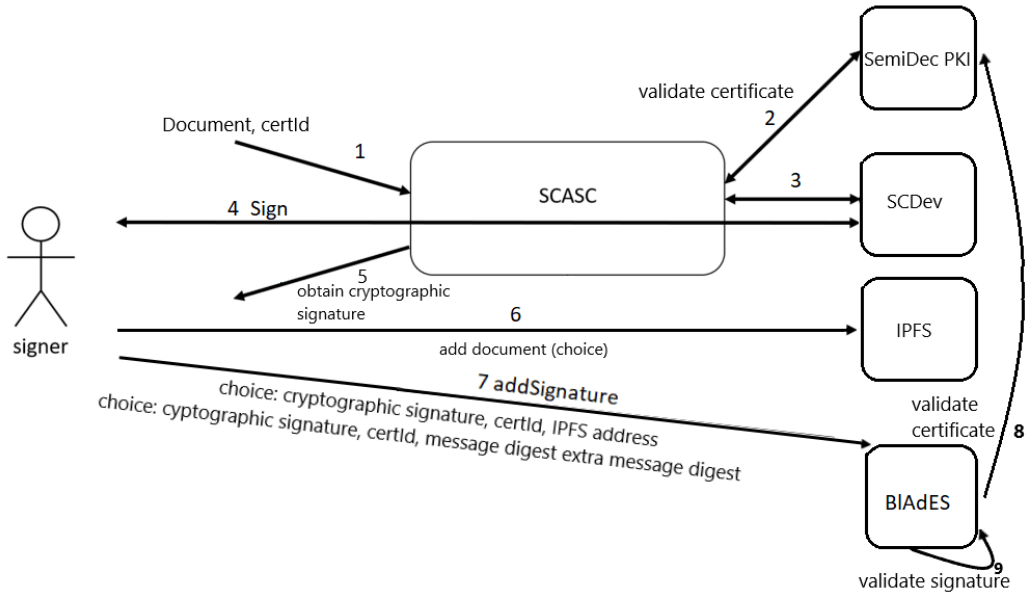

Figure 5.2 Signing process with conventional private key

5.3. Signature Verification and Preservation

The signature verification process is where the signature is sent to one party, who checks the existence and accuracy of the signature. The party verifying the signature is expected to be a light node. If the document is uploaded to IPFS, only the *signatureId* information is deemed sufficient; in which case the following verifications are conducted: (1) Verifier obtains BlAdES and content via SC. (2) Verifier verifies Merkle roots using Merkle proof. (3) If the *signatureStatus* value is true and the identity information is consistent with the subject in the certificate, the signature is accepted.

If the document has not been uploaded to IPFS, the signer transmits the document to the verifier next to the *signatureId*, after which the following verifications are conducted: (1) Verifier obtains BlAdES and extra message digest via SC. (2) Merkle proof is run. (3) Verifier calculates and compares the SHA256 and Keccak-256 hash of the document accompanying the signature. They move on to the next step if the hash values match. (4) The signature is considered as verified if the signature status value is true and the identity information is consistent with the subject in the certificate.

If the signature is a serial, parent signatures need to be verified recursively from the first signature. The first signature can be verified according to the options (with/without document), as previously explained.

Signature preservation is provided thanks to the immutable structure of the blockchain and the prescribed security measures are automatically invoked for confidential documents.

5.4. Security and Performance

Leveraging the blockchain technology, BlAdES addresses the challenges associated with the temporal integrity and security of electronic signatures over extended periods. This innovative format ensures the longevity of electronic signatures by embedding them within a blockchain, providing an immutable and transparent ledger. BlAdES not only enhances the tamper resistance of electronic signatures but also introduces a future-proof solution for

evolving technological landscapes. The use of blockchain ensures that the integrity and authenticity of signed documents can be reliably verified over time, making BlAdES a robust and sustainable option for long-term electronic signatures in various domains, from legal and financial sectors to broader applications in secure digital transactions. In this section, the applicability of the designed system from security and performance aspects are discussed.

5.4.1. Security

Threats to the proposed design are investigated under five groups: Threats against the infrastructure; cryptographic threats; blockchain-based threats, signature-based and content-based threats.

Signature-based Threats In conventional PKI, users may create signatures offline without any log. Hence, people may not notice if an attacker creates a signature using his/her private key. BlAdES, on the other hand, allows people to monitor their signatures thanks to its blockchain-based approach. In addition, since all signatures are logged to the blockchain, it uses the time stamp of the blockchain instead of a time stamp and does not need an external time stamp. In the current system, since the signatures are created offline, their verification depends on timestamps, revocation data, and certificate safety. The archive timestamp protects the signature, the certificates, and the revocation data. The archiving process needs to renew with the new timestamp to preserve the revocation data, certificate data, and previous archive timestamps. In BlAdES, it does not require extra archiving in case the root, subordinate, or OCSP certificates are revoked or expire after the signature is created.Hence, it eliminates the continuous process of renewing the archive timestamp.

Content-based Threats In the scenario where the content is confidential and not included in the blockchain, two different digest algorithms eliminate the risk of content manipulation if the content's digest algorithm loses its safety.

5.4.2. Performance

End-users are light nodes, and voters and auditors are defined as full nodes in the proposed infrastructure. The PoW algorithm is utilized in the current Ethereum system. However, the increased number of transactions (Txn) and the subsequent rise in Ethereum prices would lead to an increase in the total price for transactions and deploying SCs. Therefore, this consensus algorithm is planned to be changed to Proof of Stake (PoS) in Ethereum 2.0. Hence, it will be able to support more transactions and the use of SCs. Moreover, it is expected to decrease energy consumption.

Here, the storage cost of the proposed infrastructure based on blockchain is calculated as in [2]. There are about 3.64×10^8 registered domain names (as of 2021 [30]) and 46 million websites using SSL certificates [110]. Therefore, in a worst-case scenario, let us consider that half of all registered domains are assumed to use SSL certificates [111]; the storage cost is calculated as follows, based on certain assumptions:

- It is not possible to determine number of signature in the world, so we try to use certificate data to make a guess.
- We assume total number of all other certificate types (code signing, e-signature, e-seal) equals the number of SSL certificates.
- Redefined certificates in this proposal would be approximately 256 bytes, or half of the average X509 certificate size (512 bytes) given in CertLedger [2].
- SSL certificate maximum lifetime period is defined as 1 year (398 days) in a recent CAB Forum Ballot [49]. For the sake of simplicity, the worst case scenario is adopted for all other certificate types, and the certificate lifetime is defined as 1 year.
- PoS is used as the consensus algorithm in Ethereum. In PoS, the average block time, which is the time it takes to generate a new block, is 12-14 seconds [89].
- Adding new certificates is expected to cover the majority of transactions (Txn) in SemiDec-PKI, so the costs of all other transactions are ignored here.

- Signature size is 181 bytes.

We assume that signatures are created evenly throughout the year, and the block time is 12 seconds [2]. Therefore, five blocks are generated every 60 seconds. Consequently, the total number of blocks generated in one year is 26,280,000 ($365 \times 24 \times 60 \times 5$). The number of transactions (Txn) is equal to the total number of certificates (3.64×10^8) divided by the number of blocks generated annually, as described in CertLedger [2], shown as follows:

$$\text{Number of Txn} := \frac{\text{Number of Signatures}}{\text{Annually Generated Blocks}} \tag{7}$$

The size of a block (BS) is given by:

$$\begin{aligned} \text{BS} := \text{Txn Size} \times \text{Number of Transactions} \\ + \text{Header} \end{aligned} \tag{8}$$

$$\text{Txn Size} := \text{Message} + \text{Signature} \tag{9}$$

$$\text{Message} := \text{PKSender} + \text{Receiver} + \text{Data} \tag{10}$$

where PKSender, Receiver, Data, and Signature refer to the size of the sender's public key (64B), the receiver's address (20B), the data payload, and the signature (˜256B), respectively. Hence, the transaction size

Hence, the blockchain size of a full node is approximately 150 GB, as given in Equation 5. Since the cost of 1 GB of disk storage is about 0.02 USD [113], the combined cost for all certificate transactions would be approximately 3 USD per annum. On the other hand, light nodes do not store the entire block, just the header (508 bytes per block). As such, the total blockchain size of a light node per year would be approximately 640 MB ($26,280,00 \times 508$).

In the literature, certificates are logged to the blockchain and sent to the client in X509 format. In this study, the log in the blockchain was used without sending a separate certificate in the X509 standard. While the X509 certificate size is 512B on average [2], our certificates are approximately 256B.

Certificate issuance time is not measurable because registration authorities checks the appliance documents and gives votes according to validation result. But in the certificate transparency which in usage on SSL when a certificate is submitted to a log successfully, the server sends a Signed Certificate Timestamp (SCT) as proof and promise to add the certificate in the Merkle Tree within a fixed amount of time known as the Maximum Merge Delay (MMD) [70]. MMD is usually 24 hours [114]. While the voting mechanism ensures cross-checking, TSP croos-check durations are expected to not cause performance issues if they are reasonable.

Signature (256B) and certificate sizes (256B) are similar in BLAdES, hence the calculation cost of electronic signatures will be similar. Since electronic signatures are not logged anywhere in the current system, it is impossible to reach the number of electronic signatures created worldwide. However, supposing that certificates used to create the electronic signature are a small portion of the digital certificates, BlAdES can be adaptable to the electronic signature ecosystem.

5.5. Comparison with Related Studies

Blockchain-based Advanced Electronic Signature is a novel work based on blockchain. Therefore, it will be compared with the current Advanced Electronic Signature Formats, especially with CAdES [50] due to its advantages over other formats such as XAdES [52], PAdES [53]. The comparison results are discussed below and summarized in Table 5.2.

Signature Monitoring: The current system creates signatures with a smart card or using remote signing solutions. Since this process is in the local system of a user and not monitored, the signer may not know whether a signature was created against his will using his keys. BlAdES ensures that signatures are checked by SCs and logged, so the signer can check whether a signature has been signed on his behalf.

Requirement of External Timestamp: The creation date of the signature is essential, since it determines its lifetime. Timestamps are used to guarantee the creation date of the signature. Otherwise, the lifetime of the certificate is used for determining the lifetime of the signature.

Table 5.2 Comparison of CAdES and BlAdES

	CAdES	**BlAdES**
Signature Monitoring	No	Yes
Requirement of External Timestamp	Yes	No
Requirement of Additional Signature Verification Data	Yes	No
Requirement of Archiving	Yes	No
Long-term Signature Sizes with Certificate	13kb	512 bytes
Interoperability Problems in Signature Verification and Preservation	Yes	No

Signers who want to create a long-term signature must get a timestamp. TSPs both issue timestamps and keep records of them. Because BlAdES uses the blockchain's timestamp, users and TSPs are not burdened by providing timestamps as in the current system.

Requirement of Additional Signature Verification Data: In addition to timestamp, the revocation status at the time the signature created is essential for long-term verification of the signature. Hence, the end-user certificate and its issuing subordinate certificate are forwarded to the other party along with the signature and stored. In BlAdES, the blockchain proves the validity of the certificate when the signature is created without requiring to keep the revocation data.

Requirement of Archiving: A timestamp is obtained by concatenating signature, signed content, signature timestamp, and revocation data. BlAdES eliminates the need for archiving, since the creation of a signature is auditable, the revocation data and timestamp can be tracked from the blockchain.

Size of Long-term Signatures: In the current system, the size of a signature may vary depending on its type and standard. Here, a Detached Archive CAdES signature is taken for comparison. Hence, the archive size that includes features, end-user certificate, timestamp certificate, subordinate certificate, revocation data (OCSP), and archive timestamp, is equal

to 13Kb. BlAdES reduces this size to 512B.

Interoperability Issues in Signature Verification and Preservation: Signature verification can be done locally as well as through signature verification services. Since there are many APIs and their implementation changes in new releases, there are interoperability issues. However, in BlAdES, the SC guarantees that the signature is created in the valid structure.

5.6. Economic Viability

This chapter discusses the creation of a commercial model utilizing smart contracts to enhance the efficiency and security of certificate issuance and management. The model incorporates Certificate Authorities (CAs) and customers, with an embedded incentivize-disincentivize mechanism within the smart contracts. Additionally, the chapter addresses the challenge posed by high Ethereum transaction fees and explores how Ethereum Virtual Machine (EVM) compatible platforms can be leveraged for issuing certificates and creating Blockchain-based Advanced Electronic Signatures (BlAdES).

5.6.1. Smart Contracts in Certificate Issuance

The core of our commercial model revolves around the use of smart contracts, which are self-executing contracts with the terms of the agreement directly written into code. These smart contracts automate and streamline the processes of certificate issuance, renewal, and revocation, ensuring transparency and reducing the potential for human error.

5.6.2. Roles and Interactions

- **Certificate Authorities (CAs)**: These entities are responsible for issuing and managing digital certificates. They interact with the smart contract to register, issue, renew, or revoke certificates.
- **Customers**: Individuals or organizations requiring digital certificates engage with the smart contract to request issuance or renewal of their certificates.

- **Activating Smart Contract**: In fist phase uploading of smart contract may be expanse by CA/B Forum which is a community is managed by CA and Browsers [49].

5.6.3. Incentivize-Disincentivize Mechanism

- **Incentives**: To ensure honest behavior and participation, CAs and customers are rewarded with tokens or discounts for timely and accurate operations. ERC tokens are used for incentivize third parties.
- **Disincentives**: Penalties are imposed for malicious behavior, delays, or incorrect handling of certificates, thus maintaining the integrity of the system.

5.6.4. Addressing Ethereum Transaction Fees

One of the significant challenges in implementing blockchain-based solutions is the high transaction fees associated with the Ethereum network. These fees, known as gas fees, can be prohibitive, particularly for commercial applications that require frequent transactions. Given the high price of Ethereum, this issue not only affects the issuance of certificates and the creation of BlAdES signatures but also impacts all operations executed within smart contracts. To mitigate these costs, EVM platforms are utilized. In the certificate ecosystem, where there is substantial revenue, these expenses can be more manageable on an EVM platform.

To mitigate the impact of high Ethereum fees, we propose leveraging EVM-compatible platforms. These platforms support the same smart contract functionality as Ethereum but offer lower transaction costs. Examples include:

- **Binance Smart Chain (BSC) [115]**: Provides a similar environment to Ethereum but with significantly lower fees.
- **Polygon (Matic) [116]**: An Ethereum layer-2 scaling solution that reduces transaction costs and increases throughput.

- **Avalanche [117]**: A highly scalable blockchain platform that supports EVM-compatible smart contracts.

By utilizing these alternative platforms, we can maintain the security and functionality of Ethereum while reducing operational costs.

5.6.5. Discussion

The proposed commercial model using smart contracts offers a robust solution for managing digital certificates and signatures. By incorporating an incentives-disincentives mechanism, we ensure the integrity and reliability of the system. Addressing the high transaction fees of Ethereum by leveraging EVM-compatible platforms allows us to maintain cost efficiency while utilizing the advanced features of blockchain technology. This model not only enhances the security and transparency of digital transactions but also paves the way for scalable and collaborative electronic signature solutions.

6. INTEGRATION of THRESHOLD SIGNATURE SCHEME AND MULTI SIGNATURE INTO BlAdES

BlAdES supports multiple signatures through the binding with a *parent signature*, which can lead to duplicate processes. In this chapter, we extend BlAdES by incorporating new methods that include threshold signatures and MuSig [67] to enhance security and efficiency. We start by detailing the integration of threshold signatures, explaining the underlying principles and the implementation process. Next, we explore the integration of MuSig, highlighting its unique features and the advantages it brings to the BlAdES method. Finally, we provide a comprehensive comparison of our new approach, demonstrating the improvements achieved with the inclusion of threshold signatures and MuSig.

6.1. Threshold Signature Scheme

As blockchain technology continues to evolve, securing smart contracts against breaches and unauthorized manipulations has become paramount. Smart contracts automate and enforce the execution of contract terms digitally within a blockchain, offering transparency, trustworthiness, and efficiency. However, their security largely depends on the underlying cryptographic mechanisms. This section explores the application of Threshold Signature Schemes (TSS) as a means to fortify the security and resilience of smart contracts.

Threshold Signature Schemes are a form of cryptographic protocol that distributes the power to generate a signature among multiple participants. A signature is only valid when a predefined subset (the threshold) of participants collaborates to create it. TSS enhances security by minimizing the risk of private key compromise since no single entity possesses the complete signing key [17, 18].

Integrating TSS with smart contracts introduces several benefits, pivotal among them being enhanced security, increased fault tolerance, and improved privacy. By dispersing the

authority to execute or alter smart contracts across multiple parties, TSS mitigates the risks of single points of failure and targeted attacks [17, 18].

Smart contracts, when coupled with TSS, require a consensus among designated parties to initiate transactions or make modifications. This requirement significantly hardens security since compromising the contract would necessitate breaching multiple participants simultaneously. Moreover, TSS obscures the decision-making process and specifics of the threshold setup from external entities, adding an additional layer of security through obscurity.

Fault tolerance within the realm of smart contracts refers to the system's ability to operate even if some participants are compromised or fail to act. With TSS, as long as the number of non-responsive or compromised parties does not exceed the threshold limit, the smart contract's functionality remains intact. This robustness ensures that smart contracts remain operational, fostering reliability and uninterrupted service.

TSS offers privacy benefits by ensuring that individual participants' actions or inactions in the signing process are not disclosed. This capability is particularly beneficial in scenarios where maintaining the anonymity of decision-makers or participants is crucial.

While the benefits of incorporating TSS into smart contracts are clear, several implementation challenges must be addressed:

Key Generation and Distribution: The process of securely generating and distributing key shares among participants is complex and requires meticulous planning and execution [17, 18].

Infrastructure Requirements: Implementing TSS demands additional infrastructure for secure communication and computation, which can introduce latency and cost considerations [17, 18].

Smart Contract Adaptation: Adapting existing smart contract architectures to incorporate TSS may require significant re-engineering, particularly for contracts not initially designed

with TSS in mind. In this chapter we propose to integrate TSS to BlAdES-based multiple signatures [17, 18].

Multiple signatures refer to the concept of having more than one digital signature on a single message, document, or transaction. In the context of cryptographic security and electronic transactions, the use of multiple signatures enhances various aspects such as security, reliability, and trust. Multiple signatures imply that the approval or authorization of a transaction involves the participation of multiple signers. Each signer contributes their unique digital signature to the data, collectively confirming their approval. Moreover, multiple signatures imply that the approval or authorization of a transaction involves the participation of multiple signers. Each signer contributes their unique digital signature to the data, collectively confirming their approval. In scenarios where multiple entities or parties are involved, multiple signatures contribute to the resilience of the system. Even if one or more signers are compromised or unavailable, the remaining valid signatures can still be used to verify the authenticity and integrity of the transaction. Each digital signature serves as proof of the origin and integrity of the data. The combination of multiple signatures provides stronger evidence of the authenticity of the transaction, as each signature is tied to the unique private key of an authorized signer. Multiple signatures can be applied in parallel, where all signers provide their approval simultaneously, or in a serial fashion, where signers contribute their signatures sequentially. The choice between parallel and serial approaches depends on the specific requirements of the use case [50, 52, 53].

In the evolving landscape of electronic signatures, the introduction of multiple signatures with a threshold signature mechanism within the Blockchain-based Advanced Electronic Signature (BlAdES) framework signifies a significant advancement. This chapter delves into the new approach of employing threshold signatures in the context of multiple signatures within BlAdES.

As we mentioned before, threshold signatures are cryptographic mechanisms that distribute the process of signing a message among multiple parties. Unlike traditional digital signatures, where a single party signs a message, threshold signatures require a predefined

number of parties to collaborate in order to produce a valid signature. This approach enhances security by mitigating the risk associated with a single point of compromise.

In this chapter firstly we shortly explain Blockchain-Enhanced Consensus Threshold Signature Scheme study [18], and we suggest an integration idea to adopt BlAdES-based multiple signature to BC-ECTSS. Finally, we criticize security and performance concerns for this integration.

6.1.1. Blockchain-Enhanced Consensus Threshold Signature Scheme (BC-ECTSS)

Huifang Yu et. al. designed the Blockchain-Enhanced Consensus Threshold Signature Scheme (BC-ECTSS), which is a sophisticated framework designed to bolster security and consensus mechanisms within a blockchain environment [18]. This scheme categorizes nodes into distinct roles, each critical for upholding the transactional integrity and security of the blockchain network. The scheme is underpinned by a set of algorithms that standardize its functionality, ensuring a secure and reliable blockchain operation [18].

Node Definition: The BC-ECTSS delineates four pivotal roles within its operational paradigm [18]:

1. **Signature Group Nodes**: These are characterized by a collection of n nodes, each participating in the generation of a partial signature for transactions. Through polynomial interactions, these nodes compute secret shares and establish public-private key pairs, culminating in the formation of partial signatures [18].

2. **Signature Combiner**: This entity initiates the transaction, validating partial signatures from group nodes. Upon reaching a predefined threshold t of valid signatures, it aggregates them to form a complete signature, which is then broadcasted across the blockchain network [18].

3. **Blockchain Network Nodes**: Responsible for block creation, these nodes aggregate transactions into new blocks, solving cryptographic challenges to propagate the new block through the blockchain [18].

4. **Verification Nodes**: Capable of verifying the authenticity of aggregated signatures and Nonce values, these nodes play a crucial role in ensuring the integrity of the blockchain ledger [18].

6.1.2. Algorithm Definition

BC-ECTSS includes algorithms as follows. Its workflow is as shown in Fig. 6.1.

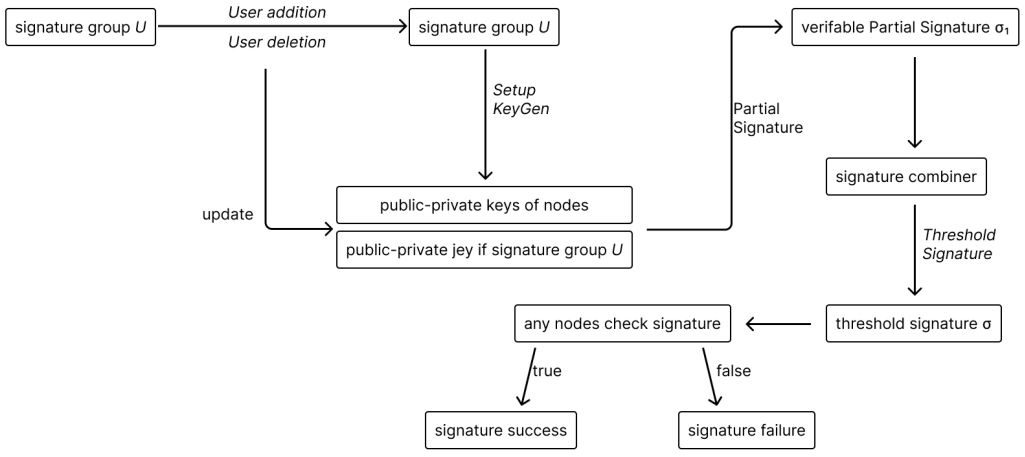

Figure 6.1 BC-ECTSS Workflow

- **Setup**: Initiates the system by setting up a parameter set ϕ, derivative of a security parameter 1γ [18].

- **KeyGen**: Utilizes ϕ to generate public-private key pairs for all participating nodes. Key generation is given in Fig. 6.2 [18].

- **Partial Signature**: Each node, upon receiving ϕ, its identity ID_i, and the message m, produces a partial signature σ_i [18].

- **Threshold Signature**: Validates and combines t partial signatures into a complete signature, given ϕ and the identity ID_c of the signature combiner. Signature creation workflow is given in Fig. 6.3. [18].
- **Verification**: Checks the validity of the complete threshold signature using ϕ, the receiver's identity ID_v, and the signature σ [18].
- **Node Addition and Deletion**: These algorithms manage the dynamic inclusion or removal of nodes within the signature group, updating public and private keys accordingly [18].

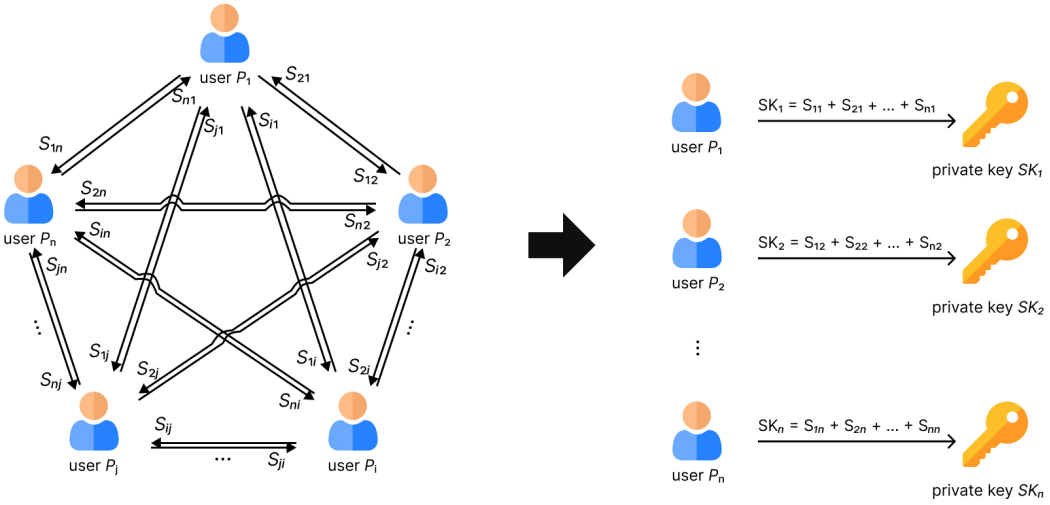

Figure 6.2 BC-ECTSS Key Gen Workflow

The BC-ECTSS [18] framework provides a comprehensive approach to enhancing the security architecture of blockchain systems. By integrating specialized node roles and clearly defined operational algorithms, it delivers a robust mechanism against forgery and unauthorized access, pivotal for the advancement of secure digital transactions in blockchain networks.

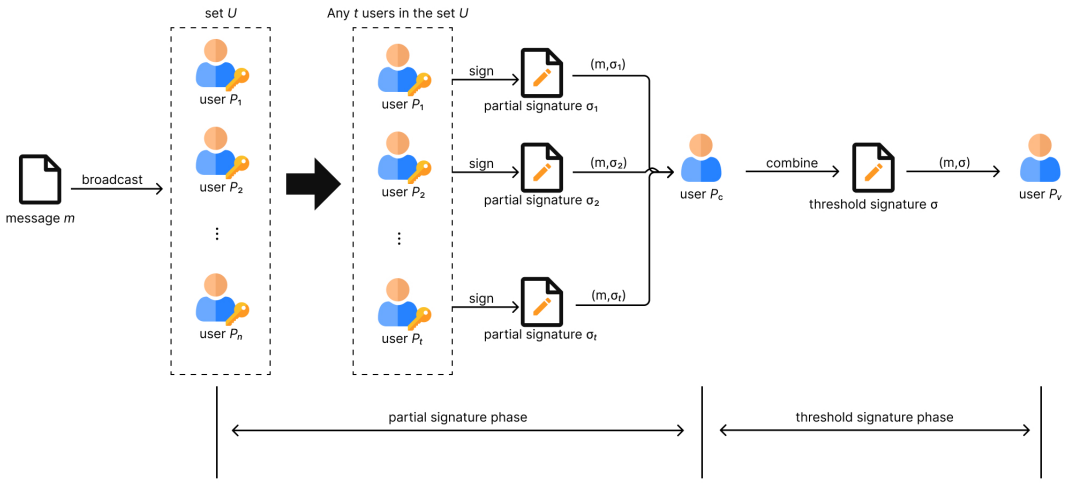

Figure 6.3 BC-ECTSS Signing Workflow

6.1.3. Integration of BC-ECTSS into BLAdES

The proposed system leverages Ethereum smart contracts to implement a robust and decentralized Threshold Signature mechanism. The smart contract, named BC-ECTSS-BlAdES, is designed to facilitate the creation and execution of jobs that require multiple signers to collectively contribute BlAdES signatures.

Within the framework of our smart contract, a suite of functions facilitates a seamless process encompassing task initiation, submission, and completion. This structured process is comparable to preparing a stage for a collective job. Users employ the function of threshold signature for the group of signers, specify the required signature threshold, and establish a collaborative atmosphere for the impending task.

The pseudo-code of the simplified algorithm is presented in Algorithm 3:

By incorporating robust authorization checks, real-time updates, and fine-grained permission management, the system not only strengthens security but also fosters collaboration and transparency among stakeholders. The decentralized nature of the network mitigates risks associated with single points of failure, underscoring the reliability and resilience of the system. This exploration highlights the integration of these features and their contributions

Algorithm 3: BC-ECTSS Adapted Collaborative Signing Workflow

Data: List of signatory nodes, Signature threshold requirement
Result: Blockchain transaction validated and executed

Signature Combiner Node proposes a new transaction
Calls the `proposeTransaction` function specifying `Signatory Nodes` and the required signature threshold
Smart Contract prepares Transaction
Initializes a Transaction structure with `Signatory Nodes` details and the defined threshold
Generates and stores a unique Transaction ID
foreach *Signatory Node in List of signatory nodes* **do**
    Smart Contract broadcasts transaction proposal
    Emits the `TransactionProposed` event with the Transaction ID and relevant details
foreach *Signatory Node in List of signatory nodes* **do**
    Signatory Node agrees to sign the transaction
    Receives a notification to contribute their signature
    Executes the `agreeToSign` function to participate in the signature process
    Signatory Node provides their partial signature
    Calls the `generatePartialSignature` function to submit their partial BC-ECTSS signature, associated with the Transaction ID
    The partial signature is recorded, pending aggregation
Smart Contract aggregates partial signatures
Continuously collects the submitted partial signatures
Once the number of signatures reaches or surpasses the signature threshold, initiates signature aggregation
Signature Aggregation
if *Signature Threshold is Met* **then**
    Combines the partial signatures to create a complete BC-ECTSS signature
    Verifies the complete signature in relation to the transaction
Smart Contract executes the transaction
- Upon successful signature validation, marks the transaction as executed
- Emits the `TransactionExecuted` event with the Transaction ID
Blockchain Update
Chronologically logs every action, from transaction proposal to execution, on the blockchain
Reinforces the blockchain's role as a secure and immutable record of the BC-ECTSS signing process
Stakeholder Notification
Updates all stakeholders about the transaction progress, signature milestones, and execution status
Finalization of the BC-ECTSS Collaborative Signing Workflow
Enables signatory nodes and stakeholders to validate the executed transaction and its collective signature on the blockchain

to a more secure and efficient digital signature workflow. In this steps some key features provided as:

- Authorization checks ensure that only valid signers can contribute, enhancing the overall security of the system.
- Once a signer submits a signature, the data is recorded on the blockchain, providing an immutable record of the transaction.
- The signing flow initiation improves the overall workflow by explicitly defining the signers and streamlining the process.
- Signers are brought into the flow collaboratively, fostering enhanced coordination during the signature collection process.
- The system efficiently checks for signature completion, reducing unnecessary computations and optimizing gas usage.
- The completion status is transparently recorded on the blockchain, providing visibility to all stakeholders.
- Stakeholders receive real-time updates on signature submissions, enhancing transparency
- The emitted events contribute to the auditability of the system, ensuring a verifiable history of actions.
- Modifiers enhance access control, allowing for fine-grained permission management.
- The system operates on a decentralized network, reducing the risk of a single point of failure.
- The Ethereum blockchain's security features contribute to the overall robustness of the system.

Key highlights of BLAdES-BC-ECTSS

In this subsection the highlights of integrating TSS in BlAdES is given as follows:

- Distributed Signing Authority: TSS within BlAdES decentralizes the signing power, thereby substantially mitigating the risk of single-point failures or targeted attacks. This dispersion of authority ensures that no single entity holds complete control over the signing process, enhancing the overall system security.
- Flexible Threshold Settings: The Multi-BlAdES form supports customizable threshold settings, enabling a tailored configuration that aligns with specific use cases or organizational security policies. This adaptability allows for the striking of a balance between operational efficiency and security requirements.
- Elevated Security Measures: The integration between MuSig [67] and threshold signatures in BlAdES establishes a robust defense mechanism against unauthorized access and potential security breaches. This multifaceted approach significantly heightens the security threshold, safeguarding sensitive documents and process.
- Smart Contracts for Authority Management: The employment of Smart Contracts (SCs) for managing the distribution of signing authority leverages the inherent benefits of decentralized execution. This not only contributes to the security framework but also enhances transparency and trust in the threshold signature process.
- Dynamic Threshold Adjustment: BC-ECTSS-BlAdES introduces the capability to dynamically adjust signature thresholds based on contextual factors, such as perceived threat levels or specific operational contexts. This flexibility ensures that the system remains responsive and adaptable to evolving security requirements for document management.
- Collaborative Decision-Making: The collaborative essence of threshold signatures fosters secure and efficient multi-party collaborations. This framework is particularly beneficial for multiple signatures. It also ensures security and verifiability of signature.

- Scalability Support: By integrating threshold signatures, BlAdES accommodates scalability concerns, enabling seamless inclusion of additional participants within the signing process without compromising on security. This scalability is vital for systems anticipating growth or facing variable participant numbers.

Hence, BlAdES-BC-ECTSS signatures finds applications in scenarios where multi-party transactions require a collective authorization mechanism, ensuring security and consensus. Organizations can utilize BlAdES with threshold signatures for consensus-based approvals of critical documents, enhancing trust and transparency in multiple signature processes. BlAdES introduces the capability to apply multiple signatures to a single job. Each signer contributes to the threshold signature, ensuring that the signature is created only when the predefined number of signatures is reached.

6.1.4. Security and Performance Issues

Integrating Blockchain Threshold Signature Schemes (BC-ECTSS) into Blockchain-aided Decentralized Execution Systems (BlAdES) introduces an new approach to enhancing security and operational efficiency within document management frameworks. However, like any technological integration, it is accompanied by several security and performance considerations. This critique examines the challenges and benefits associated with the BC-ECTSS integration into BlAdES, focusing on key aspects such as key generation, distribution, and the implications on performance and optimization.

Security Issues:

New Key Generation: A fundamental requirement in BC-ECTSS is the generation of new keys for participants. While this is a cornerstone for enhancing security — by ensuring that keys are fresh and not compromised — it also introduces complexities. Each new session or transaction may necessitate the generation of new keys, posing logistical and computational challenges. The security benefit of having new keys for each transaction comes at the cost of increased overhead and the potential for delays in transaction initiation.

Key Distribution: The distribution of keys in a BC-ECTSS integrated BlAdES framework is another critical concern. With potentially numerous participants in a document management scenario, securely distributing keys to all parties without compromising them is a significant challenge. The process must ensure confidentiality and integrity, requiring robust encryption and secure channels for key transmission. Any vulnerability in this process could expose the system to risks of key interception or manipulation.

Sharing of Keys Among Many Users: In scenarios where document management involves numerous participants, the sharing of keys becomes even more problematic. Each participant must securely receive their portion of the key, necessitating a scalable and secure distribution mechanism. The complexity increases with the number of participants, escalating the risk of security breaches. Ensuring that each participant securely stores and manages their key share also becomes a concern, as any mishandling could compromise the entire system.

Signature-based Threats As we mentioned before, BC-ECTSS BlAdES also employs a blockchain-based approach, enabling users to monitor their signatures in real-time. Hence, security of digital signature format. Additionally, all signatures are logged on the blockchain, utilizing its timestamp, thus eliminating the need for an external timestamp. In conventional systems, the offline creation of signatures means their verification relies on timestamps, revocation data, and certificate safety. As we mention in background the archive timestamp secures the signature, certificates, and revocation data, requiring periodic renewal to maintain the integrity of the revocation data, certificate data, and previous archive timestamps. Similar to BlAdES; BC-ECTSS BlAdES, however, does not require additional archiving even if root, subordinate, or OCSP certificates are revoked or expire after the signature is created, due to the blockchain's immutable proof. This eliminates the need for the continuous renewal of archive timestamps.

Content-based Threats Similar to BlAdES, when the content is confidential and not included on the blockchain, BC-ECTSS BlAdES employs two different digest algorithms to mitigate the risk of content manipulation. This dual-algorithm approach ensures that if one digest algorithm is compromised, the other can still maintain the integrity of the content.

Performance Issues:

Optimization Through Signature Aggregation: One of the significant advantages of integrating BC-ECTSS into BlAdES is the optimization of signature processes. In traditional systems, the participation of (n) signers would typically result in (n) signatures, each needing to be verified independently. BC-ECTSS, however, allows for the aggregation of these signatures into a single signature representing the consent of all participants. This approach not only streamlines the verification process but also significantly reduces the computational overhead associated with handling multiple signatures, thus enhancing system performance.

Performance Overhead caused by Key Generation and Distribution: While the aggregation of signatures offers performance benefits, the initial steps of key generation and distribution introduce overheads that can negate some of these gains. The computational resources required to generate new keys for each transaction and the time needed to distribute these keys securely can lead to increased latency in transaction processing. In high-throughput environments, such as document management systems dealing with numerous transactions, this overhead could become a bottleneck, affecting the overall system performance.

Balancing Security and Performance: The integration of BC-ECTSS into BlAdES necessitates a careful balance between enhanced security measures and performance efficiency. The security benefits of using threshold signatures and the operational efficiencies gained through signature aggregation must be weighed against the performance overhead introduced by the key generation and distribution processes. Optimizing these aspects requires innovative solutions that minimize latency and computational requirements while maintaining the highest security standards.

BC-ECTSS and BlAdES Comparison

In the BlAdES design, each signature can have only one signer. Consequently, for multiple signatures, some attributes are duplicated. However, in the BC-ECTSS BlAdES design, multiple signers can be accommodated, optimizing performance.

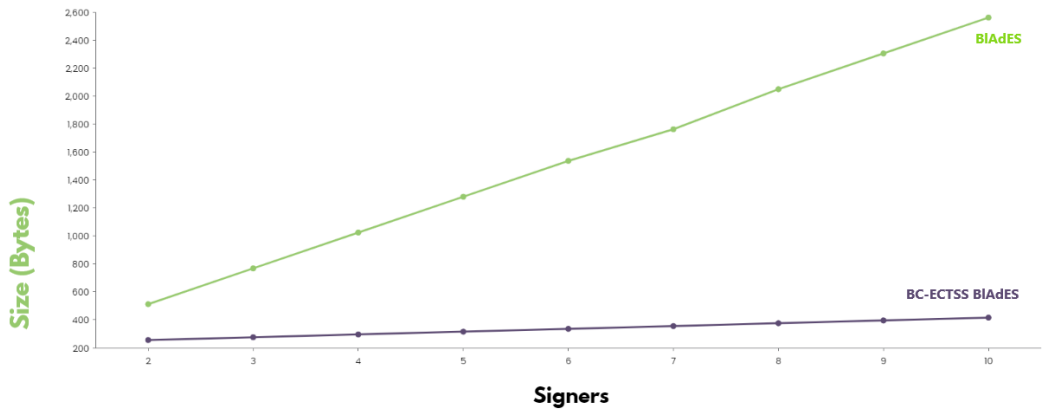

Figure 6.4 Comparison of Number of Signers and Signature Sizes Between BlAdES and BC-ECTSS BlAdES

In our new structure, the parent signature address is eliminated, and multiple signer addresses are introduced. As mentioned earlier, a signature length is 256 bytes without a certificate. For example, in the previous design, ten signers would require 256 bytes per signer, totaling 2560 bytes. In the new design, the parent signature ID is not used, and multiple signer addresses are added. For ten signers, this would result in $256\,\text{bytes} - 20 + (9 \times 20) = 416\,\text{bytes}$. Thus, for ten signers, the size decreases from 2560 bytes to 416 bytes. Figure 6.4 illustrates the signature sizes for 2 to 10 signers.

Performance Evaluation of BC-ECTSS

Our performance assessment based on the BC-ECTSS [18].

Table 6.1 which is adapted by BC-ECTSS [18] outlines the running time for cryptographic operations given on study of Yu et al. [18] as Intel(R) Core(TM) i5-6200U CPU @2.30Hz; the operating system is 64-bit windows 10 [18]. Subsequently, Table 6.2 which is adapted by BC-ECTSS paper [18] delves into the computational efficiency and delineates features of the scheme. Communication overheads, showcased in Table 6.3 which is adapted by BC-ECTSS paper [18] incorporate measures such as the size of private keys, public keys, and signature lengths, denoted respectively by $|F_p|$, the length of an element within the finite field F_p, and $|Z_q^*|$, signifying the length of an element within Z_q^*.

Adapting standard compression technologies, within an 80-bit security paradigm involving a finite field F_p with order q, outlined over an elliptic curve $y^2 = x^3 + ax + b \mod p$, where p spans 512 bits and q extends over 160 bits. The length of an element in F_p is compacted to 65 bits, while an element in Z_q^* is compressed to 20 bits. Consequently, the signature lengths for the contrasting schemes are as follows: 235 bytes for scheme [*19*], and 150 bytes each for schemes [*20*] and [*21*], whereas BC-ECTSS manifests a signature length of 105 bytes, computed as $(2|Z_q^*| + |F_p|)$ bits [18].

Table 6.1 Execution Times for Cryptographic Operations of BC-ECTSS

Operation	Symbol	Time (ms)
Exponential Operation	TEX	5.086
Scalar Multiplication	TM	0.002
Elliptic Curve Scalar Multiplication	TE	7.984
Inverse Operation	TI	2.011
Bilinear Pairing	TP	20.15

Table 6.2 Signature/Verification Times and Adaptability

Schemes	**Signature Time**	**Verification Time**	**Support for Adding and Deleting Nodes**	**Suitable for Blockchain**
BC-ECTSS	$6tTM + 3tTE$	$2TM + 2TE$	✓	✓

Table 6.3 Communication Overhead of BC-ECTSS

Scheme	Private Key Size	Public Key Size	Signature Length
BC-ECTSS	$\|Z_q^*\|$	$\|F_p\|$	$(2\|Z_q^*\| + \|F_p\|)$

6.2. MuSig Scheme

MuSig is a multi-signature scheme developed for the Bitcoin protocol [67] but can be applied to other blockchain-based systems. It extends Schnorr signatures to support secure and efficient multi-signature operations, whereby a set of multiple signers can collaboratively produce a single, compact signature that is both secure and indistinguishable from an ordinary Schnorr signature.

6.2.1. Schnorr Signatures and their Attack Vectors

Schnorr signatures offer several advantages, such as smaller on-chain size, faster validation, and improved privacy. They allow for combining multiple signatures into one through aggregation and support more complex spending policies. However, Schnorr signatures are not without their challenges.

Attack Vectors

6.2.1.1. Rogue-Key Attack This attack involves a participant creating a specifically constructed key to steal funds. It can be mitigated in simple multi-signature protocols through an enrollment procedure where keys sign themselves. For transactions involving multiple inputs, plain public-key security is required, avoiding the need for a setup phase.

6.2.1.2. Russell Attack Discovered by Russel O'Connor, this attack allows a party in a multi-party scheme to claim ownership of another participant's private key, thereby spending the other's funds.

Pieter Wuille addresses several of these issues, refining the Bellare-Neven (BN) scheme and implementing performance improvements for scalar multiplication, aiding in batch validation on the blockchain.

MuSig [67] is a multi-signature protocol that enables a group of signers, each possessing a private/public key pair, to produce a single unified signature on a message. Verification can then be publicly performed using just the message and the set of public keys from all signers.

6.2.2. Transformation into Multi-Signature

Traditional multi-signature schemes involve each signer creating a stand-alone signature for the message with their private key, then concatenating all individual signatures. However, this approach may not be practical due to the increased size of the multi-signature.

The core idea behind MuSig [67] is to transform a standard signature scheme into a multi-signature scheme while keeping the signature size independent of the number of signers and similar to the original scheme's size.

6.2.3. Key Aggregation in Multi-Signature Schemes

MuSig [67] is notable for incorporating support for both key aggregation and security in the plain public-key model.

Key Aggregation

Multi-signatures should resemble a single-key signature associated with an aggregated public key derived from the participants' public keys. This approach enhances privacy and performance, as verifiers only need the aggregated key instead of individual public keys.

Properties of an Effective Multi-Signature Scheme

- Must be secure in the plain public-key model.
- Must comply with the normal Schnorr equation.
- Must support Interactive Aggregate Signatures (IAS).
- Must support Non-Interactive Aggregate Signatures (NAS).
- Each signer can sign the same message.
- Each signer can sign their own message.

MuSig [67] meets these criteria, making it a robust key aggregation scheme for Schnorr signatures.

6.2.4. Overview of Multi-Signatures

The most obvious use case for multi-signatures recently has been Bitcoin. Multi-signatures serve as a more efficient replacement for multisig scripts, providing a combination of smaller on-chain footprints, faster validations, and better privacy.

A key aggregation scheme reduces the number of public keys per input to one. When sending coins, a user can send them to the aggregate of all involved keys rather than listing all of them in the script. This streamlines the process and leverages the benefits of MuSig, including across multiple inputs by having the verifier handle the key aggregation.

6.2.5. Integration Steps for BLADES

In this section, the integration steps for BlAdES are thoroughly detailed. The integration of MuSig into the BlAdES method involves a series of well-defined steps to ensure seamless functionality and optimal performance.

Initialization

- **Elliptic Curve Parameters:** Define ECC parameters and ensure all participants agree on these.
- **Key Generation:** Each participant generates their private and public keys using ECC.

Key Aggregation

- **Public Key Exchange:** Each signer publicizes their public key.
- **Aggregation:** Form the aggregated public key P using:

$$P = H(P_1, P_2, \ldots, P_m)$$

Commitment Phase

- **Nonce Generation:** Each signer generates a nonce R and the corresponding commitment:

$$R_i = r_i \cdot G$$

- **Commitments Exchange:** Commitments are exchanged among all signers.

Nonce Aggregation

- **Aggregate Commitments:** Combine individual commitments to form:

$$R = R_1 + R_2 + \cdots + R_m$$

Partial Signature Generation

- **Partial Signatures:** Signers compute their partial signatures for a message M:

$$s_i = r_i + H(R||P||M) \cdot x_i$$

Signature Aggregation

- **Final Signature:** Aggregate partial signatures to form:

$$s = s_1 + s_2 + \cdots + s_m$$

- **Result:** The final signature (R, s) is verified using the aggregated public key P.

6.3. Comparison

In the realm of digital signatures and cryptographic security, threshold signature schemes and multi-signature (MuSig) [67] protocols offer robust solutions to enhance security and decentralization. This chapter compares the adaptation of these schemes to the Blockchain-based Advanced Electronic Signature (BlAdES) framework. By evaluating the strengths and limitations of each approach, we aim to highlight the most effective integration for BlAdES to meet the demands of secure and scalable electronic transactions.

6.3.1. Threshold Signature Scheme

A threshold signature scheme allows a group of participants to collectively generate a signature. A subset of these participants, meeting a predefined threshold, is sufficient to produce a valid signature. This decentralization of signing authority enhances security by eliminating single points of failure.

Key Features and Benefits

Decentralization: By distributing the signing authority among multiple participants, threshold signatures reduce the risk associated with compromising a single entity. Fault Tolerance: The system remains operational even if some participants are unavailable or compromised, provided the threshold number of participants can still generate a signature. Enhanced Security: Threshold signatures are resistant to certain types of attacks, such as those targeting individual private keys. Integration with BlAdES Adapting a threshold signature scheme to BlAdES involves the following steps:

Setup: Participants generate their individual key shares and a common public key. Signature Generation: When a document needs to be signed, a subset of participants collaborates to generate a partial signature. These partial signatures are then combined to form the final signature. Verification: The signature can be verified using the common public key, ensuring that the document has been signed by the required subset of participants.

Challenges Complexity: Implementing threshold signatures can be complex, requiring careful management of key shares and coordination among participants. Performance: The process of generating and combining partial signatures may introduce additional computational overhead.

6.3.2. MuSig (Multi-Signature)

MuSig [67] is a specific protocol for multi-signature schemes, designed to be efficient and secure. It allows multiple participants to collaboratively sign a document, producing a single compact signature.

Key Features and Benefits Efficiency: MuSig is designed to be efficient, producing compact signatures that are easy to verify. Scalability: The protocol supports a large number of participants without a significant increase in signature size or verification complexity. Security: MuSig offers strong security guarantees, ensuring that signatures cannot be forged without the cooperation of the required participants.

Integrating MuSig with BlAdES involves:

Key Aggregation: Participants generate their key pairs and aggregate their public keys to form a combined public key. Signature Generation: Participants generate their individual signature shares. These shares are then aggregated to produce a single, compact multi-signature. Verification: The multi-signature is verified against the aggregated public key, confirming the participation of all required signers. Challenges Coordination: Like threshold signatures, MuSig requires coordination among participants to aggregate keys and signatures. Implementation Complexity: While efficient, MuSig protocols can be complex to implement correctly, requiring careful handling of cryptographic primitives. Comparative Analysis When comparing the adaptation of threshold signature schemes and MuSig to BlAdES, several factors must be considered:

Security: Both approaches offer enhanced security by decentralizing signing authority. However, threshold signatures may provide greater fault tolerance in environments where participant availability is uncertain.

Efficiency: MuSig is generally more efficient, producing compact signatures that are easier to manage and verify.

Complexity: Both schemes introduce additional complexity compared to single-signature schemes, but MuSig may be simpler to implement in terms of signature aggregation and verification. **Scalability:** MuSig supports a larger number of participants without a significant impact on performance, making it more suitable for large-scale applications.

Table 6.4 Comparison of Threshold Signature Scheme and MuSig for BlAdES

Criteria	**Threshold Signature Scheme**	**MuSig**
Decentralization	High	High
Fault Tolerance	Higher	Lower
Security	Strong	Strong
Efficiency	Moderate	High
Scalability	Moderate	High
Signature Size	Larger	Compact
Verification Complexity	Moderate	Low
Implementation Complexity	High	Moderate
Coordination Required	High	High
Key Aggregation	Requires New Key Pairs	Aggregates Existing Keys
Use Case Suitability	Fault Tolerance, High Security	Large Scale, High Efficiency

6.3.3. Discussion About Future Work

When comparing the adaptation of threshold signature schemes and MuSig to the BlAdES framework, several key factors emerge:

Security: Both threshold signature schemes and MuSig offer strong security by decentralizing the signing authority. This reduces the risk of compromising a single entity and ensures the integrity of the signatures.

Fault Tolerance: Threshold signature schemes provide higher fault tolerance, making them suitable for environments where participant availability might be uncertain. This robustness ensures that the system remains operational even if some participants are unavailable or compromised.

Efficiency: MuSig outperforms threshold signature schemes in terms of efficiency. MuSig produces compact signatures that are easier to manage and verify, resulting in lower computational overhead and faster processing times.

Scalability: MuSig is more scalable compared to threshold signature schemes. It supports a larger number of participants without significantly increasing the complexity or size of the signatures, making it ideal for large-scale applications.

Signature Size: MuSig generates compact signatures, which are easier to store and transmit compared to the larger signatures produced by threshold schemes.

Verification Complexity: MuSig offers lower verification complexity, allowing for faster and more straightforward signature verification processes.

Implementation Complexity: Implementing threshold signature schemes is generally more complex due to the need for generating new key pairs and coordinating partial signatures. MuSig, while still requiring coordination, aggregates existing keys and simplifies the overall process.

Key Aggregation: Threshold signature schemes require the creation of new key pairs for each participant, which can add to the implementation complexity. In contrast, MuSig aggregates existing keys, making it more user-friendly and easier to integrate.

Coordination: Both schemes require coordination among participants, but MuSig's aggregation of existing keys simplifies the coordination process compared to the management of new key pairs in threshold schemes.

Use Case Suitability: Threshold signature schemes are better suited for scenarios where high fault tolerance and security are paramount. MuSig, with its efficiency and scalability, is more suitable for large-scale applications requiring quick and efficient signature verification.

Given the comparative advantages, MuSig emerges as the more usable and efficient option for integrating with BlAdES. Its ability to aggregate existing keys and produce compact, easily verifiable signatures aligns well with the dynamic needs of modern digital transactions. As part of future work, we will focus on implementing MuSig within the BlAdES framework. This integration will aim to leverage MuSig's efficiency and scalability to create a robust and scalable solution for secure electronic transactions, ensuring that BlAdES remains at the forefront of digital signature technology.

7. CONCLUSION

Our study introduces a novel and robust Public Key Infrastructure (PKI) solution known as SemiDec-PKI, which leverages blockchain technology and smart contract-based mechanisms to address long-standing security challenges in the conventional PKI model. By combining elements of Web of Trust and centralized approaches, SemiDec-PKI significantly reduces the risks associated with single points of failure and impersonation attacks, thus enhancing the overall security of certificate issuance. One of the key innovations of SemiDec-PKI is the implementation of a stake-based reward-punishment mechanism, which ensures that stakeholders have a vested interest in maintaining the system's integrity. This mechanism not only incentivizes honest participation but also penalizes malicious actors, deterring them from attempting to manipulate the system. Moreover, the system's voting scheme provides a collaborative approach to certificate issuance and validation, involving multiple Certificate Authorities (CAs) and Supervisor Bodies (SBs). This approach not only increases security through cross-checking but also ensures that the PKI system can operate independently, eliminating the need for centralized control. Another noteworthy feature is the decentralized certificate revocation mechanism, which empowers certificate owners to initiate revocation independently through smart contracts. This decentralization eliminates the revocation monopoly that exists in traditional PKI systems, enhancing system responsiveness and trustworthiness. Moreover, SemiDec-PKI is designed to be highly adaptable, allowing for the adjustment of the voting threshold value to strike a balance between security and efficiency. This flexibility ensures that the system can evolve to meet changing security requirements.

To sum up, SemiDec-PKI presents a unique approach to the PKI architecture, leveraging blockchain and smart contracts to create a secure, adaptable, and decentralized system. By addressing critical issues such as single points of failure, impersonation attacks, and revocation monopolies, SemiDec-PKI contributes to the advancement of secure digital communication and transaction environments. This study underscores the suitability of Ethereum's smart contract capabilities for implementing such a system and paves the way for

future research and development in the field of blockchain-based PKI solutions. Moreover, it relies on the proposed stake-reward-punishment mechanism in order to prevent TSPs and SBs from being compromised.

Moreover, The Blockchain-based Advanced Electronic Signature (BlAdES) marks a transformative leap forward in the domain of electronic signatures, presenting a sophisticated framework that seamlessly integrates blockchain technology and cryptographic advancements. This conclusion reflects on BlAdES' key contributions and its significance in addressing contemporary challenges in the digital signature landscape, emphasizing its alignment with the evolving demands of the digital era. BlAdES is foundational in fortifying the security and resilience of electronic transactions. The incorporation of blockchain technology ensures an immutable and transparent ledger, safeguarding the integrity of signatures and transactions against tampering or fraud. The integration of threshold signatures further enhances security by distributing signing authority across multiple entities, mitigating the risks associated with a singular point of compromise. BlAdES emerges as a pioneering force, propelling electronic signatures into a new era. Its fusion of blockchain, threshold signatures, and innovative features positions it as a comprehensive solution for secure, collaborative, and scalable electronic transactions. As organizations navigate the intricacies of digital transactions, BlAdES serves as a testament to the transformative potential of blockchain technology in advancing the reliability, security, and efficiency of electronic signatures. With its diverse applications and forward-thinking design, BlAdES sets the stage for a future where electronic transactions are not only secure but also adaptable to the dynamic landscape of the digital age. This research marks a significant contribution to the academic discourse on electronic signatures, offering insights and solutions that resonate with the challenges and opportunities of the contemporary digital landscape.

The integration of multiple signatures with a threshold signature mechanism in the BlAdES framework marks a significant step towards advancing the security, resilience, and flexibility of electronic signatures. The integration of BC-ECTSS [18] into BlAdES represents a promising approach to secure and efficient document management in decentralized systems. However, it is imperative to address the accompanying security and performance

challenges proactively. Innovations in key generation, secure key distribution protocols, and optimizations in the signature aggregation process are critical for harnessing the full potential of this integration. Future research should focus on developing lightweight, secure key management strategies and enhancing the efficiency of the BC-ECTSS framework to ensure it meets the demanding requirements of modern document management scenarios.

REFERENCES

[1] Alexander Yakubov, Wazen M. Shbair, Anders Wallbom, David Sanda, and Radu State. A blockchain-based pki management framework. In *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, pages 1–6. **2018**. doi:10.1109/NOMS.2018.8406325.

[2] Murat Kubilay, Mehmet Sabir Kiraz, and Haci Mantar. Certledger: A new pki model with certificate transparency based on blockchain. *Computers & Security*, 85, **2019**. doi:10.1016/j.cose.2019.05.013.

[3] Mustafa Al-Bassam. Scpki: A smart contract-based pki and identity system. *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts*, **2017**.

[4] Chong-Gee Koa, Swee-Huay Heng, and Ji-Jian Chin. Etherst: Ethereum-based public key infrastructure identity management with a reward-and-punishment mechanism. *Symmetry*, 13:1640, **2021**. doi:10.3390/sym13091640.

[5] Abba Garba, Qinwen Hu, Zhong Chen, and Muhammad Rizwan Asghar. Bb-pki: Blockchain-based public key infrastructure certificate management. In *2020 IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International Conference on Smart City; IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pages 824–829. **2020**. doi:10.1109/HPCC-SmartCity-DSS50907.2020.00108.

[6] Murat Kubilay, Mehmet Sabir Kiraz, and Haci Mantar. Korgan: An efficient pki architecture based on pbft through dynamic threshold signatures. *The Computer Journal*, 64, **2020**. doi:10.1093/comjnl/bxaa081.

[7] Gwan-Hwan Hwang, Tao-Ku Chang, and Hung-Wen Chiang. A semidecentralized pki system based on public blockchains with automatic

indemnification mechanism. *Security and Communication Networks*, 2021:1–15, **2021**. doi:10.1155/2021/7400466.

[8] Iulian Aciobanitei, Vlad Dedita, Mihai-Lica Pura, and Victor-Valeriu Patriciu. Sabres - a proof of concept for enhanced cloud qualified electronic signatures. In *2020 13th International Conference on Communications (COMM)*, pages 103–108. **2020**. doi:10.1109/COMM48946.2020.9141954.

[9] Tiffany Hyun-Jin Kim, Lin-Shung Huang, Adrian Perrig, Collin Jackson, and Virgil Gligor. Accountable key infrastructure (aki): a proposal for a public-key validation infrastructure. In *Proceedings of the 22nd International Conference on World Wide Web*, WWW '13, page 679–690. Association for Computing Machinery, New York, NY, USA, **2013**. ISBN 9781450320351. doi:10.1145/2488388.2488448.

[10] David Basin, Cas Cremers, Tiffany Hyun-Jin Kim, Adrian Perrig, Ralf Sasse, and Pawel Szalachowski. ARPKI: Attack resilient public-key infrastructure. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pages 382–393. **2014**. ISBN 978-1-4503-2957-6. doi:10.1145/2660267.2660298.

[11] Andreas Loibl. Namecoin. Seminar report, Technische Universität München, Munich, Germany, **2014**. Seminar Innovative Internettechnologien und Mobilkommunikation.

[12] Louise Axon and Michael Goldsmith. Pb-pki: A privacy-aware blockchain-based pki. In *International Conference on Security and Cryptography*. **2017**.

[13] Jing Chen, Shixiong Yao, Quan Yuan, Kun He, Shouling Ji, and Ruiying Du. Certchain: Public and efficient certificate audit based on blockchain for tls connections. In *IEEE INFOCOM 2018 - IEEE Conference on Computer*

Communications, pages 2060–2068. **2018**. doi:10.1109/INFOCOM.2018.8486344.

[14] Weibiao Liang, Lin You, and Gengran Hu. Lrs pki: A novel blockchain-based pki framework using linkable ring signatures. *Computer Networks*, 237:110043, **2023**. ISSN 1389-1286. doi:https://doi.org/10.1016/j.comnet.2023.110043.

[15] Pawel Szalachowski, Stephanos Matsumoto, and Adrian Perrig. Policert: Secure and flexible tls certificate management. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, page 406–417. Association for Computing Machinery, New York, NY, USA, **2014**. ISBN 9781450329576. doi:10.1145/2660267.2660355.

[16] Salabat Khan, Zijian Zhang, Liehuang Zhu, Meng Li, Qamas Gul, and Xiaobing Chen. Accountable and transparent tls certificate management: An alternate public-key infrastructure with verifiable trusted parties. *Security and Communication Networks*, 2018:1–16, **2018**. doi:10.1155/2018/8527010.

[17] Josep-Lluis Ferrer-Gomila and M. Francisca Hinarejos. A multi-party contract signing solution based on blockchain. *Electronics*, 10(12), **2021**. ISSN 2079-9292. doi:10.3390/electronics10121457.

[18] Huifang Yu and Han Wang. Elliptic curve threshold signature scheme for blockchain. *Journal of Information Security and Applications*, 70:103345, **2022**. ISSN 2214-2126. doi:https://doi.org/10.1016/j.jisa.2022.103345.

[19] Carlisle Adams and Steven Lloyd. An introduction to Public Key Infrastructure (PKI). *Computer Communications*, 22(17):1646–1653, **1999**.

[20] Niels Ferguson, Bruce Schneier, and Tadayoshi Kohno. *Cryptography Engineering: Design Principles and Practical Applications*. Wiley, **2010**.

[21] Carl M Ellison and Bruce Schneier. Ten risks of pki: what you're not being told about public key infrastructure. *Computer Security Journal*, 16(1):1–7, **2000**.

[22] David W Chadwick. From trusted third parties to PKI and identity federation. *IFIP/SEC*, pages 41–54, **2002**.

[23] John A. Smith. Understanding certification authorities in public key infrastructure. *Journal of Cybersecurity*, **2020**.

[24] Emily R. Brown. Security and trust in online transactions: A certification authority perspective. In *International Conference on Computer Security*. **2018**.

[25] Ravi S. Patel. The role of certification authorities in digital certificate management. *Journal of Information Security*, **2019**.

[26] Maria L. Garcia. Ensuring trustworthiness: Certification authorities and the future of digital security. In *Annual Symposium on Information Assurance*. **2021**.

[27] Sung H. Kim. A survey of certification authority practices and challenges. *Journal of Computer Security*, **2017**.

[28] Liang Chen. Trust in the digital age: Certification authorities and their impact on e-commerce. In *International Conference on Cybersecurity and Privacy*. **2016**.

[29] Hui Wang. An overview of public key infrastructure and certification authority models. *Journal of Computer Networks and Security*, **2018**.

[30] Michael J. Turner. Evaluating the effectiveness of certification authorities in ensuring web security. In *ACM Conference on Computer and Communications Security*. **2019**.

[31] Xin Liu. Key management challenges in public key infrastructures: A certification authority perspective. *IEEE Transactions on Information Forensics and Security*, **2020**.

[32] Sarah E. Davis. Strengthening cybersecurity through effective certification authority policies. In *International Symposium on Secure Systems*. **2021**.

[33] Nicole van der Meulen. Diginotar: Dissecting the first dutch digital disaster. *Journal of Strategic Security*, 6(2):46–58, **2013**. ISSN 19440464, 19440472.

[34] Rob Wright. 23,000 symantec certificates revoked following leak of private keys. https://www.techtarget.com/searchsecurity/news/252436120/23000-Symantec-certificates-revoked-following-leak-of-private-keys. Accessed: 2022-11-10.

[35] Adobe. Adobe approved trusted list website. https://helpx.adobe.com/acrobat/kb/approved-trust-list2.html. Accessed: 2022-11-10.

[36] Eric Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, **2018**. doi:10.17487/RFC8446.

[37] T. Dierks and E. Rescorla. The transport layer security (tls) protocol version 1.2. RFC 5246, **2008**.

[38] John Kohl and Clifford Neuman. Ssl/tls protocols. *Internet Engineering Task Force (IETF)*, RFC 6101, **2011**.

[39] Eric Rescorla. *SSL and TLS: Designing and Building Secure Systems*. O'Reilly Media, **2008**.

[40] Alan O Freier, Phil Karlton, and Paul C Kocher. The secure sockets layer (ssl) protocol version 3.0. *Internet Engineering Task Force (IETF)*, RFC 6101, **1996**.

[41] RFC Editor. Certificate Transparency. Standard, RFC Editor, **2013**.

[42] Laurent Chuat, Pawel Szalachowski, Adrian Perrig, Ben Laurie, and Eran Messeri. Efficient gossip protocols for verifying the consistency of certificate logs. In *2015 IEEE Conference on Communications and Network Security (CNS)*, pages 415–423. **2015**. doi:10.1109/CNS.2015.7346853.

[43] Clemens Orthacker, Martin Centner, and Christian Kittl. Qualified mobile server signature. In Kai Rannenberg, Vijay Varadharajan, and Christian Weber, editors, *Security and Privacy – Silver Linings in the Cloud*, pages 103–111. Springer Berlin Heidelberg, Berlin, Heidelberg, **2010**.

[44] Heiko Rossnagel. Mobile qualified electronic signatures and certification on demand. In Sokratis K. Katsikas, Stefanos Gritzalis, and Javier López, editors, *Public Key Infrastructure*, pages 274–286. Springer Berlin Heidelberg, Berlin, Heidelberg, **2004**.

[45] Antonio Ruiz-Martínez, Daniel Sánchez-Martínez, María Martínez-Montesinos, and Antonio F. Gómez-Skarmeta. A survey of electronic signature solutions in mobile devices. *Journal of Theoretical and Applied Electronic Commerce Research*, 2(3):94–109, **2007**. ISSN 0718-1876. doi:10.3390/jtaer2030024.

[46] Information and Communication Technologies Authority (BTK). Btk 2023 3rd quarter report. `https://www.btk.gov.tr/uploads/pages/pazar-verileri/ceyrek-raporu-2023-3-c-eyrek-18-01-24-kurum-dis-i.pdf`, **2023**. Accessed on: June 2024.

[47] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Bitcoin.org*, **2008**.

[48] Vitalik Buterin. Ethereum: A next-generation smart contract and decentralized application platform. *Whitepaper*, **2013**.

[49] CAB Forum. CAB Forum Ballot 185. `https://cabforum.org/2017/02/24/ballot-185-limiting-lifetime-certificates`. Accessed: 2022-11-10.

[50] ESI. CAdES digital signatures Part 1: Building blocks and CAdES baseline signatures. Standard, Electronic Signatures and Infrastructures (ESI), Sophia Antipolis Cedex, FR, **2021**.

[51] Russell Housley, William Polk, William Ford, and David Solo. Cryptographic message syntax (cms) advanced electronic signatures (cades). RFC 5126, **2008**.

[52] ESI. XAdES Digital Signatures Part 1: Building blocks and XAdES baseline signatures. Standard, Electronic Signatures and Infrastructures (ESI), Sophia Antipolis Cedex, FR, **2022**.

[53] ESI. PAdES digital signatures Part 1: Building blocks and PAdES baseline signatures. Standard, Electronic Signatures and Infrastructures (ESI), Sophia Antipolis Cedex, FR, **2016**.

[54] ESI. JAdES digital signatures Part 1: Building blocks and JAdES baseline signatures. Standard, Electronic Signatures and Infrastructures (ESI), Sophia Antipolis Cedex, FR, **2021**.

[55] ESI. Procedures for Creation and Validation of AdES Digital Signatures; Part 1: Creation and Validation. Standard, Electronic Signatures and Infrastructures (ESI), Sophia Antipolis Cedex, FR, **2021**.

[56] ESI. Policy and security requirements for trust service providers providing long-term preservation of digital signatures or general data using digital signature techniques. Standard, Electronic Signatures and Infrastructures (ESI), Sophia Antipolis Cedex, FR, **2019**.

[57] ESI. Procedures for Creation and Validation of AdES Digital Signatures; Part 2: Signature Validation Report. Standard, Electronic Signatures and Infrastructures (ESI), Sophia Antipolis Cedex, FR, **2021**.

[58] ESI. Signature Validation Policy for European Qualified Electronic Signatures/Seals Using Trusted Lists. Standard, Electronic Signatures and Infrastructures (ESI), Sophia Antipolis Cedex, FR, **2021**.

[59] ESI. ETSI Plug Test Page, howpublished = https://signature-plugtests.etsi.org/pub/index.php. Accessed: 2022-11-10.

[60] A. De Luca, A. Hang, F. Brudy, and J. Lindqvist. Understanding shoulder surfing in the wild: Stories from users and observers. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. **2015**.

[61] V. Oleshchuk. Security of smart card-based signature protocols. *International Journal of Network Security*, **2004**.

[62] S. Picek et al. Security evaluation of digital signature algorithms. In *Proceedings of the 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. **2015**.

[63] Electronic signature fraud: A legal case. `https://digt.com/electronicsignature/tpost/z4zb23dcb1-electronic-signature-fraud-a-legal-case`. Accessed on: June 2024.

[64] Ali Evren GÖKSUNGUR. An electronic signature application for electronic document management systems. *International Journal of Engineering Science and Application*, 1(4):142–144, **2017**.

[65] Cláudio Pereira, Luís Barbosa, José Martins, and Jorge Borges. Digital signature solution for document management systems - the university of trás-os-montes and alto douro. In Álvaro Rocha, Hojjat Adeli, Luís Paulo Reis, and Sandra Costanzo, editors, *Trends and Advances in Information Systems and Technologies*, pages 16–25. Springer International Publishing, Cham, **2018**.

[66] Author(s) of the DocuSign Whitepaper. Docusign: A comprehensive overview. *DocuSign Documentation*, **Year**. `https://www.docusign.com/resources/whitepapers/overview`.

[67] P. Wuille. Key aggregation for schnorr signatures. https://blockstream.com/2018/01/23/musig-key-aggregation-schnorr-signatures/. Accessed: 2022-11-10.

[68] Ethereum Request for Comments. Ethereum Request for Comments (ERC-20) Homepage, howpublished = https://ethereum.org/en/developers/docs/standards/tokens/erc-20/. Accessed: 2022-11-10.

[69] ITU. The Directory: Public-key and attribute certificate frameworks. Standard, International Telecommunication Union (ITU), **2019**.

[70] RFC Editor. X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP. Standard, RFC Editor, **2013**.

[71] Marco Anisetti, Claudio Agostino Ardagna, and Nicola Bena. Multi-dimensional certification of modern distributed systems. *IEEE Transactions on Services Computing*, 16(3):1999–2012, **2023**. doi:10.1109/TSC.2022.3195071.

[72] Ricardo Neisse, José L. Hernández-Ramos, Sara N. Matheu, Gianmarco Baldini, and Antonio Skarmeta. Toward a blockchain-based platform to manage cybersecurity certification of iot devices. In *2019 IEEE Conference on Standards for Communications and Networking (CSCN)*, pages 1–6. **2019**. doi:10.1109/CSCN.2019.8931384.

[73] ISO. Information technology — Security techniques — Hash-functions — Part 1: General. Standard, ISO, **2016**.

[74] ISO. Information technology — Security techniques — Hash-functions — Part 2: Hash-functions using an n-bit block cipher. Standard, ISO, **2000**.

[75] ISO. IT Security techniques — Hash-functions — Part 3: Dedicated hash-functions. Standard, ISO, **2018**.

[76] ISO. Information technology — Security techniques — Hash-functions — Part 4: Hash-functions using modular arithmetic. Standard, ISO, **1998**.

[77] ISO. Information technology — Security techniques — Hash-functions — Part 4: Hash-functions using modular arithmetic. Standard, ISO, **1998**.

[78] RFC Editor. US Secure Hash Algorithms (SHA and HMAC-SHA). Standard, RFC Editor, **2006**.

[79] Madhura Patil and Pradeep Karule. Design and implementation of keccak hash function for cryptography, **2015**. doi:10.1109/ICCSP.2015.7322620.

[80] European Commission. Regulation (eu) no 910/2014 of the european parliament and of the council on electronic identification and trust services for electronic transactions in the internal market and repealing directive 1999/93/ec. *OJ*, L 257:1–42, **2014-08-28**.

[81] 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, **2008**.

[82] ITU. OSI networking and system aspects – Abstract Syntax Notation One (ASN.1). Standard, International Telecommunication Union (ITU), **2021**.

[83] RFC Editor. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. Standard, RFC Editor, **2008**.

[84] RFC Editor. Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP). Standard, RFC Editor, **2001**.

[85] RFC Editor. CMS Advanced Electronic Signatures (CAdES). Standard, RFC Editor, **2008**.

[86] W3C. Xml signature syntax and processing version 2.0. https://www.w3.org/TR/xmldsig-core2/. Accessed: 2022-11-10.

[87] Markus Jakobsson and Ari Juels. *Proofs of Work and Bread Pudding Protocols(Extended Abstract)*, pages 258–272. Springer US, Boston, MA, **1999**. ISBN 978-0-387-35568-9. doi:10.1007/978-0-387-35568-9_18.

[88] Cong T. Nguyen, Dinh Thai Hoang, Diep N. Nguyen, Dusit Niyato, Huynh Tuong Nguyen, and Eryk Dutkiewicz. Proof-of-stake consensus mechanisms for future blockchain networks: Fundamentals, applications and opportunities. *IEEE Access*, 7:85727–85745, **2019**. doi:10.1109/ACCESS.2019.2925010.

[89] Ethereum.org. Ethereum website: Nodes, howpublished = `https://ethereum.org/en/developers/docs/nodes-and-clients/`. Accessed: 2022-11-10.

[90] 101blockchains.com Website. 101Blockchains Homepage, howpublished = https://101blockchains.com/blockchain-vs-database-the-difference/. Accessed: 2022-11-10.

[91] Konstantinos Christidis and Michael Devetsikiotis. Blockchains and smart contracts for the internet of things. *IEEE Access*, 4:2292–2303, **2016**. doi:10.1109/ACCESS.2016.2566339.

[92] Jesse Yli-Huumo, Deokyoon Ko, Sujin Choi, Sooyong Park, and Kari Smolander. Where is current research on blockchain technology?—a systematic review. *PLOS ONE*, 11, **2016**. doi:10.1371/journal.pone.0163477.

[93] Zibin Zheng, Shaoan Xie, Hongning Dai, Xiangping Chen, and Huaimin Wang. An overview of blockchain technology: Architecture, consensus, and future trends. In *2017 IEEE International Congress on Big Data (BigData Congress)*, pages 557–564. **2017**. doi:10.1109/BigDataCongress.2017.85.

[94] Ahmed Afif Monrat, Olov Schelén, and Karl Andersson. A survey of blockchain from the perspectives of applications, challenges, and opportunities. *IEEE Access*, 7:117134–117151, **2019**. doi:10.1109/ACCESS.2019.2936094.

[95] Salabat Khan, Zijian Zhang, Liehuang Zhu, Mussadiq Abdul Rahim, Sadique Ahmad, and Ruoyu Chen. Scm: Secure and accountable tls certificate

management. *International Journal of Communication Systems*, 33(15):e4503, **2020**. doi:https://doi.org/10.1002/dac.4503. E4503 dac.4503.

[96] Salabat Khan, Liehuang Zhu, Zijian Zhang, Mussadiq Abdul Rahim, Khalid Khan, and Meng Li. Attack-resilient tls certificate transparency. *IEEE Access*, 8:98958–98973, **2020**. doi:10.1109/ACCESS.2020.2996997.

[97] Salabat Khan, Fei Luo, Zijian Zhang, Farhan Ullah, Farhan Amin, Syed Furqan Qadri, Md Belal Bin Heyat, Rukhsana Ruby, lu Wang, Shamsher Ullah, Meng Li, Victor Leung, and Kaishun Wu. A survey on x.509 public-key infrastructure, certificate revocation, and their modern implementation on blockchain and ledger technologies. *IEEE Communications Surveys & Tutorials*, PP:1–1, **2023**. doi:10.1109/COMST.2023.3323640.

[98] Comodo. Comodo incident report. https://www.comodo.com/Comodo-Fraud-Incident-2011-03 Accessed: 2022-11-10.

[99] Henry Birge-Lee, Yixin Sun, Anne Edmundson, Jennifer Rexford, and Prateek Mittal. Bamboozling certificate authorities with BGP. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 833–849. USENIX Association, Baltimore, MD, **2018**. ISBN 978-1-939133-04-5.

[100] Salabat Khan, Fei Luo, Zijian Zhang, Mussadiq Abdul Rahim, Mubashir Ahmad, and Kaishun Wu. Survey on issues and recent advances in vehicular public-key infrastructure (vpki). *Commun. Surveys Tuts.*, 24(3):1574–1601, **2022**. ISSN 1553-877X. doi:10.1109/COMST.2022.3178081.

[101] Turkish Ministry of Transport and infrastructure. News. https://www.uab.gov.tr/haberler/2023-yili-2-nci-ceyregine-iliskin-turkiye-elektronik-haberlesme-sektoru -3-aylik-pazar-verileri-raporu-aciklandi. Accessed: 2022-11-10.

[102] Turkish Information Technologies Institution. Market data in turkey. https://www.btk.gov.tr/pazar-verileri. Accessed: 2022-11-10.

[103] CAB Forum. Cab forum ballot 185. https://cabforum.org/wp-content/uploads/CA-Browser-Forum-BR-v2.0.1.pdf. Accessed: 2022-11-10.

[104] ENISA. The EU Cyber Security Agency: Guidelines on Supervision of Qualified Trust Services - Technical guidelines on trust services. Guideline, The European Union Agency for Cybersecurity (ENISA, **2017**.

[105] Commission Implementing Desicion, howpublished = `https://ec.europa.eu/futurium/en/system/\files/ged/celex_32015d1505_en_txt.pdf`. Accessed: 2022-11-10.

[106] Ca/browser forum. https://cabforum.org/. Accessed: 2023-07-21.

[107] Investopedia. Diginotar 531 fraudulent certificates, howpublished = `https://www.investopedia.com/terms/1/51-attack.asp`. Accessed: 2022-11-10.

[108] Jonah Groendal. Asn1 decode project. https://www.btk.gov.tr/pazar-verileri. Accessed: 2022-11-10.

[109] Code4Rena. Bytes transform codes for asn.1 decoding. https://github.com/code-423n4/2023-04-ens/tree/main/contracts/dnssec-oracle. Accessed: 2022-11-10.

[110] Serpwatch. Sepwatch SSL Statistics, howpublished = `https://serpwatch.io/blog/ssl-stats`. Accessed: 2022-11-10.

[111] Wired.com. Wired Homepage, howpublished = `https://www.wired.com/2017/01/half-web-now-encrypted-makes-everyone-safer/`. Accessed: 2022-11-10.

[112] Daniel Davis Wood. Ethereum: A secure decentralised generalised transaction ledger. In *Yellow Paper*, pages 1–34. **2014**.

[113] Google Cloud Website. HDD Prices, howpublished = https://cloud.google.com/storage/pricing. Accessed: 2022-11-10.

[114] Certificate Transparency Dev Website. How CT works, howpublished = https://certificate.transparency.dev/howctworks/. Accessed: 2022-11-10.

[115] Binance. Binance smart chain. https://www.bnbchain.org/en/bnb-smart-chain. Accessed: 2022-11-10.

[116] Polygon. Poltgon. https://polygon.technology/. Accessed: 2022-11-10.

[117] Avalanche. Avalanche chain. https://www.avax.network/. Accessed: 2022-11-10.