ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

A NOVEL KEY MANAGEMENT FRAMEWORK FOR SECURE AND SCALABLE DECENTRALIZED IDENTITY SYSTEMS

M.Sc. THESIS

Mert YILDIZ

Department of Computer Engineering

Computer Engineering Programme

MAY 2025

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

A NOVEL KEY MANAGEMENT FRAMEWORK FOR SECURE AND SCALABLE DECENTRALIZED IDENTITY SYSTEMS

M.Sc. THESIS

Mert YILDIZ
(504171518)

Department of Computer Engineering

Computer Engineering Programme

Thesis Advisor: Prof. Dr. Şerif BAHTİYAR

MAY 2025

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

GÜVENLİ VE ÖLÇEKLENEBİLİR DAĞITIK KİMLİK SİSTEMLERİ İÇİN YENİ BİR ANAHTAR YÖNETİM MİMARİSİ

YÜKSEK LİSANS TEZİ

Mert YILDIZ
(504171518)

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

Tez Danışmanı: Prof. Dr. Şerif BAHTİYAR

MAYIS 2025

Mert YILDIZ, a M.Sc. student of ITU Graduate School student ID 504171518 successfully defended the thesis entitled "A NOVEL KEY MANAGEMENT FRAMEWORK FOR SECURE AND SCALABLE DECENTRALIZED IDENTITY SYSTEMS", which he/she prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Prof. Dr. Şerif BAHTİYAR** Istanbul Technical University	.....
Jury Members :	**Asst. Prof. Dr. Ayşe YILMAZER METİN** Istanbul Technical University	.....
	Prof. Dr. Emin ANARIM Boğaziçi University	.....
		.....

Date of Submission : **25 April 2025**
Date of Defense : **15 May 2025**

To my spouse and family,

FOREWORD

First and foremost, I would like to express my deepest gratitude to my advisor, Prof. Dr. Şerif BAHTİYAR, for their invaluable guidance, support, and encouragement throughout this journey. Your insights and expertise have been instrumental in shaping this research, and I am truly grateful for your patience and mentorship.

I am also profoundly thankful to my beloved spouse, Büşra, whose unwavering support, understanding, and love have been my greatest source of strength. Your belief in me, even during the most challenging moments, has been my motivation to keep moving forward. To my family, I extend my heartfelt appreciation for their endless encouragement, sacrifices, and support. Your unwavering faith in me has been a constant source of inspiration, and I would not have reached this milestone without you. This work is dedicated to all of you with great gratitude.

May 2025

Mert YILDIZ
(Computer Engineer)

TABLE OF CONTENTS

ABBREVIATIONS

AES	**:** Advanced Encryption Standard
App	**:** Appendix
BIP32	**:** Bitcoin Improvement Proposal 32
DIDs	**:** Decentralized Identifiers
DoS	**:** Denial of Service
FIPS	**:** Federal Information Processing Standards
GCM	**:** Galois/Counter Mode
HD	**:** Hierarchical Deterministic
HMAC	**:** Hash-based Message Authentication Code
HSM	**:** Hardware Security Module
ID	**:** Identity
ISO	**:** International Organization for Standardization
KEK	**:** Key Encryption Key
NIST	**:** National Institute of Standards and Technology
RNG	**:** Random Number Generator
SHA	**:** Secure Hash Algorithms
SLIP10	**:** SatoshiLabs Improvement Proposal 10
TEE	**:** Trusted Execution Environment
TLS	**:** Transport Layer Security

SYMBOLS

$\mathbf{m_i}$	**:** Child Index
t	**:** Time

LIST OF TABLES

Page

LIST OF FIGURES

A NOVEL KEY MANAGEMENT FRAMEWORK FOR SECURE AND SCALABLE DECENTRALIZED IDENTITY SYSTEMS

SUMMARY

In today's digital landscape, managing identities securely and privately is a significant challenge. Users often maintain multiple identities across various platforms, leading to privacy concerns, data breaches, and identity theft. A centralized approach to identity management can expose sensitive information and weaken user control over personal data. This situation has driven the need for decentralized identity (DID) systems.

The rapid adoption of decentralized identity systems has introduced significant challenges in managing cryptographic keys securely and efficiently, particularly in large-scale implementations such as national identity programs. Traditional key management approaches often struggle with scalability, security isolation, recovery, and delegation, posing risks to the integrity and usability of decentralized identity solutions. This study proposes a novel key management framework that integrates the hierarchical deterministic (HD) key management, Hardware Security Modules (HSMs), and key wrapping mechanisms to provide a scalable and secure solution for decentralized identity systems.

The proposed framework leverages BIP32 hierarchical deterministic key derivation, enabling efficient management of cryptographic keys by deriving multiple keys from a single master seed. This method significantly reduces the complexity of managing individual keys and enhances security by maintaining strict key isolation. The Hardware Security Module (HSM) serves as a root of trust, securely storing the Key Encryption Key (KEK) and performing cryptographic operations, including key wrapping and unwrapping. Key wrapping ensures that master seeds can be stored externally without compromising security, facilitating scalability without burdening the HSM with excessive storage requirements.

The framework addresses five key challenges in decentralized identity management: Scalability by supporting large-scale key storage with external key-wrapped master seeds, Security isolation by implementing hierarchical key derivation structures, Recovery by securely restoring keys through deterministic derivation, Secure delegation by enabling hierarchical access control, and Complexity reduction by centralizing key management under a single master seed.

A comprehensive security analysis demonstrates the framework's resilience against various attack vectors, including side-channel attacks, cryptanalytic attacks, insider threats, and quantum computing threats. By leveraging HSMs with FIPS 140-2 Level 4 certification, secure algorithm selection for key wrapping, and multi-factor authentication mechanisms, the proposed approach significantly enhances security compared to existing decentralized identity key management solutions.

We analyze the security and performance of BIP32 vs. SLIP10 in an HSM environment, measuring key derivation speed, transaction signing efficiency, and resistance to attacks. Our results show that SLIP10 offers stronger security guarantees, while BIP32 remains dominant due to blockchain compatibility. Also, for the first time in the literature, performance comparison of BIP32 and SLIP10 algorithms on HSM is performed.

The primary contributions of this study to the literature are as follows: It presents a framework design that integrates BIP32, HSMs, and key wrapping techniques for secure and scalable key management in decentralized identity systems. It demonstrates how the proposed framework addresses the challenges of scalability, recovery, security isolation, secure delegation, and complexity in decentralized identity management. It provides a detailed security analysis of the framework, including a threat model and discussion of security properties. It delivers a comparative analysis of BIP32 (ECDSA) and SLIP10 (EdDSA) in HSM environments. It presents empirical evaluations of key derivation speed, signing latency, and computational efficiency.

This research presents the first known integration of BIP32-based key derivation with HSMs and key wrapping for decentralized identity systems, providing a robust and scalable key management framework. The findings of this study have significant implications for national identity infrastructures, digital authentication mechanisms, and blockchain-based identity management systems, offering a secure, efficient, and scalable approach to decentralized identity key management. Future research will focus on prototype implementation, performance testing, and integration of post-quantum cryptographic algorithms to further enhance the framework's resilience against emerging threats.

GÜVENLİ VE ÖLÇEKLENEBİLİR DAĞITIK KİMLİK SİSTEMLERİ İÇİN YENİ BİR ANAHTAR YÖNETİM MİMARİSİ

ÖZET

Günümüz dijital ortamlarında, kimlikleri güvenli ve özel bir şekilde yönetmek önemli bir zorluktur. Kullanıcılar genellikle çeşitli platformlarda birden fazla kimliğe sahiptir ve bu da gizlilik endişelerine, veri ihlallerine ve kimlik hırsızlığına yol açar. Kimlik yönetimine yönelik merkezi bir yaklaşım hassas bilgileri açığa çıkarabilir ve kullanıcıların kişisel verileri üzerindeki kontrolünü zayıflatabilir. Bu durum merkezi olmayan kimlik sistemlerine yönelim ihtiyacını doğurmuştur.

Merkezi olmayan kimlik sistemlerinin hızla benimsenmesi, özellikle ulusal kimlik programları gibi büyük ölçekli uygulamalarda kriptografik anahtarların güvenli ve verimli bir şekilde yönetilmesi konusunda önemli zorlukları gün yüzüne çıkarmıştır. Geleneksel anahtar yönetim yaklaşımları, ölçeklenebilirlik, güvenlik izolasyonu, anahtar kurtarma ve güvenli yetkilendirme gibi alanlarda yetersiz kalmakta, merkezi olmayan kimlik çözümlerinin bütünlüğü ve kullanılabilirliği açısından riskler oluşturmakta ve yaygınlaşmasını engellemektedir. Bu çalışma, Bitcoin Improvement Proposal 32 (BIP32) hiyerarşik deterministik anahtar yönetimi, Donanım Güvenlik Modülleri (HSM) ve anahtar sarma (key wrapping) mekanizmalarını entegre eden yeni bir anahtar yönetim çerçevesi önermektedir.

Bu çerçevede kullanılan algoritmalar, teknolojiler ve yöntemlerden bahsetmek gerekirse, Hiyerarşik Deterministik Anahtarlar olarak da bilinen BIP32 (Bitcoin Improvement Proposal 32), tek bir ana "tohum"dan çok sayıda kriptografik anahtarın türetilmesini sağlayan bir standarttır. Bu sayede, kullanıcılar tek bir ana anahtarı güvende tutarak, farklı amaçlar için (örneğin, imzalama, şifreleme) farklı anahtarlar oluşturabilirler. Bu, güvenli saklanması gereken anahtar sayısını azaltarak anahtar yönetimini kolaylaştırır ve güvenliği artırır. Yine benzer amaçla hiyerarşik deterministik anahtar yönetimi için SLIP10 algoritması da kullanılabilir. BIP32 algoritması ECDSA tabanlı olup secp256k1 eğrisini kullanırken, SLIP10 algoritması EdDSA tabanlı olup Ed25519 eğrisini kullanmaktadır.

Donanım Güvenlik Modülü (HSM), kriptografik işlemleri güvenli bir şekilde gerçekleştirmek ve hassas verileri (örneğin, özel anahtarlar) korumak için tasarlanmış özel donanım cihazıdır. HSM'ler, kurcalamaya karşı dayanıklıdır ve içerdikleri verilerin yetkisiz erişime karşı korunmasını sağlar.

Anahtar Sarmalama (Key Wrapping), bir anahtarı başka bir anahtarla şifreleyerek koruma işlemidir. Bu sayede, anahtarların güvenli bir şekilde saklanması ve taşınması sağlanır. Anahtar sarmalama, özellikle büyük miktarda anahtarın yönetildiği sistemlerde önemlidir.

Önerilen çerçeve, BIP32 hiyerarşik deterministik anahtar türetme algoritmasını kullanarak, tek bir ana tohumdan (master seed) türetilen çok sayıda anahtarın verimli bir şekilde yönetilmesini sağlar. Bu yöntem, bireysel anahtarları yönetme karmaşıklığını azaltırken, her anahtarın güvenli bir şekilde izole edilmesini sağlayarak sistemin genel güvenliğini artırır. Donanım Güvenlik Modülü (HSM), sistemin kök güveni olarak hareket eder, Anahtar Şifreleme Anahtarını (KEK) güvenli bir şekilde saklar ve anahtar sarma gibi kriptografik işlemleri gerçekleştirir. Anahtar sarma mekanizması, ana tohumların dış ortamda güvenli bir şekilde saklanmasına olanak tanır ve HSM üzerindeki depolama yükünü azaltarak sistemin ölçeklenebilirliğini artırır.

Bu çerçeve, merkezi olmayan kimlik yönetimiyle ilgili beş temel sorunu ele almaktadır. Bu sorunlardan ilki ölçeklenebilirliktir. Bu çerçeve, büyük ölçekli sistemlerde, HSM'nin yükünü artırmadan milyonlarca anahtarı güvenli bir şekilde yönetmeyi sağlar. İkinci temel sorun güvenlik izolasyonudur. Bu çerçevede hiyerarşik anahtar türetme yapıları sayesinde, farklı işlevler için kullanılan anahtarlar birbirinden izole edilir. Üçüncü temel sorun kurtarmadır. Bu çerçevede deterministik anahtar türetme yöntemiyle kaybolan anahtarlar güvenli bir şekilde geri yüklenebilir. Dördüncü temel sorun güvenli yetkilendirmedir. Bu çerçevede hiyerarşik erişim kontrol mekanizması sayesinde, farklı yetki seviyelerinin güvenli bir şekilde yönetilmesi sağlanır. Bir diğer temel sorun ise karmaşıklık azaltmadır. Bu çerçevede kullanıcıların sadece tek bir ana tohum yönetmesi gerekmekte olduğundan, bu durum genel sistem karmaşıklığını azaltmaktadır.

Önerilen çerçevenin güvenliği, yan kanal saldırıları, kriptoanalitik saldırılar, içeriden gelen tehditler ve kuantum bilişim tehditleri gibi çeşitli saldırı vektörlerine karşı kapsamlı bir şekilde analiz edilmiştir. Sistem, FIPS 140-2 Seviye 4 sertifikalı HSM'ler, güvenliği kanıtlanmış algoritmalar ile şifreleme ve çok faktörlü kimlik doğrulama mekanizmaları kullanarak mevcut merkezi olmayan kimlik anahtar yönetimi çözümlerine kıyasla önemli ölçüde daha yüksek güvenlik sağlamaktadır. Önerilen çerçeve, FIPS 140-2 Seviye 4+ sertifikalı HSM'ler kullanarak fiziksel ve yan kanal saldırılarına karşı koruma sağlar. AES-256 şifreleme ve KEK güncelleme ile kriptanalitik saldırılar engellenir. X.509 sertifikaları ve TLS 1.3 ile ağ tabanlı saldırılara karşı güvenli iletişim sağlanır. En az ayrıcalık ilkesi, görev ayrımı ve çok faktörlü kimlik doğrulama ile içeriden gelen tehditler azaltılır. Hız sınırlama ve yük dengeleme ile hizmet reddi saldırıları önlenir. Sosyal mühendislik saldırılarına karşı güvenlik eğitimi verilir. Veritabanı şifreleme ve erişim kontrolleri ile harici depolamaya yönelik saldırılar engellenir. Kripto çevikliği ile kuantum bilgisayar tehditlerine karşı hazırlık yapılır. Bu çok katmanlı yaklaşım, çeşitli saldırı vektörlerine karşı kapsamlı bir koruma sağlayarak merkeziyetsiz kimlik sistemlerinin güvenliğini önemli ölçüde artırır.

Güvenlik analizine ek olarak, önerilen çerçevede anahtar türetme hızı, işlem imzalama verimliliği ve saldırılara karşı direnç bir HSM ortamında ölçülerek BIP32 ile SLIP10'un güvenlik ve performansı analiz edilmiştir. Elde edilen sonuçlara göre, SLIP10'un daha güçlü güvenlik garantileri sunduğunu, BIP32'nin ise blokzincir uyumluluğu nedeniyle baskın olmaya devam ettiğini göstermektedir. Ayrıca,

literatürde ilk kez BIP32 ve SLIP10 algoritmalarının HSM üzerinde performans karşılaştırması yapılmıştır.

Bu araştırma, BIP32 tabanlı anahtar türetme ile HSM ve anahtar sarma mekanizmalarını merkezi olmayan kimlik sistemleri için entegre eden ilk çalışmalardan biridir ve ulusal kimlik altyapıları, dijital kimlik doğrulama sistemleri ve blokzincir tabanlı kimlik yönetim sistemleri için güvenli, verimli ve ölçeklenebilir bir çözüm sunmaktadır.

Bu çalışmanın literatüre temel katkıları şunlardır: Merkeziyetsiz kimlik sistemlerinde güvenli ve ölçeklenebilir anahtar yönetimi için BIP32, HSM'ler ve anahtar sarmalama tekniklerini entegre eden bir çerçeve tasarımı sunulmuştur. Önerilen çerçevenin, merkeziyetsiz kimlik yönetiminde ölçeklenebilirlik, kurtarma, güvenlik yalıtımı, güvenli devretme ve karmaşıklık zorluklarını nasıl çözdüğü gösterilmiştir. Bir tehdit modeli ve güvenlik özelliklerinin tartışılması dahil olmak üzere, önerilen çerçevenin ayrıntılı bir güvenlik analizi yapılmıştır. BIP32 (ECDSA) ve SLIP10 (EdDSA) algoritmalarının HSM üzerinde kullanımının karşılaştırmalı analizi yapılmıştır. Ayrıca, deneysel olarak anahtar türetme hızı, imza işlemindeki gecikme ve hesaplama işlemindeki efektiflik bakımlarından ölçümleme yapılmıştır.

Gelecekteki çalışmalar, prototip uygulaması, performans değerlendirmesi ve kuantum sonrası kriptografik algoritmaların entegrasyonu üzerine yoğunlaşacaktır. Böylece önerilen çerçevenin kuantum bilgisayarlar gibi yeni ortaya çıkan tehditlere karşı dayanıklılığı daha da artırılacaktır.

1. INTRODUCTION

Decentralized identity management systems have received considerable attention in recent years due to their potential to increase privacy, security and user control over personal data. While traditional centralized identity management systems often struggle with issues such as data breaches, lack of user control and single points of failure, decentralized systems aim to address these concerns by decentralizing control and increasing user autonomy [1]. However, as these systems scale to millions of users, as in a national ID system, they face new challenges in managing the large number of cryptographic keys required. In particular, there are critical key management challenges in ensuring the security and scalability of cryptographic key generation, storage and recovery [2]. Decentralized identity systems face several critical challenges in key management:

- **Scalability:** Traditional key generation and management methods struggle to efficiently handle the large number of keys required in systems with millions of users [3].
- **Recovery and Backup:** Traditional key recovery methods are often cumbersome and insecure, risking permanent loss of digital identity in case of key loss [4].
- **Security Isolation:** Inadequate separation between keys used for different purposes increases the risk of systemic compromise if a single key is exposed.
- **Secure Delegation:** Implementing hierarchical access control and secure delegation of authority without exposing sensitive credentials is challenging in decentralized systems [5].
- **Complexity:** Managing multiple individual cryptographic keys for various operations leads to increased complexity and potential for key loss.

Key management challenges collectively pose significant obstacles to the widespread adoption and efficient operation of decentralized identity systems and require a more robust and scalable key management solution.

In this thesis, we address the aforementioned key management challenges of scalability and security. A novel framework is proposed that addresses these key management challenges by leveraging the Bitcoin Improvement Proposal 32 (BIP32) algorithm [6], Hardware Security Modules (HSMs) and key wrapping techniques. The proposed framework effectively addresses the aforementioned challenges by providing for the first time in the literature a secure and scalable method for managing cryptographic keys in large-scale decentralized identity systems.

Traditional key management systems require store separate private keys for each identity, leading to complex key management and increased exposure to security threats. To address this challenge, Hierarchical deterministic (HD) BIP32 algorithm is introduced that allow all cryptographic keys to be derived from a single master seed. While HD algorithms offer scalability and recovery, their software-based implementations introduce new security risks, which are shown as.

- Master Seed Exposure: If the master seed is compromised, all derived keys can be reconstructed [7].
- Memory and Malware Attacks: Software based solutions store keys in volatile memory, exposing them to RAM scraping and keyloggers [8].
- Non-Hardened Key Derivation Risks: In HD algorithms, non-hardened child keys can leak parent keys, compromising the hierarchy [9].
- Insider Threats: Decentralized identity systems without hardware security are vulnerable to internal fraud [10].

In this research, we propose integrating HD algorithms with *HSMs* to provide tamper-resistant key storage and secure cryptographic operations. An HSM is a tamper-resistant cryptographic device that provides isolated key storage and secure transaction signing. By integrating HSMs with HD algorithms, institutions may have the following benefits.

- Protect the master seed from extraction attacks.
- Ensure hardware-based key derivation without software exposure.
- Enforce multi-party authentication (MPA) to prevent unauthorized transactions.
- Comply with security standards.

The key contributions of this thesis are:

- A framework design that integrates BIP32, HSMs, and key wrapping techniques for secure and scalable key management in decentralized identity systems.
- Demonstrate how the proposed framework solves the challenges of scalability, recovery, security isolation, secure delegation and complexity in decentralized identity management.
- A detailed security analysis of the proposed framework, including a threat model and discussion of security properties.
- Evaluates the performance and security trade-offs of using BIP32 (ECDSA) and SLIP10 (EdDSA) in an HSM environment.

The remainder of this thesis is organized as follows: Section II provides background information and discusses related work. Section III details the proposed framework. Section IV presents a comprehensive security analysis of the framework and discusses how the proposed approach addresses key management challenges, its advantages, limitations and results. Finally, Section VI concludes the thesis and suggests directions for future work.

1.1 Purpose of Thesis

The purpose of this work is to address the critical challenges of security and scalability in decentralized identity systems by proposing a novel key management framework. The framework integrates the BIP32 hierarchical deterministic key derivation algorithm, hardware security modules (HSMs), and key wrapping techniques to ensure efficient and secure cryptographic key management in large-scale identity infrastructures, such as national digital identity programs. By leveraging HSMs as the root of trust and implementing key wrapping for scalable external storage, the framework enhances security isolation, simplifies key recovery, and mitigates key management complexities. This study aims to facilitate the widespread adoption of decentralized identity systems by overcoming key management issues that hinder their scalability, usability, and security. The proposed framework ensures secure delegation, seamless key recovery, and efficient management of large-scale cryptographic keys, thereby making decentralized identity systems more practical for real-world applications. A comprehensive security analysis demonstrates the framework's resilience against various threat vectors, reinforcing its potential as a foundational approach for the secure and scalable deployment of decentralized identity systems at a national and global level.

1.2 Contribution

The proposed framework makes significant contributions by solving critical key management challenges that have hindered the adoption of decentralized identity systems. These contributions can be categorized into five major areas:

- Scalability Through Hierarchical Key Derivation (BIP32),
- Security Enhancement with Hardware Security Modules (HSMs),
- Solving the Problem of Secure Key Recovery, Secure Delegation and Access Control in Decentralized Identity Systems,
- Comprehensive Security Analysis and Threat Mitigation,

- A comparative analysis of BIP32 (ECDSA) vs. SLIP10 (EdDSA) in HSM environments.
- Experimental benchmarking of key derivation speed, signing latency, and computational efficiency.

One of the main contributions of this work is the application of BIP32 hierarchical deterministic key derivation to decentralized identity management. While BIP32 has been widely used in cryptocurrency wallets [11], [12], its application in national-scale identity infrastructures has not been extensively explored.

The study extends the application of BIP32 beyond blockchain wallets to identity systems, where hierarchical key structures significantly reduce storage and management overhead. The framework enables users to derive multiple cryptographic keys from a single master seed, solving the challenge of handling millions of keys in large-scale deployments. The approach provides a deterministic recovery mechanism, ensuring that lost keys can be regenerated without requiring centralized backups. By integrating BIP32-based key derivation with decentralized identity frameworks, this study advances the state-of-the-art in scalable key management.

The study enhances existing key management models by incorporating HSM-based key protection, addressing critical security vulnerabilities associated with software-based key storage.

Unlike conventional software wallets, where keys are often stored on disk and vulnerable to malware or unauthorized access [13], the proposed framework stores the master seed inside an HSM, ensuring that sensitive cryptographic material never leaves a tamper-resistant environment. HSM-based key wrapping provides scalable and secure storage by encrypting the master seed and enabling its storage in external databases without risk of exposure. The framework isolates cryptographic operations, meaning that all key generation, signing, and encryption operations occur within the secure boundaries of an HSM, protecting against software-based attacks. This contribution is particularly significant for national identity systems, where the protection of cryptographic keys is paramount for ensuring privacy and preventing identity fraud.

A major drawback of traditional cryptographic key management approaches is the lack of efficient recovery mechanisms. The loss of a private key often leads to the permanent loss of digital identity [14]. The proposed framework introduces a key recovery mechanism based on securely stored wrapped master seeds, ensuring that users can regain access to their identities without compromising security.

Unlike previous approaches, which rely on centralized key escrow or user-maintained backups, this study proposes a secure recovery mechanism where wrapped master seeds are stored externally and can only be decrypted by the HSM. The framework allows for hierarchical recovery, meaning that if a derived key is lost, a new key can be generated from the master seed without requiring full identity reissuance. This method significantly improves usability and resilience in decentralized identity systems, ensuring that users do not lose access to their identities due to lost or compromised keys. By introducing an HSM-based key recovery mechanism, the study provides a robust alternative to traditional key backup and restoration approaches, reducing reliance on centralized key escrow services that could introduce security risks.

Decentralized identity systems require a way to delegate access to specific services without compromising security. Traditional models often rely on direct key sharing, which can lead to unintended exposure of cryptographic material [5].

This study introduces a hierarchical key delegation model, where specific keys can be derived from the master seed for delegation purposes, ensuring that access rights can be granted without exposing the master key. By leveraging BIP32's hierarchical structure, the framework allows users to generate role-based cryptographic keys, enabling secure delegation of identity verification operations. The model supports revocable delegation, allowing users to invalidate compromised keys and generate new ones dynamically. This contribution significantly enhances privacy and security in decentralized identity frameworks, making them more suitable for enterprise and government applications.

A critical contribution of this study is its detailed security analysis, which evaluates the proposed framework against various threat models. Previous research in key

management [15] has focused primarily on theoretical implementations without thoroughly assessing security vulnerabilities.

The study presents a multi-layered security approach, addressing threats such as side-channel attacks, insider threats, network-based attacks, and quantum computing risks. It evaluates physical security measures, demonstrating how HSM-based key storage mitigates the risk of physical tampering and cryptographic key extraction. The research introduces a novel risk assessment model, providing a quantitative evaluation of potential attack vectors and their impact on decentralized identity systems. The security analysis validates the resilience of the proposed framework against a wide range of attack scenarios, bridging the gap between theoretical key management models and real-world security implementations.

This paper evaluates the performance and security trade-offs of using BIP32 (ECDSA) and SLIP10 (EdDSA) in an HSM environment. This study provides a comparative analysis of BIP32 (ECDSA) and SLIP10 (EdDSA) in HSM environments and an experimental comparison of key derivation speed, signing latency and computational efficiency.

This study makes several novel contributions to the field of decentralized identity management by introducing an integrated key management framework that combines scalability (BIP32), security (HSMs), efficient recovery mechanisms, and secure delegation models. The proposed approach solves long-standing challenges in cryptographic key management, enabling the widespread adoption of decentralized identity systems in national and enterprise settings.

By addressing both scalability and security challenges, this study lays the foundation for the next generation of secure and scalable identity management systems, paving the way for real-world adoption of decentralized digital identities.

2. DECENTRALIZED IDENTITY AND KEY MANAGEMENT

This chapter reviews existing studies in the field, identifies their limitations, and highlights how the proposed framework addresses these gaps. One of the foundational components of decentralized identity systems is the management of cryptographic keys. The issue of key management complexity is central to decentralized identity systems due to the vast number of keys required for different purposes such as authentication, signing, and encryption. Traditional approaches often involve managing individual keys, leading to a significant overhead in key storage and handling. Nakamoto's initial work on Bitcoin [16] and subsequent extensions introduced hierarchical deterministic (HD) key management, allowing multiple keys to be derived from a single master seed. The BIP32 algorithm, introduced by Wuille in 2012 [6], revolutionized cryptocurrency wallet management through hierarchical deterministic wallets. Studies have introduced HD key management techniques like BIP32 to efficiently derive multiple keys from a single master seed [12]. These techniques have been pivotal in improving scalability and reducing key management complexity. However, these studies often assume secure storage of the master seed and key encryption keys (KEKs), without providing a comprehensive solution for securely storing these sensitive values outside of software-based wallets. The BIP32 algorithm, as implemented in various studies [17], [18], has shown promise in hierarchical key management but has primarily been limited to cryptocurrency applications. Although it has been proposed to be used for IoT devices other than these scenarios [19], this is the first time in the literature that it has been proposed to solve key management problems in decentralized identity systems. Das et al. [11] provided formal security analysis of BIP32 wallets but did not explore its application in national identity systems. Moreover, while BIP32 addresses scalability issues, it does not offer a robust mechanism to protect against physical tampering or side-channel attacks on HSMs. Existing studies tend to overlook the security challenges associated with key management in large-scale deployments, particularly

when HSMs are integrated as the root of trust in decentralized systems. Previous research in decentralized identity systems and key management has made significant contributions but also revealed several limitations. Ahmed et al. [13] proposed a blockchain-based identity management system but did not address the scalability challenges of key management in large-scale deployments. Similarly, Lim et al. [14] focused on blockchain-based solutions but failed to provide comprehensive security measures for key storage and protection. In the realm of key management, Farhad et al. [20] introduced Solid Pod technology for key storage, but their solution lacked integration with hardware security modules, making it vulnerable to physical attacks. Barker et al. [15] discussed best practices for key management but did not consider the specific requirements of national-scale identity systems. Although Sivanantham et al. [21] studied a secure key management method for blockchain use case, their work did not provide a complete framework for scalable key management. Various studies have been conducted to analyze HSM against various attacks. While Genkin et al. [22] demonstrate vulnerabilities to side-channel attacks. Dib and Pierre [23] analyze insider attack vulnerabilities, emphasizing the need for comprehensive security measures addressing both external and internal risks. From these analyses, the security status of HSM against various attacks has been examined. Additionally, existing HSM implementations [24], [25] have focused on general cryptographic operations without specifically addressing the needs of decentralized identity systems. There have also been proposals for HSM-based key management systems [26]. However, since the keys are stored in the HSM, the scalability problem has not been solved in these proposals. In addition, these proposals have been implemented for specific blockchain networks and are not recommended for use in distributed identity systems. Also, these studies lack an efficient mechanism for key wrapping and secure recovery, particularly when managing millions of user keys in large-scale identity systems. In summary, while existing research has laid a solid foundation in key management for decentralized identity systems, the proposed framework extends these works by incorporating HSM-based secure key storage and key wrapping mechanisms, enhancing scalability through BIP32 HD key derivation, strengthening the framework

against various attacks, and providing secure and scalable key management solutions for national level decentralized identity systems.

2.1 Decentralized Identity Systems

Traditional identity management systems rely on centralized authorities such as governments, corporations, and financial institutions to issue and verify identities. These centralized systems introduce several challenges, including the risk of single points of failure, identity theft, privacy concerns, and lack of user control over personal data [27]. As a response to these issues, **Decentralized Identity Systems (DIDs)** have emerged as a new paradigm, enabling individuals to have self-sovereign control over their digital identities. The Table 2.1 summarizes the comparison of different aspects of decentralized and centralized identity approaches.

Table 2.1 : Comparison of Centralized and Decentralized Identity Systems.

Aspect	**Centralized Identity Systems**	**Decentralized Identity Systems**
Data Control	Central authority controls and stores user data	Users have complete control over their data
Privacy	User data is vulnerable to data breaches and misuse	Enhanced privacy as users decide what data to share
Security	Centralized systems are susceptible to single points of failure	Blockchain-based security and resilience
Identity Verification	Verification relies on centralized authorities	Decentralized verification through cryptographic proofs
User Autonomy	Users have limited control over their identities	Users have full ownership and autonomy over their identities
Transparency	Lack of transparency in data usage and access	Transparent and tamper-resistant data records on blockchain

Decentralized identity systems leverage blockchain technology and cryptographic principles to provide users with verifiable, tamper-proof, and privacy-preserving identity management. Unlike traditional models, decentralized identity frameworks

eliminate the need for a central authority, giving users full ownership and control over their identity credentials. These systems align with emerging standards such as the *World Wide Web Consortium* (W3C) *Decentralized Identifiers* (DIDs) and *Verifiable Credentials* (VCs), which define a framework for trustless and interoperable identity solutions [28].

2.1.1 Fundamental concepts of decentralized identity

Decentralized identity systems are built on several core principles that differentiate them from traditional identity models. These principles include **self-sovereignty**, **verifiability**, **privacy-preserving authentication**, and **interoperability**.

The concept of **self-sovereignty** ensures that individuals fully control their identity without reliance on third-party identity providers. In traditional identity systems, users must trust centralized authorities to issue and manage their credentials. In contrast, decentralized identity systems allow users to create and manage their own digital identities using cryptographic key pairs [29].

Verifiability is another fundamental property of decentralized identity systems. Using blockchain-based distributed ledgers, identities and credentials can be cryptographically signed and verified without requiring direct interaction with a centralized authority. Verifiable Credentials (VCs) enable entities to issue digital attestations, which can be selectively shared and verified by third parties without compromising user privacy [30].

Privacy-preserving authentication is achieved through techniques such as *zero-knowledge proofs* (ZKPs) and selective disclosure mechanisms, allowing users to prove certain attributes of their identity without revealing unnecessary personal information. These cryptographic techniques enhance security and minimize data exposure [31].

Interoperability is crucial for the adoption of decentralized identity systems. Standards such as W3C's DIDs and VCs ensure that decentralized identities can be used across different platforms, applications, and jurisdictions, facilitating a global identity ecosystem.

2.1.2 Decentralized identifiers (DIDs)

A **Decentralized Identifier (DID)** is a unique, persistent, and verifiable digital identity that is not dependent on a centralized registry or authority. DIDs are stored on decentralized networks, such as blockchain or distributed ledgers, ensuring immutability and resistance to censorship.

A DID consists of three main components:

`did:`*`method`*`:`*`identifier`*

where:

- **did**: A universal prefix indicating that the identifier conforms to the DID standard.
- **method**: A specific method that defines how the DID is created, resolved, and managed.
- **identifier**: A unique string that distinguishes a particular identity within the selected DID method.

For example, a DID might look like:

`did:example:123456789abcdefghi`

DIDs are controlled by cryptographic key pairs, where the owner of a DID possesses the private key required to authenticate and authorize identity-related transactions. The corresponding public key and associated metadata are stored in a **DID Document**, which is accessible on a decentralized ledger. This document contains information such as public keys, authentication methods, and service endpoints [28].

2.1.3 Verifiable credentials (VCs)

Verifiable Credentials (VCs) are digitally signed attestations that prove specific claims about an identity. These credentials are issued by a trusted entity and presented by the identity owner to verifiers when proving identity-related attributes. Unlike traditional identity credentials such as passports or driver's licenses, VCs allow for selective

disclosure, meaning users can reveal only necessary information while keeping other details private [30].

A Verifiable Credential consists of:

- **Issuer:** The entity that issues the credential.
- **Holder:** The user who owns and manages the credential.
- **Verifier:** The entity that verifies the authenticity of the credential.
- **Digital Signature:** A cryptographic signature that ensures the integrity and authenticity of the credential.

Verifiable Credentials are cryptographically signed using the issuer's private key, ensuring their authenticity and preventing forgery. The verification process involves checking the digital signature against the issuer's public key stored in a decentralized ledger.

2.1.4 Security and privacy considerations

Decentralized identity systems provide enhanced security and privacy protections compared to traditional identity models. However, they also introduce new challenges.

One of the key security advantages is **decentralization**, which eliminates single points of failure and reduces the risk of large-scale data breaches. Unlike centralized identity providers that store vast amounts of user data, decentralized identity systems distribute identity-related information across a decentralized network, reducing the impact of security compromises [31].

Another significant security benefit is the use of **cryptographic authentication**. Instead of relying on passwords, decentralized identity systems authenticate users using cryptographic key pairs, reducing the risk of phishing and credential theft.

However, decentralized identity systems also pose challenges, particularly concerning **key management**. Since users control their own cryptographic keys, losing access to private keys can result in permanent loss of identity credentials. Solutions such as

hierarchical deterministic (HD) key derivation and secure key recovery mechanisms are being explored to address these issues [30].

2.1.5 Applications of decentralized identity systems

Decentralized identity systems have a wide range of applications across various industries:

In **financial services**, decentralized identity can enable more secure and privacy-preserving *Know Your Customer* (KYC) and *Anti-Money Laundering* (AML) processes, reducing fraud and improving compliance.

In **healthcare**, decentralized identity solutions allow patients to control their medical records and share them securely with healthcare providers without relying on centralized repositories.

In **government services**, decentralized identity can be used for digital passports, national ID programs, and voting systems, ensuring transparency, security, and self-sovereign control over identity data.

Decentralized identity systems offer a paradigm shift from traditional, centralized identity management to a self-sovereign, privacy-preserving, and verifiable model. By leveraging cryptographic techniques, blockchain technology, and emerging standards such as DIDs and VCs, these systems provide enhanced security, interoperability, and user control. Despite their advantages, challenges such as key management, regulatory compliance, and standardization must be addressed for widespread adoption. Future research should focus on developing scalable and quantum-resistant identity solutions that maintain privacy and security in an evolving digital landscape.

2.2 BIP32: Hierarchical Deterministic Key Management

Cryptographic key management is a fundamental aspect of modern digital identity, blockchain, and cybersecurity systems. Traditional key management methods often require users to generate and store multiple private-public key pairs, leading to issues related to scalability, key loss, and security vulnerabilities. To address these challenges, the Bitcoin Improvement Proposal 32 (BIP32) introduced a Hierarchical Deterministic

(HD) key management framework, enabling the derivation of multiple cryptographic keys from a single master seed [6].

Traditional cryptocurrency wallets manage individual private keys independently, leading to complex backup and management procedures, especially when dealing with numerous addresses. BIP32 addresses this challenge by allowing the derivation of a virtually unlimited number of keys from a single master seed. This hierarchical structure simplifies key management, facilitates backups, and enables the creation of complex spending policies. The BIP32 algorithm has revolutionized key management by introducing a tree-like key hierarchy, making it easier to generate, store, and recover cryptographic keys securely. This section explores the foundations, structure, advantages, and security aspects of BIP32 and its applications in decentralized identity and blockchain-based systems.

2.2.1 Theoretical foundations of BIP32

2.2.1.1 Hierarchical deterministic key generation

Before the introduction of hierarchical deterministic wallets, key management relied on individual private-public key pairs, requiring users to manually store or back up each key separately. This approach was inefficient and prone to key loss [11]. BIP32 introduced deterministic key derivation, allowing multiple keys to be generated from a single master seed, significantly simplifying the backup and recovery process.

2.2.1.2 Key derivation in BIP32

BIP32 defines a hierarchical structure where cryptographic keys are generated using a deterministic function. The key derivation process is based on **HMAC-SHA512 hashing**, which ensures both security and uniqueness in key generation [12].

Master key generation process begins with the generation of a master seed s, which is a random value typically 128 to 256 bits in length. Using this seed, a master private key k_m and a master chain code c_m are derived using HMAC-SHA512:

$$I = \text{HMAC-SHA512}(\text{"Bitcoin seed"}, s) \tag{2.1}$$

where:

- The first 256 bits of I correspond to k_m (master private key).
- The last 256 bits correspond to c_m (master chain code).

The master public key K_m is then computed using elliptic curve multiplication:

$$K_m = P \times k_m \tag{2.2}$$

where P represents the elliptic curve base point.

For child key derivation, each child key is generated from a parent key using HMAC-SHA512. The derivation function follows:

$$I = \text{HMAC-SHA512}(c_{\text{parent}}, K_{\text{parent}} || \text{index}) \tag{2.3}$$

where:

- c_{parent} is the parent chain code.
- K_{parent} is the parent public key.
- The **index** determines whether the derived key is a **normal child key** (publicly derivable) or a **hardened child key** (requiring the private key for derivation).

The child private key k_i is then computed as:

$$k_i = (I_L + k_{\text{parent}}) \mod n \tag{2.4}$$

where I_L is the leftmost 256 bits of I, and n is the curve order.

2.2.1.3 Hierarchical structure and key path notation

BIP32 uses a hierarchical structure to organize keys in a tree format, enabling organized and role-based key delegation [17]. The hierarchical path notation follows:

```
m / 44′ / 0′ / 0′ / 0
```

where:

- m represents the **master key**.
- $44'$ denotes the **purpose identifier** (e.g., BIP44 for multi-currency wallets).
- $0'$ represents the **coin type** (e.g., Bitcoin, Ethereum).
- $0'$ refers to the **account number**.
- 0 indicates the **key index**.

2.2.2 Applications of BIP32

BIP32 is widely used in cryptocurrency wallets, where it enables deterministic wallet structures. Users can generate an entire wallet using a single seed, reducing the risk of key loss [11].

BIP32 is also being explored for IoT security, where devices require dynamic key generation without persistent storage. Its hierarchical structure enables devices to generate unique session keys, reducing long-term key exposure risks [19].

2.2.3 Security considerations

This architecture has its own advantages and potential risks. First, the advantages can be summarized as follows:

- **Key Isolation:** A compromised child key does not expose other keys.
- **Single Backup Requirement:** Only the master seed needs to be backed up.

- **Efficient Key Management:** Hierarchical structures simplify multi-user access control.

The potential risks can be summarized as follows:

- **Master Key Vulnerability:** If the master seed is compromised, all derived keys are at risk.
- **Lack of Post-Quantum Security:** BIP32 relies on elliptic curve cryptography, which is potentially vulnerable to quantum computing attacks [26].
- **Side-Channel Attacks:** Without hardware-based protection, attackers could extract keys through timing or memory attacks.

BIP32 has significantly improved cryptographic key management, providing a scalable and deterministic approach to hierarchical key derivation. Its applications extend beyond blockchain into decentralized identity management and IoT security. However, future research should focus on integrating post-quantum cryptographic mechanisms and improving security against hardware-based attacks.

2.3 SLIP-0010 and Its Comparison with BIP32

Hierarchical deterministic (HD) key derivation is a fundamental mechanism in blockchain and cryptographic systems for managing large numbers of cryptographic keys from a single root seed. The Bitcoin Improvement Proposal 32 (BIP32) introduced this concept using elliptic curve cryptography based on the secp256k1 curve [6]. However, BIP32 is tailored specifically to Bitcoin and uses curve-specific assumptions, limiting its flexibility for multi-curve cryptographic systems.

To address this limitation, the **SLIP-0010** specification, proposed by the Satoshilabs team, generalizes HD key derivation to support multiple elliptic curves beyond secp256k1, including Ed25519, which is widely used in modern cryptographic protocols such as SSH, TLS, and newer blockchain platforms [32].

This section presents an overview of the SLIP-0010 key derivation process and compares its properties, structure, and cryptographic assumptions with the BIP32 standard.

2.3.1 SLIP-0010 key derivation scheme

SLIP-0010 stands for *SatoshiLabs Improvement Proposal 0010*, and it defines a hierarchical deterministic key derivation scheme compatible with any elliptic curve that supports key generation using private scalars. Unlike BIP32, which is tightly coupled with the secp256k1 curve and allows both hardened and non-hardened derivation, SLIP-0010 only supports **hardened key derivation** for enhanced security and compatibility with curves like Ed25519.

2.3.1.1 Master key generation

SLIP-0010 begins with a master seed, typically between 128 and 512 bits of entropy. The seed is processed through a **curve-specific HMAC-SHA512** function to derive the master key and chain code.

For example, for the Ed25519 curve, the derivation is defined as:

$$\text{Key} \parallel \text{ChainCode} = \text{HMAC-SHA512}(\text{"ed25519 seed"}, \text{seed}) \tag{2.5}$$

Here, the left 256 bits of the output are used as the master private key, and the right 256 bits as the chain code.

2.3.1.2 Child key derivation

Child private keys are derived using the parent key and chain code, applying HMAC-SHA512 again with a fixed prefix:

$$I = \text{HMAC-SHA512}(c_{\text{parent}}, 0x00 \parallel k_{\text{parent}} \parallel \text{index}) \tag{2.6}$$

The output is split into the child key and new chain code, similar to BIP32, but with the restriction that all derivations are **hardened**. This ensures that public keys alone cannot derive child keys or compromise the tree structure.

2.3.2 Comparison between SLIP-0010 and BIP32

2.3.2.1 Supported curves and generality

A key distinction between BIP32 and SLIP-0010 lies in their cryptographic generality. BIP32 is explicitly designed for the secp256k1 elliptic curve, which is native to Bitcoin. It cannot be directly applied to other curves such as Ed25519 or NIST P-256 without substantial modifications.

In contrast, SLIP-0010 is curve-agnostic and supports any elliptic curve where key derivation from private scalars is possible. This includes Ed25519, which is increasingly adopted due to its performance and resistance to common implementation errors [33].

2.3.2.2 Hardened vs non-hardened derivation

BIP32 supports both **normal (non-hardened)** and **hardened** child key derivation. Non-hardened derivation allows public child keys to be derived from a parent public key, enabling useful wallet features such as watch-only wallets. However, it introduces

potential vulnerabilities where a compromised child private key and parent public key can reveal the parent private key [11].

SLIP-0010 **only allows hardened derivation**, removing this class of attack entirely and simplifying the implementation for curves that do not support non-hardened derivation, such as Ed25519.

2.3.2.3 Compatibility and use cases

BIP32 remains the standard for Bitcoin and Ethereum-compatible HD wallets due to its deep integration into legacy systems. SLIP-0010 is particularly useful for newer blockchain platforms (e.g., Solana, Polkadot), identity systems, and protocols requiring Ed25519 keys or other modern cryptographic primitives.

SLIP-0010 is also favored in systems that prioritize **security over flexibility**, as its exclusive use of hardened derivation avoids certain attack vectors and is easier to implement securely across different platforms and programming environments.

2.3.3 Security considerations

From a security standpoint, SLIP-0010's restriction to hardened keys provides a stronger isolation between nodes in the key hierarchy, reducing the risk of key compromise through mathematical inference. This makes SLIP-0010 attractive for use in environments where strong key isolation is required, such as self-sovereign identity systems and secure enclave key storage.

BIP32, while flexible and efficient in terms of key visibility and partial delegation, must carefully manage non-hardened derivation to prevent potential key leakage in case of a partial compromise.

SLIP-0010 generalizes hierarchical deterministic key derivation beyond the Bitcoin-specific BIP32 standard, enabling compatibility with a wider range of elliptic curves, particularly Ed25519. By enforcing hardened derivation and simplifying the derivation process, SLIP-0010 offers stronger security guarantees and broader applicability in modern cryptographic applications. While BIP32 continues to serve

legacy systems well, SLIP-0010 is increasingly adopted in decentralized identity frameworks, secure messaging, and next-generation blockchain platforms.

2.4 Hardware Security Modules (HSMs)

The security of cryptographic operations relies heavily on the protection of private keys and cryptographic materials. Hardware Security Modules (HSMs) are dedicated physical devices designed to ensure the confidentiality, integrity, and controlled usage of cryptographic keys. Unlike software-based key storage solutions, which are vulnerable to malware, side-channel attacks, and unauthorized access, HSMs provide a tamper-resistant environment that enforces strict security policies. These devices play a crucial role in securing financial transactions, cloud-based services, blockchain applications, and public key infrastructures (PKI).

The fundamental advantage of HSMs lies in their ability to perform cryptographic operations without ever exposing private keys outside the module. By isolating cryptographic processes within a hardware-protected boundary, HSMs significantly reduce the attack surface and mitigate risks associated with key compromise. Figure 2.1 summarizes the various features of HSM.

2.4.1 Cryptographic properties of HSMs

HSMs support a wide range of cryptographic operations, including key generation, encryption, decryption, digital signatures, and secure key wrapping. These operations are performed within a secure execution environment, ensuring that cryptographic keys are never available in plaintext outside the module. The cryptographic engine of an HSM typically includes hardware-accelerated implementations of symmetric encryption algorithms such as AES (Advanced Encryption Standard) and asymmetric encryption techniques such as RSA and Elliptic Curve Cryptography (ECC). Additionally, HSMs incorporate secure hash functions, such as SHA-256 and SHA-3, to support digital signatures and message authentication codes.

The key management capabilities of an HSM extend beyond simple storage. These devices enforce strict access controls, ensuring that only authorized entities can use cryptographic keys. Role-based authentication mechanisms restrict cryptographic

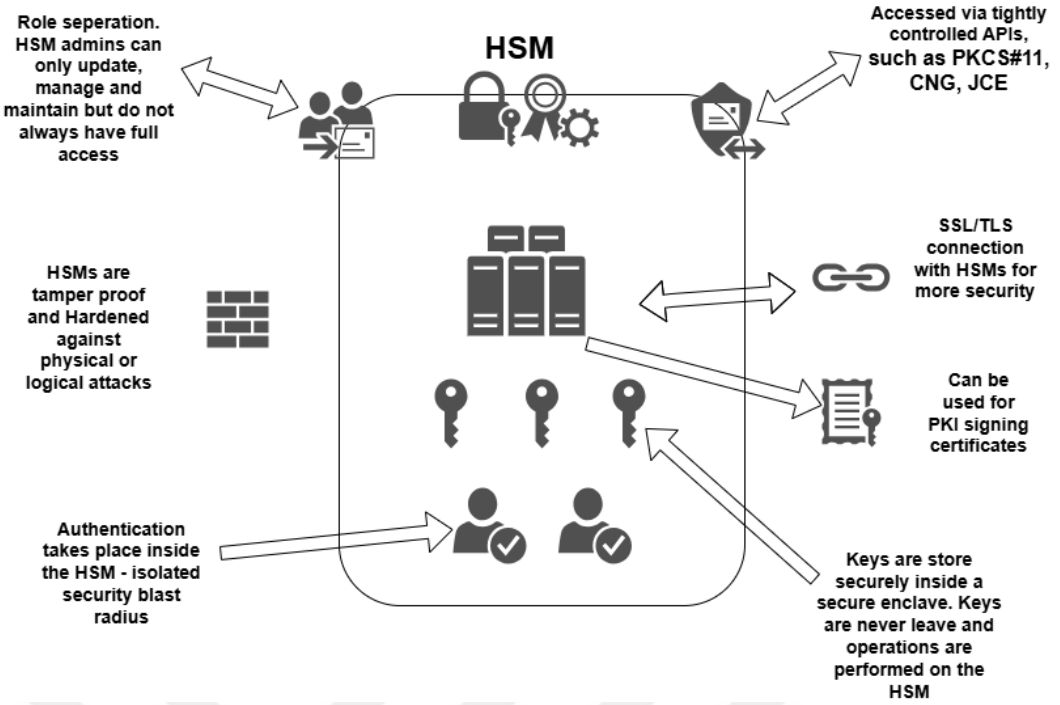

Figure 2.1 : HSM Features.

operations based on predefined policies, while multi-factor authentication techniques, such as smart cards and biometric verification, further enhance security.

To prevent unauthorized key extraction, HSMs implement countermeasures against physical and logical attacks. Techniques such as secure memory encryption, tamper detection circuits, and environmental monitoring ensure that sensitive cryptographic material is erased in case of tampering. In addition, HSMs employ differential power analysis (DPA) and electromagnetic shielding techniques to protect against side-channel attacks that attempt to infer key material from hardware emissions.

2.4.2 Industry standards and compliance

HSMs are designed to comply with stringent security standards to ensure their reliability and robustness in handling cryptographic materials. The most widely recognized standards governing HSM security include the Federal Information Processing Standard (FIPS) 140-2/3, the Common Criteria for Information Technology Security Evaluation, and the Public-Key Cryptography Standards (PKCS).

The FIPS 140-2 [34] and FIPS 140-3 [35] standards, established by the National Institute of Standards and Technology (NIST) [36], define security requirements

for cryptographic modules used in federal applications. These standards classify HSMs into four levels of security, ranging from Level 1, which provides basic cryptographic protection, to Level 4, which incorporates advanced tamper detection and response mechanisms. Compliance with FIPS 140-2/3 is mandatory for HSMs used in government and financial applications, ensuring that cryptographic modules meet rigorous security benchmarks.

The Common Criteria (CC) [37] is an internationally recognized standard for evaluating the security of information technology products, including HSMs. Under the Common Criteria framework, HSMs are assessed based on their security functional requirements and assurance levels. The Evaluation Assurance Level (EAL) system categorizes HSMs from EAL1 to EAL7, with higher levels indicating greater security assurance through extensive testing and verification. Many high-security HSMs are certified at EAL4+ or higher, demonstrating their resilience against sophisticated attack scenarios.

One of the most widely used interfaces for HSMs is PKCS#11 [38], a cryptographic token interface standard developed by RSA Security. PKCS#11 defines an API for applications to interact securely with cryptographic tokens, including HSMs and smart cards. This standard provides a consistent method for performing cryptographic operations such as key generation, encryption, decryption, and digital signing. By supporting PKCS#11, HSMs can seamlessly integrate with security applications, including TLS/SSL encryption, secure email communication, and certificate authorities.

In addition to PKCS#11, HSMs also support other cryptographic interfaces such as Microsoft Cryptographic API Next Generation (CNG) [39] and Java Cryptography Architecture (JCA) [40]. These APIs facilitate the use of HSMs in enterprise environments by providing secure key management and cryptographic operations across different platforms.

2.4.3 HSM integration in cryptographic systems

HSMs are widely integrated into cryptographic infrastructures to enhance security in various domains. In financial institutions, HSMs are used to protect credit card transactions, secure payment processing systems, and encrypt PIN codes. The Payment Card Industry Data Security Standard (PCI DSS) [41] mandates the use of HSMs in securing financial transactions to prevent unauthorized access to sensitive customer data.

In the blockchain ecosystem, HSMs provide secure storage for cryptocurrency private keys, mitigating the risk of key theft in digital wallets. The loss or compromise of private keys in blockchain applications can result in irreversible financial losses, making the use of HSMs a critical security measure. HSMs also facilitate secure multi-signature transactions by ensuring that cryptographic approvals are executed within a tamper-resistant environment.

Cloud service providers employ HSMs to protect sensitive data stored in cloud environments. Hardware Security as a Service (HSaaS) is a model in which cloud providers offer HSM functionality as an on-demand service, allowing organizations to perform cryptographic operations securely in the cloud. Leading cloud platforms such as AWS CloudHSM and Azure Key Vault HSM provide scalable cryptographic solutions for securing cloud-based workloads.

2.4.4 Security considerations and future developments

Despite their high level of security, HSMs are not immune to attacks. Potential vulnerabilities include insider threats, firmware exploitation, and supply chain attacks. To mitigate these risks, manufacturers implement strict access control mechanisms, regular firmware updates, and rigorous supply chain security protocols. Additionally, ongoing research focuses on enhancing HSM security against emerging threats, such as quantum computing.

Quantum computing poses a significant challenge to existing cryptographic standards, particularly asymmetric algorithms such as RSA and ECC, which rely on the difficulty

of factoring large prime numbers or solving discrete logarithm problems. As quantum computing capabilities advance, post-quantum cryptographic solutions are being explored to ensure the long-term security of HSMs. Some modern HSMs are beginning to integrate support for quantum-resistant algorithms, including lattice-based and hash-based cryptographic schemes.

Hardware Security Modules (HSMs) provide a robust and secure environment for cryptographic operations, ensuring the confidentiality and integrity of cryptographic keys. By adhering to industry standards such as FIPS 140-2/3, Common Criteria, and PKCS#11, HSMs deliver high-assurance security for applications in financial transactions, blockchain security, cloud computing, and public key infrastructures. While HSMs offer unparalleled security advantages, continued research and development are necessary to address emerging threats, particularly in the era of quantum computing. The integration of post-quantum cryptographic algorithms into HSMs will be a crucial step in future-proofing cryptographic security.

2.5 The PKCS#11 Standard: A Unified Cryptographic Token Interface

The PKCS#11 standard, officially known as the Cryptographic Token Interface Standard, provides a platform-independent API to enable secure and standardized interactions between software applications and cryptographic hardware such as Hardware Security Modules (HSMs) and smart cards. Initially developed by RSA Laboratories and now maintained by OASIS, the standard abstracts essential cryptographic functionalities—including key management, digital signatures, encryption, and hashing—allowing sensitive materials such as private keys to remain securely stored within the token's environment [42]. The architecture is grounded in a slot-token model: a *slot* represents the logical interface or reader, while a *token* denotes the physical or virtual cryptographic module. Application-level interaction is organized through sessions and defined roles (user and security officer), employing PIN-based authentication to control access to private objects. PKCS#11 categorizes its operations via cryptographic mechanisms (algorithms), objects (e.g., keys and certificates), and well-structured function calls (e.g., `C_Sign`, `C_Decrypt`) [43]. Widely used in secure communications, identity management, and cloud-based HSM infrastructures, it facilitates high assurance operations in industries requiring secure key custody and cryptographic interoperability [44]. Despite its benefits, implementation flaws—such as improper role separation or insecure PIN retry logic—have exposed some systems to attacks, underscoring the importance of robust conformance testing and secure configuration [45]. The standard's evolution has brought support for modern primitives like AES-GCM and elliptic curve cryptography (ECC), ensuring continued relevance in modern security ecosystems [15]. As a result, PKCS#11 remains foundational in the cryptographic stack, offering a practical and secure interface for diverse application needs.

2.6 Cryptographic Key Wrapping Techniques

Key management is a fundamental component of cryptographic security, ensuring that encryption keys remain protected throughout their lifecycle. One of the most critical aspects of key management is secure key storage and transmission, which prevents unauthorized access and tampering. **Key wrapping** is a cryptographic technique designed to protect encryption keys by encapsulating them using a symmetric encryption algorithm, ensuring their confidentiality and integrity [46]. Unlike conventional encryption methods, key wrapping is specifically optimized for securing cryptographic keys, incorporating additional mechanisms to ensure resistance against brute-force attacks and unauthorized modification.

Key wrapping plays a crucial role in applications such as secure key exchange, cloud key management, hardware security modules (HSMs), and cryptographic key backup systems. This section explores the principles of key wrapping, the cryptographic algorithms used, and the security properties required to ensure robust protection. Figure 2.2 summarizes the key wrap and unwrap operations.

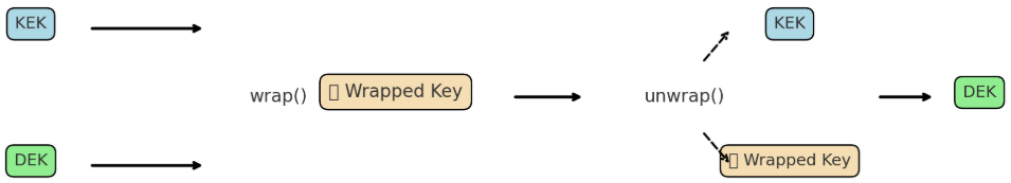

Figure 2.2 : Key wrap-unwrap diagram.

2.6.1 Principles of key wrapping

Key wrapping involves encrypting a cryptographic key (known as the *key to be wrapped*) using a *key encryption key* (KEK). This process ensures that the protected key remains confidential and tamper-proof. The KEK is typically a long-term symmetric key, often stored in a secure environment such as an HSM or a trusted execution environment (TEE) [47].

A key wrapping scheme must satisfy the following security properties:

- **Confidentiality:** The wrapped key must remain confidential, ensuring that only authorized entities with the correct KEK can unwrap it.
- **Integrity:** The wrapping mechanism should provide integrity protection, preventing unauthorized modifications that could lead to key compromise.
- **Deterministic or Randomized Output:** Some key wrapping schemes produce deterministic ciphertexts, while others incorporate randomization to enhance security.
- **Resistance to Cryptanalysis:** The wrapping scheme must be resilient against brute-force attacks, chosen-ciphertext attacks, and related-key attacks.

Key wrapping is commonly used in scenarios where cryptographic keys must be securely stored in external environments or transmitted over insecure channels. In such cases, even if the wrapped key is exposed, its confidentiality remains protected as long as the KEK is securely stored.

2.6.2 Key wrapping algorithms

Several standardized key wrapping algorithms have been developed to meet different security requirements. The most widely used algorithms include:

2.6.2.1 AES key wrap (AES-KW)

The **AES Key Wrap** algorithm, standardized in NIST Special Publication 800-38F, is a widely adopted method for protecting cryptographic keys using the Advanced Encryption Standard (AES) [46]. AES-KW operates in a deterministic manner, using AES in ECB mode along with an integrity check mechanism. The core algorithm consists of multiple rounds of encryption, where the key material is divided into blocks and transformed using AES encryption and XOR operations.

Given a plaintext key P of n blocks and an AES key K, AES-KW operates as follows:

$$A_0 = IV, \quad R_i = P_i \quad \text{for } i = 1 \text{ to } n \tag{2.7}$$

where IV is a predefined integrity check value. The algorithm iteratively applies AES encryption and XOR transformations to produce a wrapped key.

AES-KW provides a high level of security, ensuring both confidentiality and integrity. However, it is not suitable for encrypting large amounts of data and is specifically designed for key wrapping purposes.

2.6.2.2 AES key wrap with padding (AES-KWP)

The **AES Key Wrap with Padding** (AES-KWP) is an extension of AES-KW that supports wrapping keys of arbitrary lengths [48]. Unlike AES-KW, which requires input keys to be a multiple of 64 bits, AES-KWP introduces a padding mechanism that allows wrapping shorter or non-aligned keys.

The AES-KWP algorithm is used in environments where variable-length keys need to be securely protected, such as cloud-based key management systems and encrypted database storage.

2.6.2.3 Secure key wrap mode (SKWM)

Secure Key Wrap Mode (SKWM) is a generalization of key wrapping techniques that incorporates authenticated encryption, ensuring that wrapped keys are protected against both confidentiality and integrity attacks. SKWM typically utilizes AES-GCM (Galois/Counter Mode) or AES-CCM (Counter with CBC-MAC) to provide both encryption and authentication in a single operation [49].

Unlike traditional AES-KW, which relies solely on deterministic transformations, SKWM introduces randomization through initialization vectors (IVs) or nonces, making it resistant to chosen-ciphertext and related-key attacks.

2.6.3 Key unwrapping and verification

The process of key unwrapping is the inverse of key wrapping, where the wrapped key is decrypted using the KEK. The unwrapping process typically includes an integrity

verification step to detect any tampering. If an attacker modifies a wrapped key during transmission or storage, the integrity check ensures that the key is rejected rather than producing an incorrect decryption result.

For AES-KW, integrity is verified by checking the integrity check value (ICV) after decryption. If the ICV does not match the expected value, the wrapped key is considered invalid. In AES-KWP and SKWM, integrity verification is performed using cryptographic message authentication codes (MACs) or authenticated encryption mechanisms.

2.6.4 Applications of key wrapping

Key wrapping techniques are used in various security-critical applications:

Key wrapping is fundamental in **Hardware Security Modules (HSMs)**, where sensitive cryptographic keys are stored securely and exported in a wrapped format. HSMs generate and manage KEKs to ensure that wrapped keys remain protected even when stored outside the module.

In **cloud key management systems**, key wrapping is used to encrypt and securely store cryptographic keys in cloud environments. Cloud providers implement key wrapping to ensure that even if encrypted key storage is compromised, the underlying key material remains inaccessible without the KEK.

Key wrapping is also crucial in **blockchain-based security models**. In cryptocurrency wallets and blockchain infrastructure, private keys are often stored in wrapped form to prevent unauthorized access. Secure key wrapping ensures that only authorized entities can access and use cryptographic keys for signing blockchain transactions.

2.6.5 Security considerations

Despite its advantages, key wrapping is not immune to attacks. If an attacker gains access to the KEK, all wrapped keys can be decrypted. Therefore, KEKs must be stored in highly secure environments such as HSMs or secure enclaves. Additionally, key wrapping schemes must be resistant to side-channel attacks, which can extract cryptographic material through power analysis or timing-based attacks [49].

Future advancements in post-quantum cryptography may influence the design of key wrapping schemes, particularly in ensuring resilience against quantum computing threats.

Key wrapping is a vital technique in modern cryptographic systems, ensuring the secure storage and transmission of cryptographic keys. Standardized algorithms such as AES-KW, AES-KWP, and Secure Key Wrap Mode provide robust security guarantees, protecting cryptographic material from unauthorized access and tampering. As security threats evolve, research continues to enhance key wrapping mechanisms to withstand emerging attack vectors, including quantum computing and advanced side-channel attacks.

3. A SECURE AND SCALABLE KEY MANAGEMENT FRAMEWORK

3.1 Overview

This section presents a detailed description of our proposed framework for scalable and secure key management in large-scale decentralized identity systems. The framework leverages the BIP32 algorithm, Hardware Security Modules (HSMs), and key wrapping techniques to address the challenges of complexity, scalability, security isolation, recovery, and secure delegation.

In national digital identity systems, the use of multiple cryptographic keys for a user is essential for enhancing security and privacy. By isolating different operations such as authentication, signing, and encryption, these systems ensure that each function is securely compartmentalized, reducing the risk of cross-contamination in the event of a key compromise. This isolation also supports privacy by preventing the linking of actions across different contexts, thereby safeguarding user anonymity. Furthermore, the use of distinct keys enables the delegation of authority without compromising the user's primary keys, allowing for flexible and secure management of digital identities. Key rotation and revocation are also facilitated, providing resilience and continuity in the face of potential key compromises. Additionally, securing communication channels with different keys for various contexts or operations aligns with the principles of security, privacy, and control. These principles are fundamental to decentralized identity systems, particularly in environments where users must maintain sovereignty over their digital identities.

In order to meet the need for so many keys in a large-scale system and to ensure the security of these keys, a framework is proposed. This framework uses the BIP32 algorithm to reduce the number of keys and facilitate key management, while using HSM to store keys securely against possible attacks. In order to securely store a large

number of keys with HSM, a key wrapping/unwrapping method is presented. Figure 3.1 summarizes the proposed architecture.

3.1.1 Framework components

3.1.1.1 Hardware security module (HSM)

- Serves as the root of trust for the system
- Generates and stores the Key Encryption Key (KEK)
- Performs cryptographic operations, including:
  - Key wrapping and unwrapping
  - Secure random number generation
  - BIP32 algorithm execution for deriving hierarchical deterministic keys

3.1.1.2 Key encryption key (KEK)

- A high-entropy key generated and stored securely within the HSM
- Used for wrapping (encrypting) and unwrapping (decrypting) master seeds

3.1.1.3 Master seed

- Unique to each user in the system
- Generated within the HSM using its secure random number generator
- Serves as the root for deriving all other keys using the BIP32 algorithm

3.1.1.4 Encrypted master seed storage

- A secure database for storing wrapped (encrypted) master seeds
- Located outside the HSM for scalability

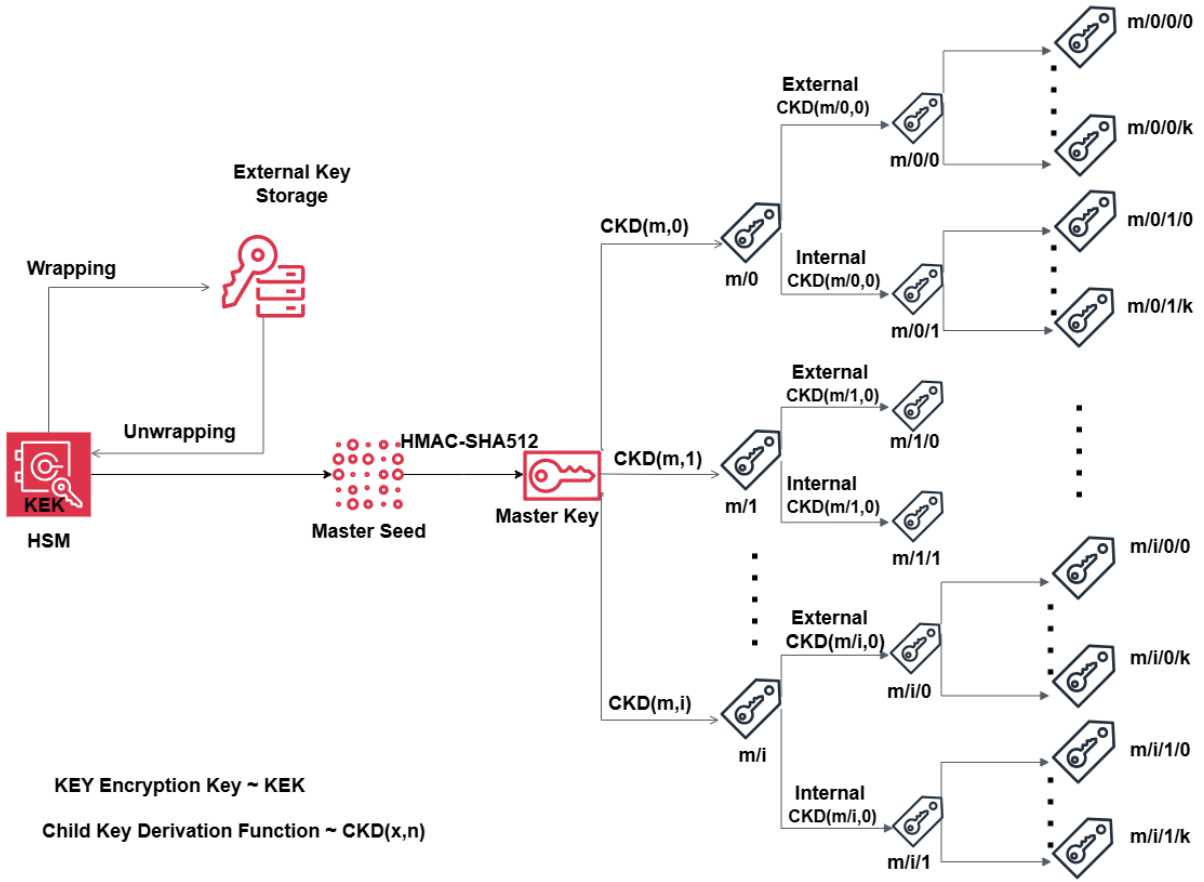

Figure 3.1 : A secure and scalable decentralized key management framework.

This architecture has been tested in real life using an HSM that supports BIP32 and SLIP10 algorithms. Performance tests were performed for both algorithms and compared and evaluated according to certain metrics.

3.2 Key Management Workflow

The proposed architecture integrates HD wallets with HSMs for institutional custody. The workflow includes:

1. Master Seed Generation: The HSM generates and stores the master seed securely.
2. Secure Key Derivation: Child keys are derived within the HSM, never exposed externally.
3. Transaction Signing: The HSM signs transactions, outputting only signed data.

In order to perform all of these operations, an extension was written for these algorithms in the PKCS11 standard [38], which is the interface supported by HSMs, and the necessary test codes were written using these extension functions.

3.2.1 Key generation and storage

The HSM-Backed Master Seed Generation algorithm is responsible for securely generating and storing the cryptographic root of the hierarchical key structure. This process initiates the framework's trust anchor by generating a high-entropy master seed within a Hardware Security Module (HSM), utilizing a Secure Random Number Generator (SRNG). The generated seed is stored securely within the HSM and used to derive the master extended private key according to the BIP32 or SLIP10 standard. Additionally, for backup purposes, the seed is wrapped using a Key Encryption Key (KEK) and exported in encrypted form. The algorithm ensures that the master seed never leaves the HSM in plaintext, preserving confidentiality and integrity from the outset.

Algorithm 1 HSM-Backed Master Seed Generation

Require: Secure Random Number Generator (SRNG)
Ensure: Securely stored master seed
1: Generate master seed inside HSM using SRNG.
2: Store master seed securely inside HSM.
3: Derive master extended private key using BIP32/SLIP10.
4: Wrap master seed with KEK for external storage.
5: Return wrapped master seed for backup.

3.2.2 Hierarchical key derivation

The HSM-Based Key Derivation algorithm facilitates the deterministic derivation of child private keys from a single master private key, following the specifications of BIP32 or SLIP10. This mechanism is essential for supporting scalable identity or account structures within the framework, where multiple keys are required without compromising security. The algorithm takes as input the master private key and a child index and computes an HMAC-SHA512-based output, which is then partitioned and mathematically combined to yield the derived private key and an updated chain code. All computations are performed within the HSM, ensuring that private key material remains protected throughout the derivation process.

Algorithm 2 HSM-Based Key Derivation (BIP32/SLIP10)

Require: Master Private Key (MPK), Child Index (CI)
Ensure: Derived Private Key (DPK)
1: Compute HMAC-SHA512 hash of (MPK, CI).
2: Split hash output into IL (left) and IR (right).
3: Compute $(DPK = (IL \times G) + MPK \mod n)$.
4: Return DPK and updated chain code.

3.2.3 Secure transaction signing

The HSM-Based Transaction Signing algorithm enables the secure digital signing of transactions without exposing private key material outside the HSM boundary. This function is invoked whenever a cryptographic operation, such as identity proof or transaction authorization, is required. The algorithm accepts transaction data and a corresponding private key as inputs. It then generates a transaction hash and

produces a digital signature using either the ECDSA algorithm (as defined in BIP32) or EdDSA (as specified in SLIP10), depending on the key format. By performing signing operations entirely within the HSM, the algorithm maintains strong guarantees of authenticity and non-repudiation.

Algorithm 3 HSM-Based Transaction Signing

Require: Transaction Data (TX), Private Key (PK)
Ensure: Signed Transaction
1: Retrieve private key from HSM.
2: Generate transaction hash.
3: Sign transaction hash using ECDSA (BIP32) or EdDSA (SLIP10).
4: Return signed transaction.

3.2.4 Key wrapping and recovery

The Key Wrapping and Secure Storage algorithm ensures the safe external storage and future recovery of sensitive key material, particularly the master seed. This mechanism is critical for enabling disaster recovery and backup processes without compromising cryptographic integrity. The algorithm encrypts the master seed using AES-256 encryption in combination with a Key Encryption Key (KEK), producing a securely wrapped key object. This object is stored in an external vault or storage system. After encryption, the plaintext seed is immediately purged from memory to prevent leakage. The wrapped key can later be retrieved and unwrapped within a secure environment, maintaining the confidentiality and usability of the original seed.

Algorithm 4 Key Wrapping and Secure Storage

Require: Master Seed (MS)
Ensure: Securely stored wrapped key
1: Encrypt MS using AES-256 with KEK.
2: Store wrapped key in external vault.
3: Erase MS from memory.
4: Return wrapped key for future retrieval.

3.2.5 User registration

Since each user will have their own master seed value, the user registration process starts when a request to generate a new master seed is sent to the HSM. Figure 3.2 shows the user registration flow diagram. The subsequent processes are as follows:

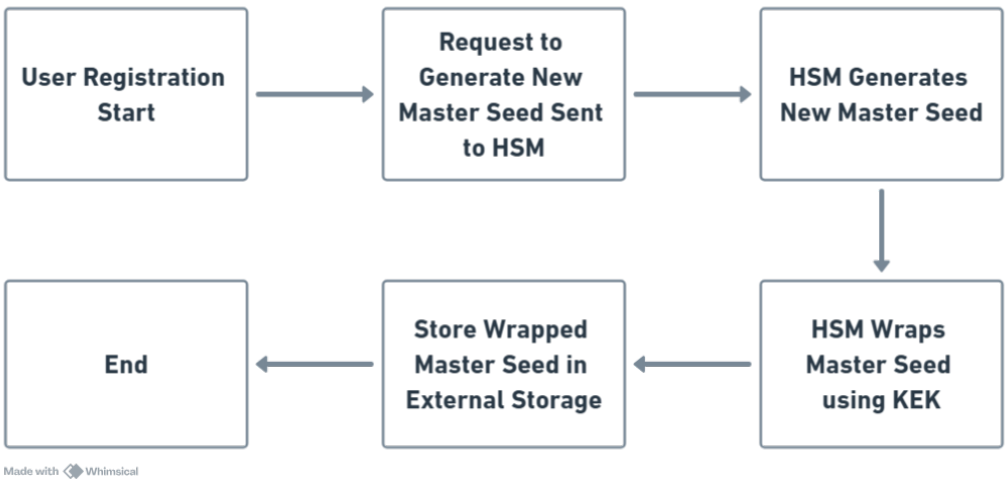

Figure 3.2 : User registration flow.

1. A new master seed is generated for the user.
2. The master seed is sent to the HSM for wrapping.
3. The HSM wraps the master seed using the KEK.
4. The wrapped master seed is stored in the encrypted master seed storage.

3.2.6 Key derivation

The key derivation process is initiated when a key is needed and starts by retrieving the wrapped master seed from external storage. Figure 3.3 shows the key derivation flow diagram.

1. When a key is needed, the wrapped master seed is retrieved from storage.
2. The wrapped master seed is sent to the HSM for unwrapping.
3. The HSM unwraps the master seed using the KEK.
4. The unwrapped master seed is used by the BIP32 module to derive the necessary keys.
5. Derived keys are used for the required cryptographic operations.
6. The unwrapped master seed is securely erased from memory after use.

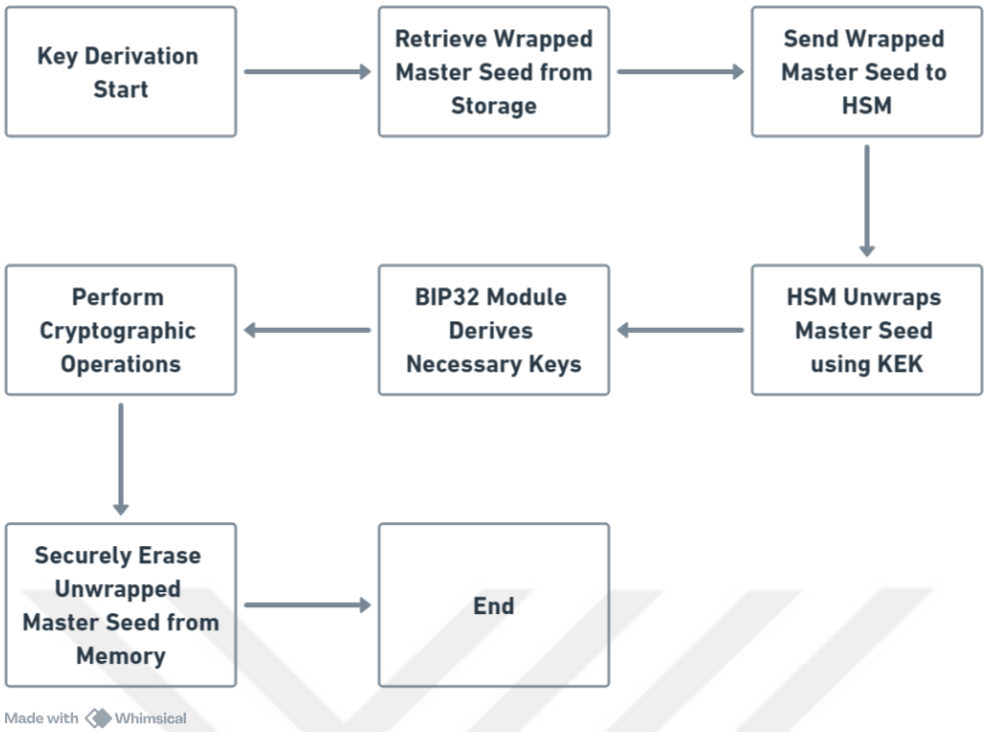

Figure 3.3 : Key derivation flow.

3.2.7 Key recovery

The key recovery process provides a robust mechanism for handling key loss scenarios by leveraging the stored wrapped master seed. Figure 3.4 shows the key recovery flow diagram.

1. In case of key loss, the wrapped master seed can be retrieved from storage.
2. The recovery process follows the same steps as key derivation.
3. All necessary keys can be re-derived from the master seed.

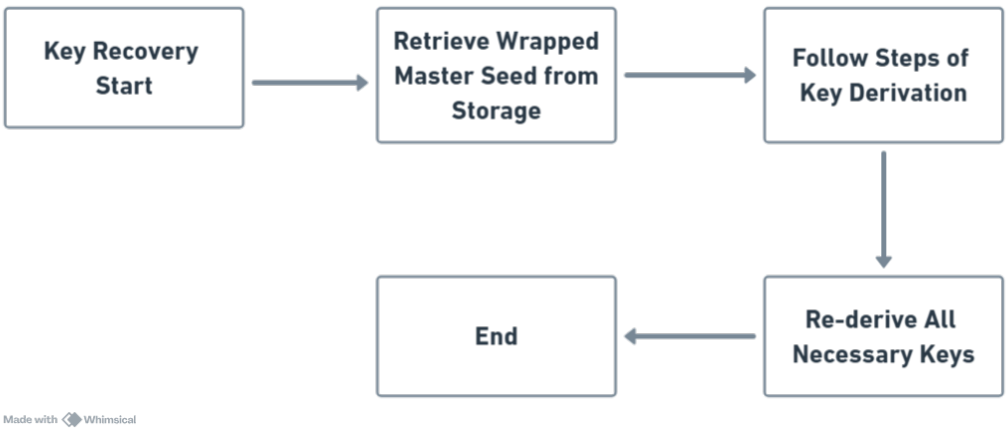

Figure 3.4 : Key recovery flow.

3.3 BIP32 Hierarchical Deterministic Key Management

The BIP32 standard enables the generation of multiple cryptographic keys from a single master key, providing significant advantages in key management efficiency and scalability. In the context of a national digital identity system, BIP32 is used to generate derived keys for various purposes, including user authentication, document signing, and encryption.

3.3.1 Master seed

The master seed is the root of all keys in the BIP32 hierarchy. Compromising the master seed would allow an attacker to regenerate all derived keys, compromising the entire identity system. Master keys in the system are also generated from master seeds.

Algorithm 5 BIP32 Master Key Generation [6]

Input: master seed
Output: master_private_key, master_chain_code
begin
    hmac ← HMAC-SHA512("Bitcoin seed", seed)
    master_private_key ← hmac[0:255]
    master_chain_code ← hmac[256:511]
    return master_private_key, master_chain_code
end

3.3.2 Derived keys

Derived keys are generated from the master key using hierarchical deterministic (HD) key derivation. Each key is associated with a unique purpose, such as:

- **Authentication Keys:** Used to verify a user's identity.
- **Signing Keys:** Used to sign digital documents or transactions.
- **Encryption Keys:** Used to encrypt sensitive information.
- **Pseudonymous Transaction Keys:** To maintain privacy, users may generate new, pseudonymous keys for each transaction. This prevents transaction linking, which would otherwise reveal patterns or details about the user's identity.

BIP32 allows for the independent generation of each derived key, ensuring key isolation and preventing the compromise of one key from affecting others.

In BIP32, a root private key k and chain code c are generated using HMAC-SHA512 and seed s. After that, the root public key K is generated by multiplying the root private key k and elliptic curve base point P. The derivation of a child private key k_{child} from a parent private key k_{par} is given as:

$$I = \text{HMAC-SHA512}(c_{\text{par}}, k_{\text{par}} || \text{index}_{\text{child}}) = (I_L || I_R) \tag{3.1}$$

$$k_i = P(I_L) + k_{\text{par}} \mod n \tag{3.2}$$

$$c_{\text{child}} = I_R \tag{3.3}$$

where:

- HMAC-SHA512 is the HMAC function using the SHA-512 hash algorithm.
- **Parent Key** is the private key of the parent node in the HD tree.

- **Child Index** is an integer representing the position of the child key in the hierarchy.
- **Chain Code** is a unique value used to generate child keys securely without compromising the parent key.

Algorithm 6 BIP32 Child Key Derivation [6]

Input: parent_key, child_index, chain_code
Output: child_private_key, new_chain_code
begin
    data ← child_index ‖ chain_code
    hmac ← HMAC-SHA512(parent_key, data)
    child_private_key ← hmac[0:255]
    new_chain_code ← hmac[256:511]
    return child_private_key, new_chain_code
end

Algorithm 7 BIP32 Extended Key Derivation [6]

Input: private_key, chain_code
Output: extended_private_key, extended_public_key
begin
    extended_private_key ← (private_key, chain_code)
    public_key ← ComputePublicKey(private_key)
    extended_public_key ← (public_key, chain_code)
    return extended_private_key, extended_public_key
end

3.3.3 Key management efficiency

The hierarchical structure of BIP32 enables efficient key management by allowing the system to derive new keys on demand, without the need to store all keys. Only the master seed needs to be backed up, reducing complexity and overhead in managing keys for millions of users.

3.4 Hardware Security Module (HSM) Integration

The Hardware Security Module (HSM) is a tamper-resistant hardware device responsible for securely managing the master seed and performing critical cryptographic operations, including key generation, key wrapping, and unwrapping. Figure 3.5 shows the position and role of the HSM in the proposed key management framework.

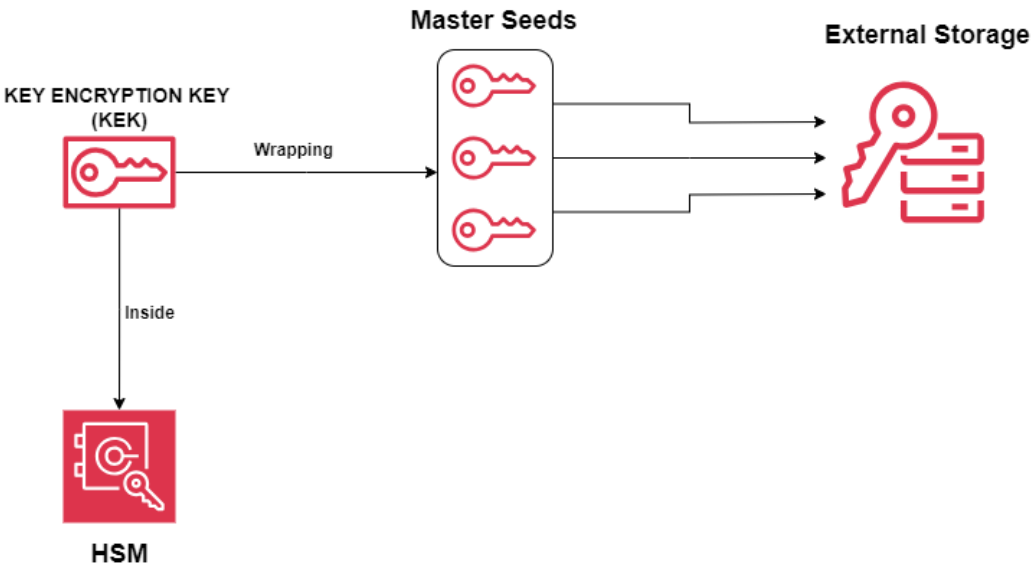

Figure 3.5 : HSM-KEK Structure.

3.4.1 Master seed

The master seed is generated inside the HSM when a new identity is created. This seed is the root of all future cryptographic keys (authentication, signing, encryption, etc.). The master seed is never exported from the HSM in plaintext.

3.4.2 Key encryption key (KEK)

The HSM generates and stores a Key Encryption Key (KEK), which is used to encrypt (wrap) and decrypt (unwrap) master seeds before they are stored or retrieved from external storage. The KEK is securely stored within the HSM and never exposed externally. It is recommended to change this key periodically in order to maintain the security of the system. When the KEK is updated, the master seeds wrapped with this key should be unwrapped with the old KEK and wrapped with the new KEK.

3.4.3 Tamper-resistant hardware protection

HSMs are specialized hardware devices designed to store cryptographic material, such as key encryption keys (KEK) or other critical keys, in a tamper-resistant environment. The HSM has tamper-evident and tamper-proof mechanisms. If an attacker physically tries to tamper with the HSM to extract the KEK, the device will detect the tampering

attempt and trigger security mechanisms such as erasing the stored KEK or shutting down the module. This makes HSMs highly secure against physical attacks, ensuring that the KEK remains protected from unauthorized access even if an attacker has physical access to the HSM.

3.4.4 Controlled access and role-based permissions

The HSM enforces strict role-based access control (RBAC) policies, ensuring that only authorized users or systems can access key management operations. Only authorized administrators can manage the HSM. Only specific operations (e.g., key derivation or signing) can be performed with the master seed inside the HSM. This mitigates insider threats by restricting access to critical cryptographic functions. The KEK is never directly exposed or extracted from the HSM, preventing it from being stolen by malware or insiders.

3.4.5 Secure key operations

One of the main benefits of HSMs is that they allow cryptographic operations (such as key derivation) to occur within the secure boundaries of the hardware. Operations like key wrapping, signing, encryption, and key derivation occur inside the HSM, ensuring that the master seed is protected from external threats.

3.4.6 Mitigation of insider threats

By storing the KEK within an HSM, the risk of insider attacks is minimized. Even privileged users (like system administrators) are restricted from directly accessing the KEK. All sensitive operations involving the seed are logged and monitored within the HSM. This audit trail can be used to detect any suspicious activities, adding an extra layer of security against internal threats.

3.5 Key Wrapping and Scalable External Storage

To enable the system to scale and securely manage large numbers of keys, master seeds are wrapped using the Key Encryption Key (KEK) and stored in external environments. This allows the system to offload the storage burden from the Hardware Security

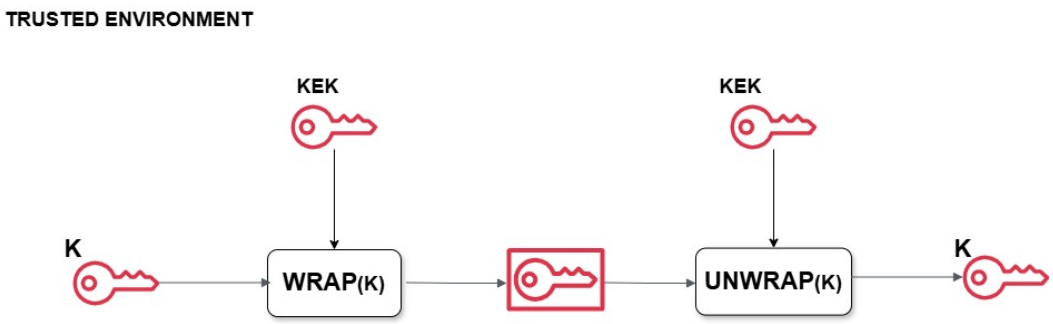

Figure 3.6 : HSM wrap unwrap mechanism.

Module (HSM) while ensuring that the keys remain secure even when stored outside of the hardware module. Figure 3.6 summarizes the key wrap-unwrap mechanism.

3.5.1 Key wrapping

The process of key wrapping involves encrypting the master seeds with the KEK inside the HSM. Once wrapped, the master seeds can be securely stored in external locations, such as secure cloud storage or databases. Wrapped master seeds are safe to store externally because they can only be unwrapped by the HSM, which holds the KEK, as shown in Equation 3.4.

$$\text{Wrapped Master Seed} = \text{Wrap}_{KEK}(\text{Master Seed}) \tag{3.4}$$

3.5.2 External storage

The wrapped master seeds are stored in external storage solutions, such as cloud databases or distributed storage systems. Even if an attacker gains access to this storage, they cannot use the wrapped master seeds without access to the HSM and its KEK.

3.5.3 Key retrieval and unwrapping

When a key is required for an operation, such as signing a document or authenticating a user, the wrapped master seed is retrieved from external storage and sent to the HSM. The HSM then unwraps the master seed using the KEK and makes it available for the requested cryptographic operation, as shown in Equation 3.5.

$$\text{Master Seed} = \text{Unwrap}_{KEK}(\text{Wrapped Master Seed}) \tag{3.5}$$

3.5.4 Scalability

The use of key wrapping allows the system to scale to handle millions of keys, as only the wrapped versions of master seeds are stored externally. This reduces the load on the HSM, which only stores the KEK and performs key wrapping/unwrapping operations, thus enabling the system to support large user bases.

3.6 Addressing Key Management Challenges

The proposed framework effectively addresses the key management challenges inherent in large-scale decentralized identity systems. In terms of complexity, the framework significantly simplifies key management for users. By leveraging the BIP32 algorithm, users only need to manage a single master seed, from which all other keys are derived. This structured approach to key derivation dramatically reduces the complexity of key management, making the system more user-friendly and less prone to user error.

Scalability, a critical concern for national-level identity systems, is achieved through a clever use of HSMs and external storage. Wrapped master seeds are stored outside the HSM, allowing for virtually unlimited scalable storage. The HSM itself is used solely for the computationally intensive but infrequent operations of wrapping and unwrapping seeds, rather than for storing individual user keys. This approach allows the system to scale efficiently to millions of users without compromising security.

Security isolation, another crucial aspect of key management, is inherently supported by the BIP32 algorithm. It allows for the derivation of separate key hierarchies for different purposes, enhancing the overall security of the system. Importantly, the compromise of a derived key does not compromise the master seed or other derived keys, containing potential security breaches and preserving the integrity of the overall system.

The framework provides robust mechanisms for key recovery and key revocation, ensuring that users can recover their identities in case of key loss or revocation

of compromised keys. Key revocation, if a derived key is compromised (e.g., an authentication key), the system can mark that key as revoked. A new key can be derived from the master key to replace the compromised one, ensuring continuity of operations while preventing the use of the revoked key. The framework also simplifies the processes of recovery and backup. Since only the master seed needs to be backed up, the recovery process is streamlined and less prone to errors. The wrapped master seed can be securely stored and easily recovered when needed, providing a robust solution for key recovery in case of loss or system failure.

Lastly, the framework supports secure delegation, a feature crucial for many identity management scenarios. The hierarchical nature of BIP32 allows for secure delegation of derived keys without exposing the master seed. Parent keys can derive child keys, enabling hierarchical access control. This feature is particularly useful in organizational settings where different levels of access and authority need to be managed securely.

In conclusion, this framework provides a robust foundation for managing cryptographic keys in large-scale decentralized identity systems. By addressing the key challenges of complexity, scalability, security isolation, recovery, revocation and secure delegation, while maintaining a high level of security, the proposed framework offers a comprehensive solution for the future of decentralized identity management.

4. SECURITY ANALYSIS

This chapter provides a comprehensive security analysis of the proposed key management framework for decentralized identity.

4.1 Assumptions

The security analysis of the proposed framework is underpinned by several key assumptions. Firstly, it is assumed that the Hardware Security Module (HSM) employed in the architecture is tamper-resistant and operates correctly. This assumption is considered critical as the HSM is relied upon as the root of trust in the system, with the most sensitive cryptographic operations being performed within it.

The security of fundamental cryptographic primitives, such as the Advanced Encryption Standard (AES) and the SHA-256 hash function, is also assumed. These algorithms are widely utilized and have been subjected to extensive scrutiny by the cryptographic community. The assumption of their security is based on current cryptographic knowledge and the absence of known practical attacks against these primitives.

A crucial assumption for the integrity of the system is that the confidentiality of the Key Encryption Key (KEK) is maintained within the HSM. The KEK is regarded as central to the key wrapping mechanism, and its confidentiality is considered paramount to the security of all wrapped master seeds in the system. It is expected that the HSM's secure design and operational procedures will ensure this confidentiality.

Lastly, it is assumed that the random number generator within the HSM is truly random and secure. This assumption is deemed vital for the generation of cryptographically strong master seeds and other keying material. It is recognized that a compromised or predictable random number generator could potentially undermine the entire key

generation process, making this assumption critical to the overall security of the system.

Table 4.1 : Security Assumptions and Their Consequences if Violated.

Assumption	Description	Consequences if Violated
HSM Tamper-Resistance	The HSM operates correctly and is tamper-resistant	Compromise of root of trust; key exposure
Cryptographic Primitive Security	AES and SHA-256 are secure against known practical attacks	Encryption and integrity protection failure
KEK Confidentiality	KEK remains confidential within the HSM	Exposure of wrapped master seeds
Secure Random Number Generation	RNG in HSM is truly random and secure	Weak master seeds; compromise of all derived keys

4.1.1 Attacks and countermeasures

The proposed key management framework for large-scale decentralized identity management incorporates multiple layers of security to protect against a wide array of potential threats. This section presents a comprehensive analysis of potential attack vectors, how these attacks might be executed, and the corresponding countermeasures implemented within the framework.

4.1.1.1 Physical and side-channel attacks on HSMs

Potential Attacks: Physical attacks involve attempts to directly access the internal components of the HSM through methods such as drilling, chemical etching, or fault injection. Side-channel attacks exploit information leaked during the HSM's operation, such as power consumption patterns or electromagnetic emissions, to infer sensitive data.

Countermeasures: The proposed framework mitigates these attacks through the use of HSMs certified to FIPS 140-2 Level 4 or higher. These modules incorporate:

- Active zeroization circuitry that erases sensitive data upon detection of physical tampering.
- Environmental failure protection against fluctuations in voltage and temperature.

- Strong enclosures resistant to physical penetration attempts.
- Implementation of constant-time algorithms for all cryptographic operations to prevent timing attacks.
- Electromagnetic shielding to prevent information leakage via electromagnetic emissions.

4.1.1.2 Cryptanalytic attacks on wrapped master seeds

Potential Attacks: Cryptanalytic attacks attempt to break the encryption of the wrapped master seeds through methods such as brute force attacks, known-plaintext attacks, or exploiting weaknesses in the encryption algorithm or its implementation.

Countermeasures: To protect against cryptanalytic attacks, the following measures are implemented:

- Utilization of AES-256 in Galois/Counter Mode (GCM) for key wrapping, providing both confidentiality and authenticity.
- Implementation of a key rotation policy for the Key Encryption Key (KEK), limiting the amount of data encrypted under a single key.
- Inclusion of additional integrity protection through the use of Hash-based Message Authentication Code (HMAC).

4.1.1.3 Network-based attacks

Potential Attacks: Man-in-the-middle attacks involve an attacker intercepting and potentially altering communications between system components. Other network-based attacks might include replay attacks or attempts to exploit vulnerabilities in network protocols.

Countermeasures: To mitigate these attacks, the framework employs:

- Mutual authentication for all system components using X.509 certificates.
- Transport Layer Security (TLS) 1.3 [50] for all inter-component communication.

- Certificate pinning to prevent unauthorized certificate substitution.
- Regular rotation of communication keys and certificates.

4.1.1.4 Insider threats and privileged user attacks

Potential Attacks: These attacks involve malicious actions by individuals with legitimate access to the system, such as system administrators attempting to abuse their privileges or compromised user accounts being used to access sensitive information. Figure 4.1 visualizes the multi-layered security approach used to prevent insider threats and privileged user attacks.

Countermeasures: The proposed framework incorporates several layers of protection:

- Implementation of the principle of least privilege.
- Multi-factor authentication for all privileged operations.
- Separation of duties and four-eyes principle for critical operations.
- Comprehensive logging and real-time alerting for suspicious activities.

4.1.1.5 Denial of service (DoS) attacks

Potential Attacks: DoS attacks attempt to overwhelm system resources by flooding it with a high volume of requests, potentially rendering the system unavailable to legitimate users.

Countermeasures: To maintain system availability, the following measures are implemented:

- Rate limiting and request throttling mechanisms.
- Load balancing across multiple HSMs.
- Implementation of traffic analysis tools for real-time detection and mitigation of DoS attacks.
- Fallback mechanisms for critical operations during high-load scenarios.

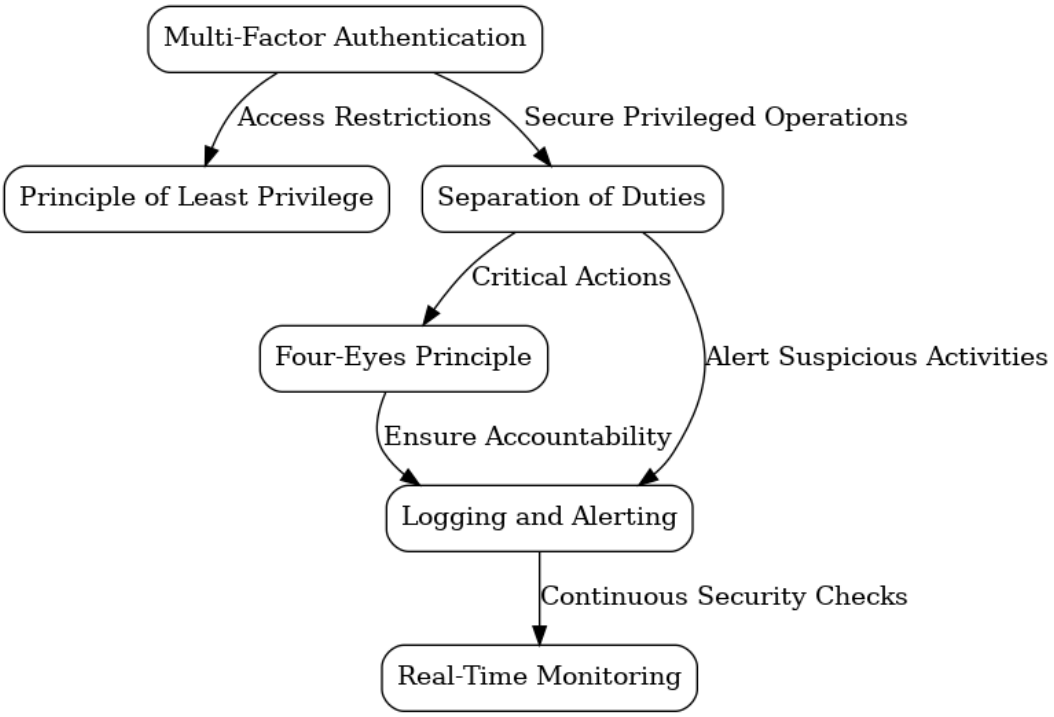

Figure 4.1 : Insider threat prevention approach.

4.1.1.6 Social engineering and malware attacks

Potential Attacks: Social engineering attacks manipulate individuals into divulging sensitive information or performing actions that compromise security. Malware attacks involve introducing malicious software into the system to gain unauthorized access or disrupt operations.

Countermeasures: Protection against these attacks is achieved through:

- Regular security awareness training for all personnel.
- Strict verification procedures for requests involving sensitive operations.
- Isolation of the key derivation process in a secure execution environment.
- Implementation of whitelisting and code signing for all software components.

4.1.1.7 Attacks on external storage

Potential Attacks: These attacks target the database storing wrapped master seeds, attempting to gain unauthorized access, modify stored data, or disrupt storage operations.

Countermeasures: The security of externally stored wrapped master seeds is ensured through:

- Implementation of strong access controls and encryption for the storage system.
- Utilization of database activity monitoring tools.
- Regular backups and integrity validation of stored wrapped seeds.
- Implementation of secure erasure techniques when deleting old wrapped seeds.

4.1.1.8 Quantum computing threats

Potential Attacks: Future large-scale quantum computers could potentially break many current cryptographic algorithms, including those used for key wrapping and digital signatures.

Countermeasures: While quantum computers capable of breaking current cryptographic standards do not yet exist, the system design incorporates measures to facilitate future quantum-resistance:

- Crypto-agility in the system design, allowing for easy algorithm upgrades.
- Monitoring of NIST's post-quantum cryptography standardization process.
- Consideration of hybrid classical/post-quantum schemes as a transitional measure.

4.2 Discussion

The implications of the proposed key management framework for decentralized identity are discussed in this section. The advantages of the framework over existing solutions are examined, potential challenges in implementation and adoption are considered.

The level of risk associated with each attack and the effectiveness of countermeasures are visually assessed in Figure 4.2.

The proposed framework leverages the security properties of Hardware Security Modules, the BIP32 algorithm, and key wrapping techniques to provide a robust

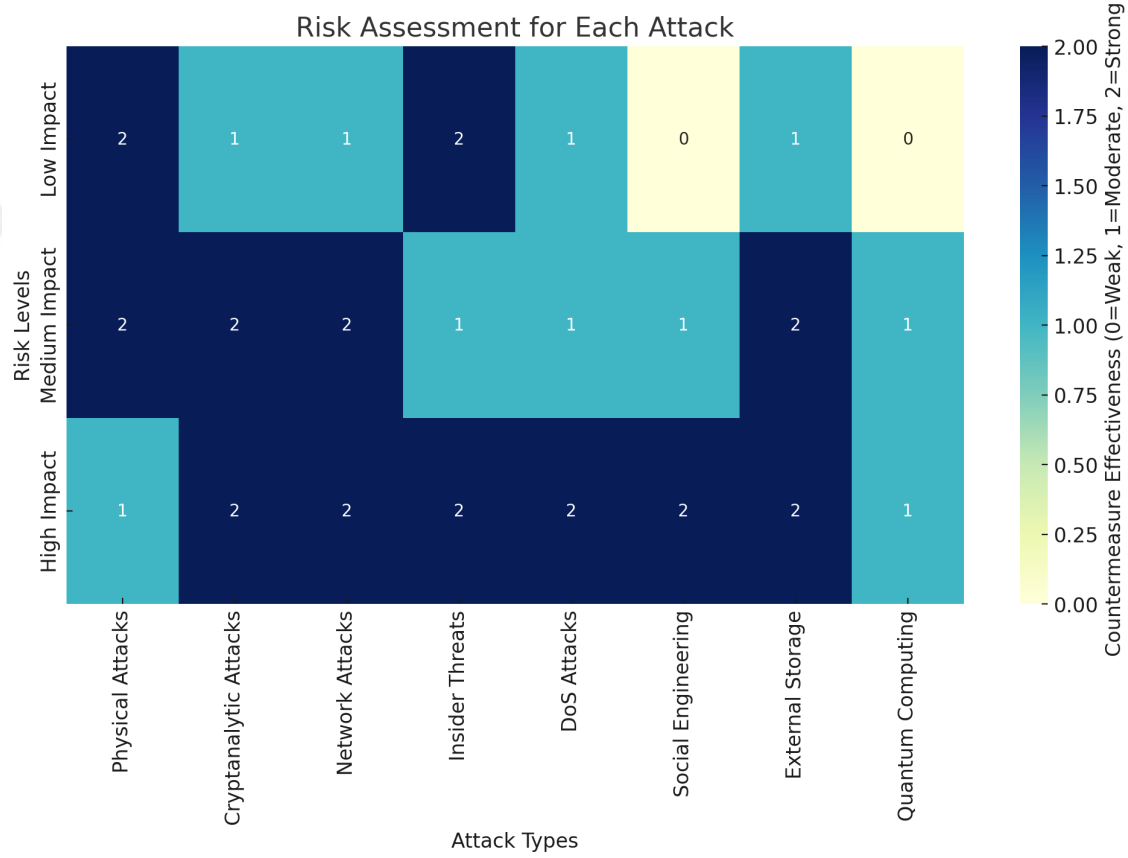

Figure 4.2 : Risk assessment heatmap.

key management solution for large-scale decentralized identity systems. By keeping master seeds and the KEK within the HSM and only storing wrapped seeds externally, the system maintains a high level of security even if the external storage is compromised.

The use of the BIP32 algorithm provides additional security benefits, such as the ability to derive multiple keys without exposing the master seed and the isolation between different key hierarchies. This approach effectively addresses the challenges of complexity, scalability, security isolation, recovery, and secure delegation in decentralized identity systems.

While potential vulnerabilities exist, as with any cryptographic system, the proposed mitigations significantly reduce the risk of successful attacks. The framework's design principle of concentrating the most sensitive operations within the HSM while allowing for scalable external storage of wrapped seeds provides a strong foundation for secure and scalable key management.

Regular security audits, updates to the system components, and ongoing threat modeling will be crucial to maintain and enhance the security posture of the system over time. Future work may include formal security proofs, third-party security audits, and the exploration of post-quantum cryptographic algorithms to ensure long-term security in the face of emerging threats.

HSM-backed architecture eliminate many critical vulnerabilities associated with software-based solutions, including malware-based key extraction, side-channel attacks, and insider threats. Unlike software based solutions, where private keys reside in volatile memory and remain susceptible to unauthorized access, HSMs ensure that keys never leave the hardware boundary that circumstance effectively prevents attacks related to memory scraping, keylogging, and unauthorized remote access. Moreover, the enforcement of MPA and RBAC in HSMs significantly reduces the risk of insider fraud, a crucial concern for decentralized identity systems. Table 4.2 shows a security comparison of software based solutions and HSM-Secured architecture while Table 4.3 and Table 4.7 summarize how the proposed HSM-secured architecture mitigates security risks compared to software based solutions.

Table 4.2 : HSM-Secured versus Software Based Solutions.

Security Factor	Software Based Solutions	HSM-Secured Solutions
Master Seed Exposure	High	None (Stored in HSM)
Side-Channel Attacks	High	Low
Insider Threats	High	Low (MPA Enforced)
Malware	High	Low (Tamper-Resistant)
Tampering Detection	None	High (Self-Destruction)
Regulatory Compliance	Low	High (FIPS 140-2, CC)
Performance Overhead	Low	Medium

4.2.1 Performance results

The integration of HSMs with decentralized identity management systems significantly enhances security and operational efficiency. Our findings indicate that while both BIP32 (ECDSA) and SLIP10 (EdDSA) provide structured key derivation, SLIP10 demonstrates a superior security, computational efficiency, and signing speed, making it a more viable option for high-frequency trading and large-scale decentralized identity management systems. However, BIP32 remains widely adopted due to its compatibility with Bitcoin [16], Ethereum [51], and other ECDSA-based blockchain networks. This makes it a necessary consideration for institutions managing multi-chain digital identities.

Performance evaluations show that SLIP10 exhibited 38% faster key derivation times and 46% lower signing latency compared to BIP32. These efficiency gains stem from EdDSA's streamlined mathematical operations, which eliminate the need for modular inversion calculations, making it computationally lighter than BIP32's ECDSA-based derivation process. Additionally, SLIP10 demonstrated higher resistance to side-channel attacks, reinforcing its suitability for institutional security environments where high assurance cryptographic operations are required. However, despite these advantages, SLIP10 does not complies with with Bitcoin and Ethereum that presents a major adoption challenge, where institutional systems must often interact with multiple blockchain ecosystems. To evaluate the performance of BIP32 (ECDSA, secp256k1) and SLIP10 (EdDSA, Ed25519) within an HSM-secured architecture, we conducted a series of benchmarking tests in a controlled environment.

The results obtained are visually shown and compared in Figure 4.3, Figure 4.4, Figure 4.5, Figure 4.6 and APPENDIX A.

Test Environment:

- **HSM Model:** Dirak NHSM [52]
- **Key Derivation Algorithms:** BIP32 (ECDSA, secp256k1) and SLIP10 (EdDSA, Ed25519)

Benchmark Parameters:

- **Key Derivation Time:** Measures the latency to generate child keys.
- **Transaction Signing Time:** Measures the time taken to sign a transaction.
- **Scalability Impact:** Examines the efficiency of deriving multiple keys simultaneously.

Key derivation is a crucial factor for decentralized identity systems, especially for institutional large scale systems managing millions of identities. Table 4.4 presents benchmark results for deriving one million child keys within the HSM.

In decentralized identity sistems, the transaction signing speed impacts user experience, operation times, and latency. Table 4.5 shows benchmark results for signing one million transactions in an HSM environment and Table 4.6 summarizes performance trade-off BIP32 and SLIP10 algorithms.

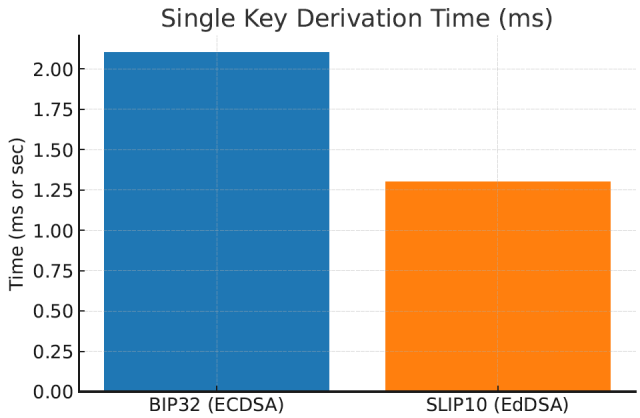

Figure 4.3 : Comparison: Single Key Derivation Time (ms).

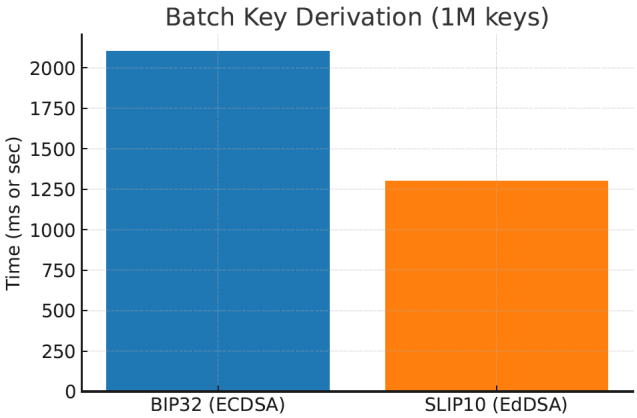

Figure 4.4 : Comparison: Batch Key Derivation (1M keys).

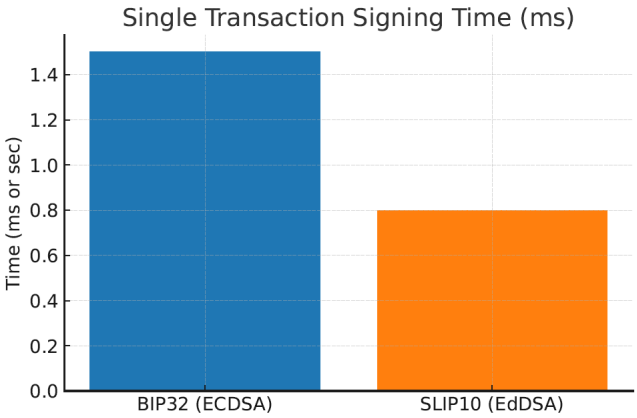

Figure 4.5 : Comparison: Single Transaction Signing Time (ms).

In this research, we highlighted trade-offs between security, performance, and blockchain compatibility in an HSM-secured architecture. While SLIP10 offers superior security and efficiency, institutions requiring cross-chain compatibility and existing infrastructure support may find BIP32 a more practical choice despite its relatively higher computational cost and key leakage risks in non-hardened derivation paths. Furthermore, while HSM integration significantly improves security, it introduces latency in cryptographic operations, making it essential for future research to optimize HSM processing for large-scale institutional deployments.

Analyses results show that SLIP10 is the preferred choice for enhanced security and efficiency, whereas BIP32 remains the industry standard for blockchain compatibility. Therefore, institutions must carefully evaluate their security, performance, and compatibility requirements when designing their HSM-integrated key management architecture to achieve optimal security and efficiency in digital identity management.

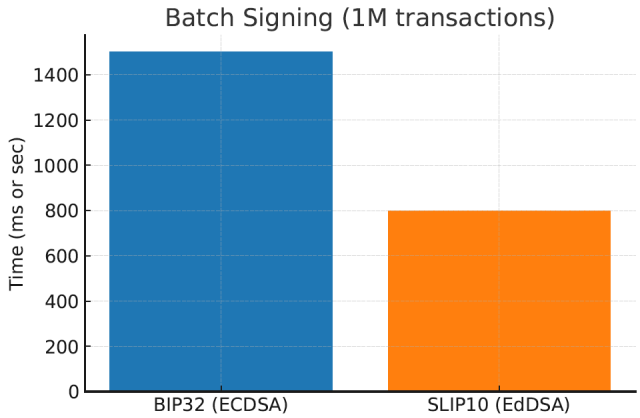

Figure 4.6 : Comparison: Batch Signing (1M transactions).

4.2.2 Scalability considerations

The storage efficiency of the proposed framework is significantly enhanced by the storage of only wrapped master seeds externally. This approach is observed to reduce storage requirements considerably when compared to systems where individual keys for each user and purpose are stored. The use of BIP32 is noted to allow for the derivation of an unlimited number of keys from a single master seed, further enhancing scalability.

Computational efficiency is achieved through a balanced approach. While critical operations are performed by the HSM, it is recognized that the bulk of key derivations can be offloaded to less secure but more scalable computing resources once the master seed is unwrapped. This strategy is seen to allow for horizontal scaling of key derivation operations to meet demand. Key management processes in large-scale deployments are greatly simplified by the need for users and system administrators to manage only a single master seed per identity. This simplification is expected to contribute significantly to the system's scalability.

4.2.3 Advantages over existing systems

Enhanced security is identified as a key advantage of the proposed framework. When compared to systems that store keys in software wallets or databases, the HSM-based approach is observed to provide significantly stronger protection against both physical and cyber attacks. Privacy improvements are noted as another benefit. The hierarchical nature of BIP32 is recognized to allow for the generation of unique keys for different

purposes, thereby enhancing user privacy by reducing the linkability of different actions.

Efficient key recovery is highlighted as a significant advantage. Unlike systems where loss of a key might result in permanent loss of access, the proposed approach is seen to allow for easy recovery of all derived keys through the secure storage of the master seed.

The use of standardized algorithms (BIP32) and hardware (HSMs) is noted to enhance interoperability with other systems and allow for flexible key usage across different applications, contributing to the framework's flexibility and interoperability.

4.2.4 Challenges and limitations

The initial setup complexity is identified as a potential challenge. It is recognized that the initial setup of HSMs and the supporting infrastructure may be complex and costly, potentially limiting adoption by smaller organizations.

Performance bottlenecks are considered as a possible limitation. While the framework is designed for scalability, it is acknowledged that the HSM could become a bottleneck during periods of extremely high demand for key operations.

Regulatory compliance is noted as a potential challenge. Depending on the jurisdiction and use case, it is recognized that regulatory challenges related to key management and user privacy may need to be addressed.

The need for user education is highlighted. It is understood that users may require education on the importance of protecting their master seed and understanding the implications of key hierarchies.

Table 4.3 : Attack Vectors and HSM-Secured Countermeasures.

Attack Type	**Risk in Software Based Solutions and Impact**	**HSM-Secured Countermeasures**
Master Seed Theft	High – Stored in software, vulnerable to malware, phishing attacks, and unauthorized access. If stolen, all derived keys are compromised.	HSM ensures the master seed never leaves the hardware in plaintext, preventing exposure to external threats.
Side-Channel Attacks	High – Timing, power analysis, and differential fault attacks can extract private keys.	HSM ensures all cryptographic operations remain within a secure enclave, eliminating external exposure.
Malware-Based Key Extraction	High – Software Based Solutions store keys in volatile memory, making them susceptible to RAM scraping, keyloggers, and remote exploits.	HSM prevents memory-based key exposure by never storing keys in RAM and enforcing strict access controls.
Non-Hardened Key Leakage (BIP32)	Medium – Non-hardened child keys can be exploited to reconstruct parent keys, leading to full wallet compromise.	SLIP10 (EdDSA) enforces hardened-only key derivation, preventing backward key leakage.
Insider Threats	High – Employees or privileged insiders can extract keys from software Based Solutions, leading to fraud or unauthorized asset transfers.	HSM enforces MPA and RBAC to prevent unauthorized access.
Man-in-the-Middle (MitM) Attacks	High – Private keys and transactions are vulnerable to interception if communication channels are compromised.	HSM signs transactions internally, ensuring that private keys are never exposed externally.
Regulatory Compliance Risks	High – Software Based Solutions often do not meet institutional security standards, leading to regulatory non-compliance.	HSM ensures compliance with FIPS 140-2, CC, PCI-DSS by enforcing strict cryptographic policies.

Table 4.4 : Key Derivation Performance: BIP32 vs. SLIP10 in HSM.

Metric	**BIP32 (ECDSA)**	**SLIP10 (EdDSA)**
Single Key Derivation Time (ms)	2.1 ms	1.3 ms
Batch Key Derivation (1M keys)	2100 sec	1300 sec
Computational Complexity	High	Low
Scalability Efficiency	Moderate	High
Energy Consumption	High	Low

Table 4.5 : Transaction Signing Performance: BIP32 vs. SLIP10 in HSM.

Metric	**BIP32 (ECDSA)**	**SLIP10 (EdDSA)**
Single Transaction Signing Time (ms)	1.5 ms	0.8 ms
Batch Signing (1M transactions)	1500 sec	800 sec
Signature Verification Speed	Moderate	High
Scalability for HFT	Moderate	High

Table 4.6 : Performance Trade-Off: BIP32 vs. SLIP10.

Factor	**BIP32 (ECDSA)**	**SLIP10 (EdDSA)**
Key Derivation Speed	Slow	Fast
Transaction Signing Speed	Slow	Fast
Computational Overhead	High	Low
Blockchain Compatibility	High	Low
Side-Channel Resistance	Moderate	High
Regulatory Compliance	Strong	Strong

Table 4.7 : Attack Vectors and Countermeasures in the Proposed Framework.

Attack Vector	**Countermeasures in Proposed Framework**
Physical/Side-Channel	• FIPS 140-2 Level 4+ HSMs • Constant-time algorithms • Electromagnetic shielding
Cryptanalytic	• AES-256 in GCM mode • KEK rotation policy • HMAC for integrity
Network-based	• Mutual authentication • TLS 1.3 • Certificate pinning
Insider Threats	• Least privilege principle • Multi-factor authentication • Separation of duties
Denial of Service	• Rate limiting • Load balancing • Traffic analysis tools
Social Engineering/Malware	• Security awareness training • Strict verification procedures • Secure execution environment
External Storage Attacks	• Strong access controls • Database activity monitoring • Regular integrity validation
Quantum Computing Threats	• Crypto-agility • Monitoring of post-quantum standards • Consideration of hybrid schemes

5. CONCLUSION AND FUTURE WORK

A novel approach to key management for large-scale decentralized identity systems has been presented in this paper. Critical challenges in security, scalability, and usability have been addressed through the integration of Hardware Security Modules (HSMs), the BIP32 algorithm, and key wrapping techniques. The proposed framework is considered to offer a robust solution for managing cryptographic keys in national-scale identity infrastructures.

Several key contributions have been made through this research. A scalable framework has been introduced that leverages HSMs for secure key generation and storage, while key wrapping is utilized to enable efficient external storage of encrypted master seeds. The adoption of the BIP32 algorithm has been shown to provide a hierarchical key derivation structure, offering improved privacy, simplified key recovery, and flexible key usage across different applications. Through a comprehensive security analysis, the system's resilience against various threat vectors, including external attacks, insider threats, and physical tampering attempts, has been demonstrated. The proposed solution has been shown to address key management challenges such as complexity, scalability, security isolation, recovery, and secure delegation, which are considered critical in large-scale decentralized identity systems.

By integrating HSMs with decentralized identity systems, the proposed architecture eliminates software-based key exposure risks while ensuring tamper-resistant cryptographic operations. A comparative analysis of BIP32 (ECDSA) and SLIP10 (EdDSA) in an HSM environment was conducted, which revealed that SLIP10 provides stronger security against key leakage and side-channel attacks, while BIP32 remains widely adopted due to its blockchain compatibility. Experimental results demonstrate that HSMs effectively prevent private key compromises, enforce MPA, and comply with security regulations (FIPS 140-2, CC, ISO). These findings may contribute to the design of secure and scalable key management for decentralized

identity systems, providing a scalable, high-security model for managing digital identities.

Significant implications for the future of decentralized identity management have been identified. A foundation for building more secure and scalable national-level identity systems has been provided, which could potentially accelerate the adoption of decentralized identity solutions. The improved key management approach is expected to enhance user privacy and control over personal data, aligning with the principles of self-sovereign identity. By simplifying key recovery and management, it is anticipated that barriers to entry for users could be reduced, promoting wider adoption of decentralized identity technologies.

While a comprehensive approach to key management in decentralized identity systems has been presented, several areas have been identified as warranting further research. The development of a prototype implementation of the proposed framework and the conduct of extensive performance testing in simulated large-scale environments are recommended. The integration of this key management approach with existing decentralized identity standards and protocols, such as Decentralized Identifiers (DIDs) and Verifiable Credentials, should be explored. The integration of post-quantum cryptographic algorithms is suggested to ensure long-term security against emerging quantum computing threats. Comprehensive user experience studies are proposed to optimize the balance between security and usability in the proposed framework. An analysis of the regulatory implications of this key management approach in various jurisdictions and the development of strategies for ensuring compliance are advised. Finally, the potential for creating networks of distributed HSMs to further enhance system resilience and scalability is identified as an area for future exploration.

In conclusion, the key management framework proposed in this paper is considered to represent a significant step forward in addressing the challenges of large-scale decentralized identity management. By providing a secure, scalable, and user-friendly approach to key management, a foundation has been laid for the next generation of decentralized identity systems. As the field continues to evolve, it is anticipated that

solutions like the one presented here will play a crucial role in shaping the future of digital identity, privacy, and security in our increasingly interconnected world.

REFERENCES

[1] **Jing, Y.**, **Li, J.**, **Wang, Y. and Li, H.** (2021). The Introduction of Digital Identity Evolution and the Industry of Decentralized Identity, *3rd International Academic Exchange Conference on Science and Technology Innovation (IAECST)*, pp.504–508.

[2] **Soltani, R.**, **Nguyen, U.T.**, **An, A. and Galdi, C.** (2021). A Survey of Self-Sovereign Identity Ecosystem, *Security and Communication Networks*, *2021*, 1–26.

[3] **Volkov, A.** (2020). Addressing the Challenges Facing Decentralized Identity Systems, *iSCHANNEL*, *15*(1), 10–15.

[4] **Wang, F. and Filippi, P.D.** (2020). Self-Sovereign Identity in a Globalized World: Credentials-Based Identity Systems as a Driver for Economic Inclusion, *Frontiers in Blockchain*, *2*, 28.

[5] **Schardong, F. and Custódio, R.** (2022). Self-Sovereign Identity: A Systematic Review, Mapping and Taxonomy, *Sensors*, *22*(15), 5641.

[6] **Wuille, P.** (2012). *BIP32: Hierarchical Deterministic Wallets*, GitHub Bitcoin/bips, `https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki`.

[7] **Rajnoha, D.** (2023). *Detection of Bitcoin keys from hierarchical wallets generated using BIP32 with weak seed*, `https://is.muni.cz/th/pnmt2/`.

[8] **Ampel, B.**, **Otto, K.**, **Samtani, S. and Chen, H.** (2023). Disrupting Ransomware Actors on the Bitcoin Blockchain: A Graph Embedding Approach, *Proceedings of the IEEE International Conference on Intelligence and Security Informatics (ISI)*, pp.1–6.

[9] **Chuang, C.M.**, **Hsu, I.J. and Lee, T.Y.** (2023). *A Two-Party Hierarchical Deterministic Wallets in Practice*, *Cryptology ePrint Archive*.

[10] **Hu, T.**, **Xin, B.**, **Liu, X.**, **Chen, T.**, **Ding, K. and Zhang, X.** (2020). Tracking the Insider Attacker: A Blockchain Traceability System for Insider Threats, *Sensors*, *20*(18), 5297.

[11] **Das, P.**, **Erwig, A.**, **Faust, S.**, **Loss, J. and Riahi, S.** (2021). The Exact Security of BIP32 Wallets, *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pp.1020–1042.

[12] **Gutoski, G. and Stebila, D.** (2015). Hierarchical Deterministic Bitcoin Wallets that Tolerate Key Leakage, *International Conference on Financial Cryptography and Data Security*, pp.497–504.

[13] **Ahmed, M.R.**, **Islam, A.K.M.M.**, **Shatabda, S. and Islam, S.** (2022). Blockchain-Based Identity Management System and Self-Sovereign Identity Ecosystem: A Comprehensive Survey, *IEEE Access*, volume 10, pp.113436–113481.

[14] **S. Y. Lim, J.L. and Won, H.T.** (2018). Blockchain Technology the Identity Management and Authentication Service Disruptor: A Survey, *International Journal on Advanced Science, Engineering and Information Technology*, *8*(4-2), 1735–1745, `http://insightsociety.org/ojaseit/index.php/ijaseit/article/view/6473`.

[15] **Barker, E. and Barker, W.** (2018). Recommendation for Key Management, Part 2: Best Practices for Key Management Organization, **Nist special publication (sp) 800-57 part 2 rev. 1 (draft)**, National Institute of Standards and Technology.

[16] **Nakamoto, S.** (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*, Bitcoin.org, `https://bitcoin.org/bitcoin.pdf`.

[17] **Kim, T.H. and Lee, I.Y.** (2020). Secure Hierarchical Deterministic Key Generation Scheme in Blockchain-based Medical Environment, *2nd International Electronics Communication Conference (IECC)*, pp.108–114.

[18] **Das, P.**, **Erwig, A.**, **Faust, S.**, **Loss, J. and Riahi, S.** (2023). BIP32-Compatible Threshold Wallets, *Cryptology ePrint Archive*, *2023*(312), `https://eprint.iacr.org/2023/312`.

[19] **P. Tedeschi, G.P. and Boggia, G.** (2018). When Blockchain Makes Ephemeral Keys Authentic: A Novel Key Agreement Mechanism in the IoT World, *IEEE Globecom Workshops (GC Wkshps)*, 1–6.

[20] **Farhad, M.**, **Saha, G.**, **Nahid, M.A.**, **Chowdhury, F.R.**, **Paul, P.P.**, **Ullah, M.R. and Ferdous, M.S.** (2024). Secure Backup and Recovery of SSI Wallets using Solid Pod Technology, *2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC)*, pp.1101–1111.

[21] **G. Sivanantham, G.S.A.M. and R, R.T.** (2024). Demonstration of Secure Key Management Solution with Use Case in Permissioned Blockchain, *IEEE International Conference on Public Key Infrastructure and its Applications (PKIA)*, pp.1–6.

[22] **D. Genkin, L. Pachmanov, I.P. and Tromer, E.** (2016). ECDH Key-Extraction via Low-Bandwidth Electromagnetic Attacks on PCs, *Cryptographers' Track RSA Conference*, pp.219–235.

[23] **Dib, M. and Pierre, S.** (2022). Insider Attack Model Against HSM-Based Architecture, *IEEE Access*, *10*, 39185–39197.

[24] **Cabrera Gutierrez, A.J.**, **Castillo, E.**, **Escobar-Molero, A.**, **Álvarez Bermejo, J.**, **Morales, D. and Parrilla, L.** (2022). Integration of Hardware Security Modules and Permissioned Blockchain in Industrial IoT Networks, *IEEE Access*, *PP*.

[25] **S. Alrubei, E.B. and Rigelsford, J.** (2022). Adding Hardware Security into IoT-Blockchain Platforms, *IEEE Latin-American Conference on Communications (LATINCOM)*, pp.1–6.

[26] **W. M. Shbair, E.G. and State, R.** (2021). HSM-based Key Management Solution for Ethereum Blockchain, *IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 1–3.

[27] **Allen, C.** (2016). The Path to Self-Sovereign Identity, *Coindesk*, `https://www.coindesk.com/path-to-self-sovereign-identity`.

[28] **W3C** (2021). *Decentralized Identifiers (DIDs) v1.0*, W3C Recommendation, `https://www.w3.org/TR/did-core/`.

[29] **Tobin, A. and Reed, D.** (2016). The Inevitable Rise of Self-Sovereign Identity, *Sovrin Foundation*, `https://sovrin.org/wp-content/uploads/2018/03/The-Inevitable-Rise-of-Self-Sovereign-Identity.pdf`.

[30] **Preukschat, A. and Reed, D.** (2021). *Self-Sovereign Identity: Decentralized Digital Identity and Verifiable Credentials*, Manning Publications.

[31] **McNamara, P.** (2020). Self-Sovereign Identity in the Blockchain Era, *IEEE Blockchain Technical Briefs*.

[32] **SatoshiLabs** (2017). *SLIP-0010: Universal Private Key Derivation from Master Private Key*, `https://github.com/satoshilabs/slips/blob/master/slip-0010.md`, accessed: 2025-04-10.

[33] **Bernstein, D.J.**, **Duif, N.**, **Lange, T.**, **Schwabe, P. and Yang, B.Y.** (2012). High-speed high-security signatures, *Journal of Cryptographic Engineering*, *2*(2), 77–89.

[34] **National Institute of Standards and Technology (NIST)** (2001). *Security Requirements for Cryptographic Modules (FIPS 140-2)*, `https://csrc.nist.gov/publications/detail/fips/140/2/final`, NIST Federal Information Processing Standard (FIPS) 140-2, May 2001.

[35] **National Institute of Standards and Technology (NIST)** (2019). *Security Requirements for Cryptographic Modules (FIPS 140-3)*, `https://csrc.nist.gov/publications/detail/fips/140/3/final`, NIST Federal Information Processing Standard (FIPS) 140-3, March 2019.

[36] **National Institute of Standards and Technology (NIST)** (2023). *NIST Cryptographic Standards and Guidelines*, https://csrc.nist.gov/publications, NIST Computer Security Resource Center.

[37] **Common Criteria Recognition Arrangement (CCRA)** (2017). *Common Criteria for Information Technology Security Evaluation, Version 3.1*, https://www.commoncriteriaportal.org/cc/, Common Criteria Standard, April 2017.

[38] **OASIS PKCS 11 Technical Committee** (2020). *PKCS #11 Cryptographic Token Interface Base Specification Version 3.0*, https://docs.oasis-open.org/pkcs11/pkcs11-base/v3.0/os/pkcs11-base-v3.0-os.html, OASIS Standard, March 2020.

[39] **Microsoft** (2023). *Cryptography API: Next Generation (CNG)*, https://learn.microsoft.com/en-us/windows/win32/seccng/cng-portal, Microsoft Developer Documentation.

[40] **Corporation, O.** (2023). *Java Cryptography Architecture (JCA)*, https://docs.oracle.com/en/java/javase/17/security/java-cryptography-architecture-jca-reference-guide.html, Java Platform, Standard Edition Security Developer's Guide.

[41] **PCI Security Standards Council** (2022). *Payment Card Industry Data Security Standard (PCI DSS) v4.0*, https://www.pcisecuritystandards.org/document_library, PCI DSS Standard, March 2022.

[42] **RSA Laboratories** (1997). *PKCS #11 v2.01: Cryptographic Token Interface Standard*, rSA Security Inc.

[43] **OASIS** (2020). *PKCS #11 Cryptographic Token Interface Current Version*, https://docs.oasis-open.org/pkcs11/pkcs11-base/v3.0/, oASIS Standard.

[44] **Krawczyk, H.**, **Bellare, M. and Canetti, R.** (2010). *HMAC: Keyed-Hashing for Message Authentication*, https://www.rfc-editor.org/rfc/rfc2104, rFC 2104.

[45] **Delaune, S.**, **Kremer, S. and Steel, G.** (2012). Formal analysis of PKCS#11 and proprietary extensions, *Journal of Computer Security*, *20*(4), 469–506.

[46] **Barker, E.** (2017). Recommendation for Block Cipher Modes of Operation: Key Wrapping, **Technical ReportSpecial Publication 800-38F**, National Institute of Standards and Technology (NIST), https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38F.pdf.

[47] **Housley, R.** (2002). *Advanced Encryption Standard (AES) Key Wrap Algorithm*, https://tools.ietf.org/html/rfc3394.

[48] **Housley, R. and Dworkin, M.** (2009). *Advanced Encryption Standard (AES) Key Wrap with Padding Algorithm*, https://tools.ietf.org/html/rfc5649.

[49] **Rogaway, P.** (2006). Authenticated-Encryption with Associated-Data, *ACM Transactions on Information and System Security*, *13*(4), 1–20.

[50] **Rescorla, E.** (2018). *The Transport Layer Security (TLS) Protocol Version 1.3*, https://datatracker.ietf.org/doc/html/rfc8446, RFC 8446, Internet Engineering Task Force (IETF).

[51] **Buterin, V.** (2013). *Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform*, https://ethereum.org/en/whitepaper/, accessed: 2025-04-10.

[52] **TÜBİTAK BİLGEM** (2023). *DİRAK Network HSM*, https://ma3.bilgem.tubitak.gov.tr/hsm/dirak-network-hsm/, accessed: 2025-04-10.

APPENDICES

APPENDIX A : Results

APPENDIX A : Results

The key derivation and transaction signing performance results and comparisons of the algorithms are shown graphically in this chapter.

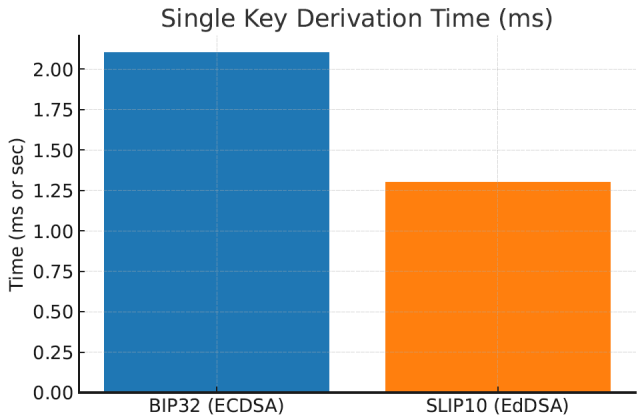

Figure A.1 : Single Key Derivation Time (ms)

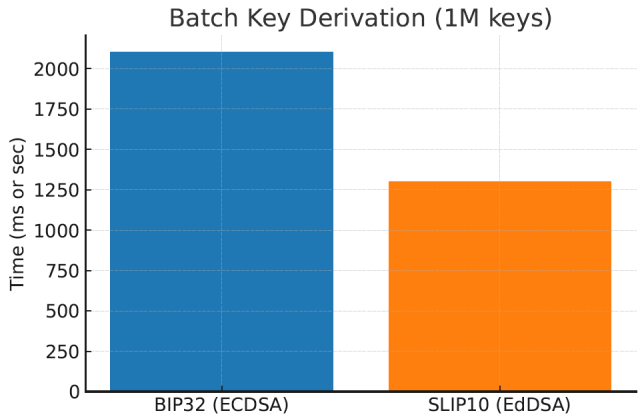

Figure A.2 : Batch Key Derivation (1M keys)

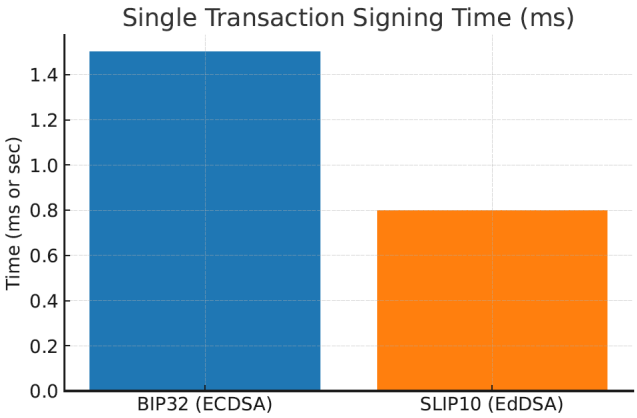

Figure A.3 : Single Transaction Signing Time (ms)

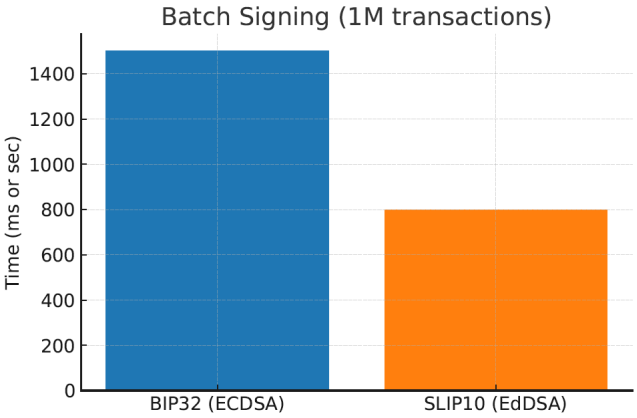

Figure A.4 : Batch Signing (1M transactions)

CURRICULUM VITAE

Name SURNAME: Mert YILDIZ

EDUCATION:

- **B.Sc.:** 2017, İstanbul Technical University, Faculty of Computer and Informatics Engineering, Computer Engineering Department

PROFESSIONAL EXPERIENCE AND REWARDS:

- 2017-2019 Software Engineer, OTOKAR Automotive and Defence Industry.
- 2019-2025 Senior Software Engineer and Team Leader, TÜBİTAK BİLGEM

PUBLICATIONS, PRESENTATIONS AND PATENTS ON THE THESIS:

- **Yildiz M.**, Bahtiyar S. (2024). A Novel Key Management Framework for Secure and Scalable Decentralized Identity Systems. *17th International Conference on Security of Information and Networks (SIN)*, Sydney, Australia, 2024, pp. 1-8, doi: 10.1109/SIN63213.2024.10871880.
- **Yildiz M.**, Bahtiyar S. (2025). *HSM-Secured Hierarchical Deterministic Wallets for Cryptocurrency Custody*. Submitted to *15th International Conference on Advanced Computer Information Technologies (ACIT'2025)*, April 2025.