

REPUBLIC OF TÜRKİYE
AKDENİZ UNIVERSITY



**AN INVESTIGATION AND EVALUATION OF BLOCKCHAIN BASED
AUTHENTICATION METHODS**

Hakan DURMAZ

INSTITUTE OF NATURAL AND APPLIED SCIENCES

DEPARTMENT OF COMPUTER ENGINEERING

MASTER THESIS

JUNE 2025

ANTALYA

REPUBLIC OF TÜRKİYE
AKDENİZ UNIVERSITY



**AN INVESTIGATION AND EVALUATION OF BLOCKCHAIN BASED
AUTHENTICATION METHODS**

Hakan DURMAZ

INSTITUTE OF NATURAL AND APPLIED SCIENCES

DEPARTMENT OF COMPUTER ENGINEERING

MASTER THESIS

JUNE 2025

ANTALYA

REPUBLIC OF TÜRKİYE
AKDENİZ ÜNİVERSİTESİ
INSTITUTE OF NATURAL AND APPLIED SCIENCES

**AN INVESTIGATION AND EVALUATION OF BLOCKCHAIN BASED
AUTHENTICATION METHODS**

Hakan DURMAZ

DEPARTMENT OF COMPUTER ENGINEERING

MASTER THESIS

This thesis was unanimously accepted by the jury on 18/07/2025

Asst. Prof. Dr. Murat AK (Advisor)

Assoc. Prof. Dr. Alper ÖZCAN

Asst. Prof. Dr. Naci ER

ÖZET

BLOKZİNCİR TABANLI KİMLİK DOĞRULAMA YÖNTEMLERİNİN İNCELENMESİ VE DEĞERLENDİRİLMESİ

Hakan DURMAZ

Yüksek Lisans Tezi, Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Dr. Öğr. Üyesi Murat AK

Haziran 2025, 81 sayfa

Son yıllarda blokzincir tabanlı kimlik doğrulama protokolleri ve merkeziyetsiz alternatifler, geleneksel şifre tabanlı sistemlerin güvenlik ve kullanılabilirlik sınırlamaları nedeniyle ön plana çıkmıştır. Literatürde çoğu çalışma yalnızca kendi yönteminin özgün uygulamasını sunmakta, farklı blockchain tabanlı kimlik doğrulama tekniklerini çoklu kriterler üzerinden sistematik şekilde karşılaştıran çalışma sayısı oldukça azdır.

Bu tezde, EVM uyumlu akıllı sözleşmeler ile sıcak cüzdan, donanım cüzdanı, çoklu imza (multisig), NFT tabanlı kimlik doğrulama ve hesap soyutlama (Account Abstraction) yöntemleri geliştirilmiş ve testnet ortamında uygulanmıştır. Her yöntem, doğrulama hızı, güvenlik, kullanıcı deneyimi, ölçeklenebilirlik, karmaşıklık, gizlilik, merkeziyetsizlik, donanım bağımlılığı ve geleceğe uyumluluk gibi nicel ve nitel kriterlere göre çok boyutlu olarak analiz edilmiştir.

Sonuçlar, multisig ve donanım cüzdanlarının yüksek güvenlik sağladığını; sıcak cüzdanlar ile NFT tabanlı kimlik doğrulamanın hız ve erişilebilirlik avantajına sahip olduğunu göstermektedir. Hesap soyutlama ise esneklik ve sosyal kurtarma gibi ileri seviye özellikleriyle öne çıkmaktadır. Her yöntemin avantaj ve dezavantajları uygulama senaryosuna ve risk profiline göre değişmekte, bazı durumlarda hibrit modeller en dengeli çözümü sunabilmektedir.

Bu çalışma, şifresiz, güvenli ve merkeziyetsiz kimlik doğrulama sistemlerine geçiş sürecine katkı sağlamayı, uygulama geliştiriciler ile araştırmacılar için güncel ve karşılaştırmalı bir rehber sunmayı ve gelecek nesil kimlik mimarileri için temel oluşturmayı amaçlamaktadır.

ANAHTAR KELİMELEER: Akıllı Sözleşmeler, Blokzincir, Cüzdan, EVM, Güvenlik, Kimlik Doğrulama.

JÜRİ: Dr. Öğr. Üyesi Murat AK

Doç. Dr. Alper ÖZCAN

Dr. Öğr. Üyesi Naci ER



ABSTRACT

AN INVESTIGATION AND EVALUATION OF BLOCKCHAIN BASED AUTHENTICATION METHODS

Hakan DURMAZ

MSc Thesis in COMPUTER ENGINEERING

Supervisor: Asst. Prof. Dr. Murat AK

June 2025, 81 pages

In recent years, blockchain-based authentication protocols and decentralized alternatives have come to the forefront due to the limitations of traditional password-based systems in both security and usability. While most existing studies focus on implementing their own methods, there remains a clear lack of systematic, multi-criteria comparative evaluations of alternative blockchain-based authentication techniques.

This thesis systematically implements and benchmarks five user authentication methods—hot wallets, hardware wallets, multisignature schemes, NFT-based identity verification, and account abstraction—using EVM-compatible smart contracts on a blockchain testnet. Each method is analyzed across a range of quantitative and qualitative criteria, including authentication speed, security, user experience, scalability, complexity, privacy, decentralization, hardware dependency, and future-proofing.

The results demonstrate that multisignature and hardware wallet authentication offer robust security, while hot wallets and NFT-based approaches provide advantages in speed and accessibility. Account abstraction stands out for its flexibility and advanced features such as social recovery. Each method exhibits distinct strengths and limitations depending on application context and risk profile; in many cases, hybrid approaches may offer the best balance.

This research aims to support the transition to secure, passwordless, and decentralized authentication systems, provide a current comparative reference for developers and researchers, and help lay the foundation for next-generation identity architectures.

KEYWORDS: Authentication, Blockchain, EVM, Security, Smart Contracts, Wallet.

COMMITTEE: Asst. Prof. Dr. Murat AK

Assoc. Prof. Dr. Alper ÖZCAN

Asst. Prof. Dr. Naci ER



ACKNOWLEDGEMENTS

It is my sincere hope that this work will lay the groundwork for more comprehensive studies in the future and inspire projects that explore the applicability of advanced authentication mechanisms. Moreover, I hope that the findings and approaches presented here will contribute not only to the field of authentication, but also to the advancement of truly decentralized and trustless systems, which I believe are essential for the digital future.

I would like to express my deepest gratitude to my esteemed advisor, Asst. Prof. Dr. Murat Ak, whose scientific guidance has been instrumental in shaping this thesis. I am also sincerely thankful to all my professors for their valuable teachings, insights, and encouragement throughout my graduate studies.

And finally, I am profoundly grateful to my family, who—as always—have been my greatest source of motivation and support during this journey. Also this is something I would also like to express in my native language:

Sevgili aileme, bana inandıkları ve koşulsuz sevgileriyle her zaman yanımda oldukları için sonsuz teşekkürler.

CONTENTS

ÖZET	i
ABSTRACT	iii
ACKNOWLEDGEMENTS	v
TEXT OF OATH	ix
SYMBOLS AND ABBREVIATIONS	x
LIST OF FIGURES	xi
LIST OF TABLES	xii
1. INTRODUCTION	1
1.1. Background and Historical Context	2
1.2. Problem Definition and Significance	3
1.3. Aims and Objectives	3
1.4. Overview of Thesis	4
2. LITERATURE REVIEW	5
2.1. Traditional Authentication Methods	5
2.2. Alternative Authentication Methods	7
2.3. Blockchain Technology and Authentication	9
2.4. Blockchain-based Authentication Methods	12
2.5. Advanced Techniques (ZKP and TEE)	16
2.6. Critical Evaluation of Existing Research and Identification of Gaps	18
3. MATERIAL AND METHOD	20
3.1. Blockchain Environment and Framework	20
3.1.1. Ethereum Virtual Machine (EVM) and Sepolia Testnet	20
3.1.2. Smart Contracts and Solidity	21
3.1.3. Development Frameworks and Libraries	21
3.1.4. Rationale for Tool Selection	22
3.2. Authentication Methods Implemented	22
3.2.1. Software (Hot) Wallet Authentication (EOA)	23
3.2.2. Hardware (Cold) Wallet Authentication	27
3.2.3. NFT-Based Authentication	30
3.2.4. Multisig Wallet Authentication	35
3.2.5. Account Abstraction Based Authentication	38
3.3. Testing Environment and Methodology	44

3.3.1.	Test Network and Deployment	44
3.3.2.	Test Automation and Scripted Workflows	44
3.3.3.	Performance Measurement Procedure	45
3.3.4.	Test Conditions and Exclusions	45
3.3.5.	Data Collection and Analysis	46
3.4.	Evaluation Criteria	46
3.4.1.	Criteria and Scoring Approach	46
3.4.2.	Scoring Summary Table	48
3.4.3.	Rationale for Criteria Selection	48
3.4.4.	Scoring Methodology and Criteria Overview	48
4.	RESULTS AND DISCUSSION	50
4.1.	Overview	50
4.2.	Summary of Results	50
4.3.	Discussion	55
4.3.1.	Software (Hot) Wallet Authentication	56
4.3.2.	Hardware (Cold) Wallet Authentication (Ledger)	58
4.3.3.	NFT-Based Authentication	59
4.3.4.	Multisignature (Multisig) Authentication	62
4.3.5.	Account Abstraction-Based Authentication	64
4.3.6.	Best Practice Recommendations	67
4.4.	Limitations and Lessons Learned	68
4.5.	Summary / Key Takeaways	69
5.	CONCLUSION	70
6.	REFERENCES	73
7.	APPENDIX	77
7.1.	Appendix A: Detailed Scoring Rubrics for Evaluation Criteria	77
7.1.1.	Speed	77
7.1.2.	Security	77
7.1.3.	User Experience (UX)	77
7.1.4.	Gas Cost	78
7.1.5.	Complexity	78
7.1.6.	Hardware Dependency	78
7.1.7.	Decentralization Level	79
7.1.8.	Scalability	79

7.1.9. Interoperability	79
7.1.10. Privacy	79
7.1.11. Future Proof	80

CURRICULUM VITAE



TEXT OF OATH

I declare that this study “An Investigation And Evaluation Of Blockchain Based Authentication Methods” which I present as a master thesis, is by the academic rules and ethical conduct. I also declare that I cited and referenced all material and results that are not original to this work.

16/06/2025

Hakan DURMAZ

İmza



SYMBOLS AND ABBREVIATIONS

Abbreviations:

2FA	: Two-Factor Authentication
AA	: Account Abstraction
CSV	: Comma-separated Values File
CTSS	: Compatible Time Sharing System
DApps	: Decentralized Applications
DBIR	: Data Breach Investigations Report
DID	: Decentralized Identity
ECDSA	: Elliptic Curve Digital Signature Algorithm
EdDSA	: Elliptic Curve Digital Signature Algorithm
EF	: Ethereum Foundation
EIP	: Ethereum Improvement Proposal
EOA	: Ethereum Only Account
ERC	: Ethereum Request for Comment
EVM	: Ethereum Virtual Machine
MFA	: Multi-Factor Authentication
MITM	: Man In The Middle
Multisig	: Multi signature
NFT	: Non Fungible Token
SBT	: Soul Bound Token
SDK	: Software Development Kit
SGX	: Software Guard Extensions
SSI	: Self Sovereign Identity
TEE	: Trusted Execution Environment
ZK	: Zero Knowledge
ZKP	: Zero Knowledge Proofs

LIST OF FIGURES

Figure 3.1	Authentication Workflow	24
Figure 3.2	Authentication method implemented on Solidity contract	25
Figure 3.3	Example Authentication Flow Steps	26
Figure 3.4	Hardware Wallet / Ledger Authentication Flow Steps	30
Figure 3.5	ERC-721 Custom NFT Contract	31
Figure 3.6	NFT Authentication Contract	32
Figure 3.7	NFT-Based Authentication Flow	34
Figure 3.8	Minimal Multisig Solidity Contract Snippet	36
Figure 3.9	Multisig Example Authentication Flow	38
Figure 3.10	Account Abstraction Authentication Flow	40
Figure 3.11	Account Abstraction Initialize Using SDK in Typescript	41
Figure 3.12	Account Abstraction Authentication Contract Code Snipped	42
Figure 3.13	Account Abstraction Authentication Example Workflow	43
Figure 4.14	Sign Time Comparison	52
Figure 4.15	Verify Time Comparison	53
Figure 4.16	Total Time Comparison	53
Figure 4.17	Gas Usage Comparison	54

LIST OF TABLES

Table 3.1	Criteria, Scoring Scale, and Key Factors	49
Table 4.2	Comparative Results for Authentication Methods	50
Table 4.3	Criteria-Based Comparative Summary of Authentication Methods	67
Table 7.4	Summary of Scoring Rubrics for Authentication Method Evaluation	81



1. INTRODUCTION

Authentication, the process of verifying user identities, is fundamental to the security infrastructure of digital systems. Traditional authentication methods, predominantly relying on passwords, have long dominated the landscape of identity verification. However, these methods have increasingly revealed critical vulnerabilities and usability issues in the face of evolving digital threats. Password-related weaknesses, such as susceptibility to brute-force attacks, credential theft, password reuse, and social engineering, continue to be significant sources of security breaches, causing severe financial and reputational losses.

The inadequacy of conventional authentication approaches necessitates the exploration and adoption of alternative methods capable of addressing both security risks and user-experience challenges. Recent technological advancements have paved the way for novel solutions, particularly through decentralized architectures such as blockchain technologies. Blockchain, characterized by its distributed, immutable, and transparent ledger system, offers a compelling platform for developing secure, decentralized identity management solutions. Within blockchain ecosystems, Ethereum, with its Ethereum Virtual Machine (EVM), has emerged as the leading smart-contract platform, enabling programmable and automated trustless interactions.

This thesis investigates the potential of blockchain technology and EVM-compatible smart contracts as foundational components for secure, decentralized, and user-centric authentication methods. Specifically, it evaluates various blockchain-based authentication mechanisms, including hot wallets, hardware wallets (such as Ledger), multisignature (multisig) schemes, NFT-based identity verification, and account abstraction techniques. Each approach provides unique features and trade-offs in terms of security guarantees, authentication latency, cost efficiency, and overall usability.

To provide comprehensive empirical insights, this research implements each authentication method using smart contracts deployed on an EVM-compatible test network. The authentication processes were systematically benchmarked, capturing quantitative data on performance metrics such as transaction latency, gas consumption, user interaction complexity, and system robustness. These experimental outcomes offer critical perspectives

on the practical feasibility of blockchain-enabled authentication methods for real-world applications.

By comprehensively analyzing and comparing the performance, practicality, and security characteristics of decentralized, blockchain-based authentication methods against traditional password systems, this thesis aims to clearly identify the strengths, limitations, and real-world suitability of these innovative approaches. The study further evaluates the integration potential of blockchain-enabled authentication methods within emerging decentralized identity (DID) frameworks, emphasizing their capabilities to support advanced identity-management functionalities. Ultimately, this research contributes toward understanding the practical implications of transitioning to secure, user-centric, and passwordless authentication infrastructures for future digital identity systems.

Ultimately, this research contributes significantly to the ongoing discourse on secure authentication paradigms, providing practical insights into the strengths and weaknesses of blockchain-based identity verification methods. By highlighting actionable pathways toward safer, more user-centric authentication infrastructures, this thesis supports the broader goal of evolving digital identity management beyond passwords.

Overall, this work aims to contribute to the evolving field of secure, user-centric, and passwordless identity verification by leveraging the capabilities of blockchain technologies and smart contract platforms.

1.1. Background and Historical Context

Authentication, the process by which a system verifies a user's identity, is a cornerstone of cybersecurity. Historically, authentication has predominantly relied on password-based mechanisms, which date back to the early days of computing in the 1960s. Despite their widespread adoption, password-based systems have persistently suffered from vulnerabilities due to weak password choices, reuse, brute-force attacks, and susceptibility to phishing and social engineering techniques. As digital interactions have become increasingly sophisticated, these vulnerabilities have significantly contributed to data breaches and identity theft, necessitating the pursuit of more secure and user-friendly alternatives.

Parallel to the evolution of authentication methods, blockchain technology emerged in 2008 as the underlying architecture for Bitcoin, introducing decentralized, transparent,

and immutable digital records. The innovative features of blockchain soon extended beyond cryptocurrencies, influencing various fields including finance, supply chain management, and notably, digital identity verification and management. Ethereum's introduction in 2015, with its Ethereum Virtual Machine (EVM), provided a flexible smart-contract platform, enabling programmable logic for decentralized applications (DApps), including sophisticated identity and authentication solutions.

1.2. Problem Definition and Significance

Traditional centralized authentication methods inherently create single points of failure and require trusting third-party intermediaries, leading to vulnerabilities such as database hacks, unauthorized surveillance, and privacy violations. In an increasingly interconnected world, these risks not only endanger individual privacy but also compromise the security of entire digital ecosystems.

Decentralized authentication methods powered by blockchain technology offer significant advantages, including improved security, user sovereignty, transparency, and reduced reliance on central authorities. However, the practicality, scalability, and user-friendliness of blockchain-based identity solutions remain subject to debate and ongoing research. Identifying the optimal balance between security, usability, and performance remains a critical challenge in adopting these technologies.

1.3. Aims and Objectives

This thesis aims to investigate and benchmark alternative authentication methods utilizing blockchain technology, particularly those compatible with the Ethereum Virtual Machine (EVM). The primary objectives include:

- Implementing multiple blockchain-based authentication methods, including hot wallets, hardware wallets (Ledger), multisignature (multisig) schemes, NFT-based identity verification, and account abstraction techniques.
- Systematically benchmarking these methods to assess their authentication latency, security strength, usability, cost efficiency, and practical implementation challenges.

- Analyzing how these decentralized approaches compare to traditional password-based systems, clearly delineating their strengths, weaknesses, and suitability for production-grade deployment.

1.4. Overview of Thesis

This thesis is structured into the following main chapters:

- Chapter 2 (*Literature Review*) provides an extensive review of existing authentication methodologies, blockchain technologies, decentralized identity frameworks, and advanced cryptographic solutions relevant to identity verification.
- Chapter 3 (*Materials and Methods*) details the methodology, experimental setup, smart contract development, and evaluation metrics used for implementing and benchmarking the various authentication mechanisms.
- Chapter 4 (*Results and Discussion*) presents the experimental findings, comparative analyses, and discusses each authentication method's practical implications.
- Chapter 5 (*Conclusions*) summarizes key findings, highlights the theoretical and practical implications, and outlines directions for future research.

Through this structured approach, the thesis aims to contribute substantively to understanding and advancing the field of secure and decentralized digital identity management, ultimately supporting a broader transition toward safer, passwordless digital environments.

2. LITERATURE REVIEW

2.1. Traditional Authentication Methods

Authentication methods have been at the core of securing digital interactions since the inception of computerized systems. Historically, the most prevalent authentication method has been password-based verification, originating in the early days of computing. The concept of passwords as a means of verifying user identity can be traced back to the Compatible Time-Sharing System (CTSS) developed at MIT in the early 1960s, marking one of the earliest known implementations of a password-protected digital environment (Corbató et al. 1962).

Despite their simplicity and widespread adoption, password-based authentication methods have exhibited numerous limitations and security weaknesses. The inherent vulnerabilities associated with passwords have been thoroughly documented in cybersecurity literature and industry reports over several decades. For instance, passwords chosen by users frequently suffer from weak complexity and are prone to guessing attacks, dictionary attacks, and brute-force methods (Bonneau 2012). A seminal study by Florencio and Herley (2007) revealed that users tend to reuse passwords across multiple platforms, significantly increasing the risk of cascading breaches when a single service is compromised. Specifically, they found that the average user had 6.5 distinct passwords, each shared across 3.9 different sites, and maintained around 25 password-protected accounts. The authors note:

"For a user with, e.g. 30 password accounts, the problem becomes not remembering 30 distinct passwords, but rather remembering which of 5 or 6 passwords was used... the average password appears to be eventually used at just under 6 distinct login sites."

This widespread reuse of passwords substantially increases risk that a compromise of one service could lead to breaches of multiple other accounts (Florêncio and Herley 2007).

Furthermore, password-based authentication is highly susceptible to social engineering attacks, such as phishing, in which attackers deceive users into disclosing their credentials. Verizon's annual Data Breach Investigations Report (DBIR) consistently highlights phishing attacks as one of the primary vectors leading to data breaches globally (Verizon 2023). Such attacks underscore the fundamental limitation of relying exclu-

sively on knowledge-based authentication mechanisms, as they fail to provide resilience against human error and deception.

Password-based authentication systems typically rely on centralized databases to store user credentials, creating single points of failure. Numerous high-profile breaches have highlighted the catastrophic consequences of this architectural weakness, such as the Yahoo breach in 2013, where approximately three billion user accounts were compromised due to poor credential storage and centralized management (Perlroth 2017). Centralized credential databases not only attract targeted cyber-attacks but also expose sensitive user data to unauthorized access and potential exploitation.

In an effort to mitigate these vulnerabilities, the cybersecurity community has attempted various improvements, including password policies mandating complexity requirements, forced periodic password changes, and user education campaigns. However, empirical research consistently indicates limited effectiveness for these approaches. Users often react to complexity requirements and frequent password changes by selecting predictable patterns or writing down passwords, thereby diminishing the intended security benefits (Adams and Sasse 1999; Ur et al. 2016).

Another widely adopted improvement in password-based authentication is two-factor (2FA) and multi-factor authentication (MFA), which combines something the user knows (password), something the user possesses (e.g., hardware token or smartphone), and something the user inherently is (biometric data). Although multi-factor authentication significantly enhances security over single-factor password systems, it is still frequently implemented alongside passwords, thus inheriting many of their original vulnerabilities, such as susceptibility to phishing and replay attacks (Bonneau et al. 2012).

From a usability perspective, traditional password-based methods have also been criticized due to cognitive burdens placed on users, especially with the increasing proliferation of online accounts. A recent survey indicates that the average user manages more than 90 online accounts, complicating password management tasks and frequently prompting unsafe user behaviors, including password reuse and insecure storage practices (LastPass 2022).

Given these persistent challenges, there is a pressing need for exploring alternatives to traditional password-based authentication. Recent research efforts have increasingly

focused on decentralized and cryptographically robust authentication mechanisms that not only mitigate the existing vulnerabilities but also enhance user experience and privacy.

2.2. Alternative Authentication Methods

Due to the well-documented security and usability challenges associated with traditional password-based authentication, researchers and industry practitioners have actively explored various alternative methods. These alternatives primarily aim to enhance security, reduce reliance on user memory, and increase overall user experience. Among these, multi-factor authentication (MFA), biometric authentication, and decentralized authentication approaches have gained significant prominence.

Multi-factor Authentication (MFA), an extension of two-factor authentication (2FA), is widely recognized as a critical step toward stronger authentication. MFA systems require users to provide at least two separate pieces of evidence (factors) to authenticate successfully: typically, something the user knows (e.g., a password or PIN), something the user possesses (e.g., a hardware token or smartphone), and something the user inherently is (e.g., biometric data such as fingerprint or facial recognition) (Bonneau et al. 2012).

Although two-factor authentication (2FA) methods substantially improve security against account compromise, they are not without usability limitations. Reese et al. (2019) conducted a two-week study with five common 2FA methods and found that, while participants generally rated the methods as usable and expressed interest in adopting 2FA for sensitive accounts, a significant portion reported inconvenience due to the need for an additional device, timing issues, and increased login time. For example, about one-third of participants experienced situations where their second-factor device was not immediately available, and users of time-based codes often struggled to enter codes before expiration. The authors highlight that, despite 2FA's security advantages, usability and convenience challenges remain barriers to widespread adoption (Reese et al. 2019).

Moreover, some MFA implementations, especially those based on SMS or email verification codes, have demonstrated vulnerabilities to social engineering attacks such as SIM-swapping and man-in-the-middle (MITM) attacks (Siadati et al. 2015). Consequently, MFA alone may not fully address the weaknesses inherent in traditional authentication systems, necessitating the exploration of other robust methods.

Biometric Authentication is another prominent alternative designed to address usability and security issues associated with traditional authentication systems. Biometric authentication verifies users based on unique physical or behavioral characteristics, such as fingerprints, iris patterns, facial recognition, voice recognition, or even behavioral biometrics like typing dynamics and gait analysis (Jain et al. 2016). The widespread availability of biometric sensors in smartphones and laptops has accelerated the adoption of biometric methods, significantly improving the convenience and speed of authentication processes.

Research consistently supports the effectiveness of biometric authentication methods in enhancing security and user convenience, significantly reducing reliance on memorization or possession of physical tokens (Jain et al. 2008). Nonetheless, biometrics also face critical challenges, particularly privacy concerns and risks associated with storing sensitive biometric data. Unlike passwords or tokens, biometric data, if compromised, cannot easily be revoked or changed, representing a significant privacy and security risk (Ratha et al. 2001). Moreover, advanced adversarial attacks, including presentation attacks (spoofing), highlight the need for continuous improvements and combined authentication methods to maintain robust security standards (Marcel et al. 2014).

Decentralized and Blockchain-based Authentication Methods represent a relatively recent innovation in identity management, addressing the weaknesses of centralized credential storage. Decentralized authentication utilizes distributed ledger technology to provide secure, tamper-resistant digital identity verification, removing central points of failure (Dunphy and Petitcolas 2018).

Blockchain-based authentication methods frequently incorporate the principles of Self-Sovereign Identity (SSI) and Decentralized Identifiers (DIDs), empowering users to control and manage their own digital identities without dependence on centralized identity providers (Preukschat & Reed 2021; Tobin and Reed 2016). Such systems leverage blockchain tech to provide strong security guarantees, enable cryptographic verification of identity claims, and support selective disclosure of user attributes thereby significantly enhancing user privacy and autonomy (Dunphy and Petitcolas 2018; Ferdous et al. 2019). Such approaches represent a paradigm shift from traditional identity management systems by removing single points of failure and promoting user-centric privacy and control.

Notably, blockchain-enabled authentication techniques include various innovations

such as hardware wallets (e.g., Ledger), multisignature (multisig) wallets, and NFT-based identity mechanisms. Hardware wallets store cryptographic keys securely offline, dramatically reducing the risk of key theft or compromise (Gutoski and Stebila 2019). Multisignature wallets provide robust authentication security by requiring multiple authorized signatures for transaction validation, thereby reducing risks associated with single-key compromise (Buterin 2014b). NFT-based identity authentication leverages non-fungible tokens to represent unique, verifiable identity credentials, providing enhanced transparency and traceability of user identities (Wang et al. 2022).

Additionally, advanced blockchain-based methods such as account abstraction enable flexible and customizable authentication logic on-chain, facilitating sophisticated security features like social recovery, key rotation, and multi-device support. This flexibility contributes significantly to enhancing user experience and addressing usability challenges commonly associated with decentralized authentication solutions (Buterin 2021a).

In summary, alternative authentication methods such as MFA, biometrics and blockchain based decentralized authentication address many traditional password-based authentication issues. However, each method presents its own advantages and limitations, highlighting the need for comprehensive evaluations to understand their practical implications. The following sections further explore blockchain technology's potential, detailing specific implementations and innovations relevant to authentication and identity management.

2.3. Blockchain Technology and Authentication

Blockchain technology emerged prominently with the introduction of Bitcoin by Satoshi Nakamoto in 2008. Initially conceived as the technological backbone for decentralized digital currencies, blockchain quickly expanded beyond cryptocurrency applications into various fields, notably including identity management and authentication. Blockchain is fundamentally characterized by its decentralized, transparent, and tamper-resistant distributed ledger, maintained through consensus algorithms across a peer-to-peer (P2P) network (Nakamoto 2008). These intrinsic features make blockchain particularly suited for secure and transparent identity management applications.

Ethereum, launched in 2015 by Vitalik Buterin and his collaborators, represented a significant leap forward in blockchain technology. Unlike Bitcoin's limited scripting lan-

guage, Ethereum introduced the Ethereum Virtual Machine (EVM), a Turing-complete execution environment capable of executing complex programmable logic via smart contracts (Buterin 2014). Smart contracts are self-executing contracts with the rules and logic directly coded into software, allowing automated execution and enforcement without intermediaries. This innovation has significantly widened blockchain's applicability, allowing for intricate, decentralized applications (DApps), including robust digital identity solutions and authentication systems.

The concept of decentralized identity (DID) leverages blockchain's decentralized architecture to address the fundamental shortcomings of traditional, centralized identity management systems. In conventional frameworks, users must trust central authorities, creating privacy risks, single points of failure, and vulnerability to large-scale data breaches. In contrast, blockchain-based DID systems empower individuals with full control over their digital identity attributes, allowing self-managed, cryptographically verifiable credentials without dependency on centralized entities. This approach significantly enhances privacy, strengthens user autonomy, and increases resilience against a wide range of attack vectors, including identity theft, correlation attacks, and centralized infrastructure compromise (Dunphy and Petitcolas 2018; Ferdous et al. 2019; Zhao et al. 2023).

Closely associated with decentralized identity is the concept of Self-Sovereign Identity (SSI), which advocates for identity management solutions where individuals maintain complete ownership and control of their identity credentials. SSI provides users with cryptographically verifiable claims stored securely on blockchain or decentralized storage solutions, which can then be selectively disclosed to third parties upon the user's explicit consent (Mühle et al. 2018). SSI platforms use cryptographic mechanisms such as public-private key pairs and digital signatures to ensure security, authenticity, and privacy of identity data, fundamentally altering traditional identity verification models by removing dependency on centralized identity providers.

Several blockchain-based decentralized identity frameworks have emerged, notably including Sovrin, uPort, Civic, and Hyperledger Indy, each implementing distinct approaches to SSI and DID. Sovrin and Hyperledger Indy, for instance, focus on providing permissioned blockchain infrastructures specifically optimized for identity management, incorporating privacy-preserving techniques and selective disclosure mechanisms

(Tobin and Reed 2016b). Conversely, uPort and Civic utilize public blockchain infrastructure (primarily Ethereum), providing open and interoperable identity verification solutions with robust security guarantees (Jacobovitz 2016; Dunphy and Petitcolas 2018).

Blockchain technology inherently enables secure authentication processes through cryptographic proofs and immutable recordkeeping on-chain. In a typical blockchain-based authentication flow, the user signs a challenge with their private key, and the verification is performed using their public key, eliminating the need for passwords and dramatically reducing risks associated with credential stuffing, brute-force attacks, or server-side credential breaches (Bonneau 2015). Public-key cryptography ensures that credentials are not reused or stored on centralized servers, greatly enhancing security. Moreover, blockchain's immutable architecture guarantees that all authentication events and identity-related transactions are auditable and tamper-proof, significantly improving transparency and accountability in decentralized authentication systems (Dunphy and Petitcolas 2018; Zyskind et al. 2015).

Ethereum's flexible EVM platform supports advanced authentication techniques such as multisignature (multisig) wallets, NFT-based identity verification, and account abstraction. Multisignature wallets enhance security by requiring multiple authorized signatures to authenticate transactions or identity-related actions, substantially mitigating risks associated with compromised private keys (Buterin 2014b). NFT-based identity solutions leverage non-fungible tokens to represent unique identity attributes securely and transparently on blockchain, facilitating verifiable and tamper-resistant identity claims (Wang et al. 2022). Account abstraction further advances Ethereum's capabilities by allowing customizable transaction logic, enabling sophisticated authentication scenarios including social recovery, multi-device authentication, and dynamic security policies directly within smart contracts (Buterin 2021a).

In summary, blockchain technology, particularly Ethereum's EVM and associated decentralized identity paradigms (DID and SSI), offers a robust foundation for secure, transparent, and decentralized authentication systems. These solutions effectively address critical vulnerabilities in traditional centralized authentication frameworks, providing enhanced privacy, autonomy, and security for digital identity management. The following section explores specific blockchain-based authentication methods, evaluating their fea-

tures, benefits, limitations, and practical implications.

2.4. Blockchain-based Authentication Methods

Blockchain technology's unique properties—decentralization, immutability, and cryptographic security—have enabled significant innovations in authentication methods. Various blockchain-based authentication solutions have been developed to mitigate vulnerabilities inherent in traditional centralized approaches, particularly those related to password storage and single points of failure. This section examines prominent blockchain-enabled authentication methods: hot wallets, hardware wallets, multisignature (multisig) wallets, NFT-based identity verification, and account abstraction.

Hot wallet authentication refers to identity verification and transaction signing using software-based cryptocurrency wallets that maintain a persistent connection to the internet. These wallets, such as MetaMask, Trust Wallet, Coinbase Wallet, and browser-integrated plugins, store users' private cryptographic keys locally on their device—often in an encrypted keystore or browser storage—but unlike hardware or “cold” wallets, these keys are exposed to online threats due to their constant internet accessibility (Li et al. 2021; Conti et al. 2018).

Hot wallets typically utilize cryptographic algorithms such as the Elliptic Curve Digital Signature Algorithm (ECDSA) or EdDSA to generate digital signatures. When a user authenticates or initiates a blockchain transaction, the wallet software signs a cryptographic challenge or transaction data with the user's private key. The resulting digital signature is then submitted to the blockchain network or a decentralized application (DApp), which verifies it using the corresponding public key. This process confirms both the user's ownership of the private key and the integrity of the signed data without exposing sensitive credentials to third parties (Nakamoto 2008; Wood 2014).

While hot wallets provide significant convenience for interacting with decentralized finance protocols, DApps, and token transfers—offering rapid access and seamless integration with web services—they are inherently more susceptible to risks such as phishing, malware infections, key extraction, and unauthorized device access. As a result, users are encouraged to follow robust security practices, such as enabling two-factor authentication where possible, using strong device passwords, and remaining vigilant against social

engineering attacks (Conti et al. 2018; Li et al. 2021).

Cold wallets, refer to cryptocurrency key management solutions in which private keys are generated and stored entirely offline, with no persistent connection to the internet. This isolation provides maximum resistance against cyberattacks, malware, phishing, and unauthorized remote access, as the private keys never reside on internet-connected devices during critical operations (Eskandari et al. 2015; Li et al. 2021; Conti et al. 2018). Cold wallets can be implemented in various forms, including hardware wallets (such as Ledger and Trezor), paper wallets, and air-gapped computers or devices that perform cryptographic signing in a secure environment.

Technically, cold wallets utilize strong asymmetric cryptography—most commonly the Elliptic Curve Digital Signature Algorithm (ECDSA) for Bitcoin and Ethereum, or EdDSA (Edwards-curve Digital Signature Algorithm) for certain modern blockchains—to generate public/private key pairs and produce digital signatures. When a transaction or authentication operation is required, the unsigned data is transferred to the offline device; the device signs the data with the private key and returns only the signature, never exposing the private key itself (Nakamoto 2008; Wood 2014; Eskandari et al. 2015). This method ensures that even if the online system is compromised, the critical secret (private key) remains inaccessible to attackers.

The usability of cold wallets presents certain trade-offs: although they offer superior security, offline storage can make transaction approval less convenient, requiring physical access to the device or backup medium. Users must also manage backup and recovery procedures for private keys or seeds, as loss or damage to the cold wallet can result in permanent loss of access to funds or identity credentials (Eskandari et al. 2015; Li et al. 2021). Despite these challenges, cold wallets are considered the gold standard for long-term secure storage of digital assets and are widely recommended for safeguarding significant cryptocurrency holdings or critical identity keys.

Multisignature (multisig) wallets are a significant advancement in blockchain-based authentication, providing enhanced security by requiring multiple independent cryptographic signatures for the authorization of transactions or identity-related actions. In a typical M-of-N multisig scheme, M out of N designated private keys must sign a transaction for it to be executed, reducing the risk posed by any single compromised key

(Bonneau et al. 2015; Buterin 2014b). This threshold signature model is widely used in both organizational and personal crypto asset management for shared custody, joint accounts, and enterprise governance.

Technically, most multisig wallets leverage standard digital signature algorithms primarily the Elliptic Curve Digital Signature Algorithm (ECDSA) on both Bitcoin and Ethereum networks, though modern implementations are increasingly supporting EdDSA (Ed25519) for higher security and efficiency (Li et al. 2021; Conti et al. 2018). When a user initiates a transaction, each participating wallet generates a digital signature using its private key. The transaction is executed on-chain only when the requisite number of valid signatures is submitted. On Ethereum, multisig functionality is implemented as smart contracts (e.g., Gnosis Safe), which orchestrate signature verification and access control logic directly on the EVM, typically relying on cryptography primitives provided by libraries such as OpenZeppelin Contracts, ethers.js, web3.js, or Bitcoin's libbitcoin, bitcoinjs-lib, and cryptography libraries like secp256k1 for key management and signature verification (Gnosis Safe 2021; Wood 2014).

These smart contracts or on-chain scripts verify that submitted signatures match the registered public keys and fulfill the defined threshold policy before authorizing an action. As such, multisig wallets dramatically reduce risks associated with private key loss or theft, but they introduce additional complexity in wallet setup, coordination among participants, and backup/recovery procedures. Usability research highlights that non-technical users may find multisig workflows more complex, requiring user education and robust management tools to minimize operational risks (Bonneau et al. 2015; Eskandari et al. 2015).

NFT-based identity verification leverages non-fungible tokens (NFTs)—unique cryptographic assets on blockchain—to represent, manage, and authenticate digital identity attributes or credentials. NFTs, typically following ERC-721 or ERC-1155 standards, provide uniqueness, verifiability, and transparency, making them well-suited for encoding tamper-proof credentials such as educational diplomas, professional licenses, and digital certificates (Wang et al. 2022; Zhao and Si 2022). In these systems, a user's identity claims are either directly recorded as metadata in the NFT or cryptographically linked to the NFT, which is controlled by the user's blockchain wallet. Verification is performed by checking on-chain proofs, ownership records, and digital signatures, enabling

decentralized and trustless authentication without dependence on centralized authorities (Wang et al. 2022; Li et al. 2021).

NFT-based identity solutions are already being explored in fields such as online education, healthcare, and digital rights management, offering traceability, transparency, and strong resistance to forgery (Zhao and Si 2022; Li et al. 2021). Nevertheless, the broader deployment of NFT-based identity systems faces challenges including privacy preservation (since on-chain credentials can be publicly visible), transaction scalability, and legal/regulatory compliance with respect to personal data storage (Wang et al. 2022; Li et al. 2021).

Account Abstraction (AA) is an innovative mechanism in the Ethereum blockchain, allowing for the flexible implementation of authentication and transaction validation logic at the smart contract level, rather than being restricted to the default externally owned account (EOA) model (Buterin 2021a; Antonopoulos and Wood 2018). Proposed by Vitalik Buterin and further developed through Ethereum Improvement Proposals (notably EIP-4337 and EIP-2938), account abstraction enables developers to define custom signature schemes, social recovery procedures, multi-factor authentication, multi-device authorization, and dynamic transaction policies within smart contracts. This approach decouples user identity and authentication from standard key management, facilitating use cases such as gasless transactions, programmable spending limits, and fine-grained access control (Buterin 2021a; Antonopoulos and Wood 2018; EF 2023).

By abstracting away key management complexity, account abstraction is expected to substantially improve both the security and usability of blockchain-based authentication and identity management solutions. Research and real-world pilots demonstrate that AA can support advanced scenarios like social recovery (where trusted parties can help restore access), time-locked transactions, and post-quantum signature integration enabling greater flexibility and resilience for users (Buterin 2021a; Antonopoulos and Wood 2018; EF 2023; Wood 2014). As Ethereum and similar platforms integrate account abstraction, broader adoption of decentralized authentication and identity management is anticipated.

In practical terms, blockchain-based authentication systems typically combine several methods described above, balancing trade-offs among security, usability, and performance. For instance, using a hardware wallet to manage primary cryptographic keys,

multisig wallets for high-value transactions, NFTs for verifiable credentials, and account abstraction for flexible identity management provides a comprehensive, secure, and user-friendly identity framework. Such combined approaches exemplify modern decentralized identity systems, meeting contemporary security requirements and user expectations.

Despite these advancements, blockchain-based authentication methods are not without challenges. Scalability issues, transaction latency, user onboarding complexity, and the necessity for robust key management practices remain barriers to widespread adoption. Additionally, regulatory uncertainty regarding blockchain-based identity solutions, data protection compliance (e.g., GDPR), and interoperability standards among various decentralized identity platforms represent critical research areas that require further exploration.

In conclusion, blockchain-based authentication methods including hot wallets, hardware wallets, multisig authentication, NFT-based verification, and account abstraction offer substantial improvements over traditional password-based methods, particularly in terms of security, decentralization, and user control. The following section expands on advanced cryptographic techniques, specifically Zero-Knowledge Proofs (ZKP) and Trusted Execution Environments (TEE), further highlighting emerging innovations in decentralized authentication frameworks.

2.5. Advanced Techniques (ZKP and TEE)

Recent advances in cryptography and secure hardware have introduced powerful new paradigms for authentication and digital identity, offering privacy, scalability, and security far beyond traditional blockchain methods. The most impactful among these are Zero-Knowledge Proofs (ZKP) and Trusted Execution Environments (TEE).

Zero-Knowledge Proofs (ZKP), first introduced by Goldwasser, Micali, and Rackoff, allow a prover to demonstrate possession of certain knowledge or attributes to a verifier without revealing the underlying information itself (Goldwasser et al. 1989). ZKPs are a powerful cryptographic primitive that enable computational integrity and privacy preservation simultaneously, making them especially valuable for privacy-preserving authentication, digital identity verification, and secure transactions. Unlike other privacy-enhancing cryptographic methods such as homomorphic encryption or secure multiparty

computation, ZKPs are notable for their universality and minimal security assumptions, which enable broad applicability to authentication, smart contracts, and secure computation tasks.

A key advantage of ZKPs lies in their ability to balance transparency and privacy—an essential requirement in blockchain environments, where all transactions are by default public and traceable. By enabling confidential proofs of claims, such as asset ownership, credential possession, or transaction correctness, without exposing underlying data, ZKPs effectively mitigate privacy breaches and targeted attacks arising from data correlation. Modern constructions, such as Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge (zk-SNARKs) and Zero-Knowledge Scalable Transparent Arguments of Knowledge (zk-STARKs), further enhance scalability and verification efficiency, making large-scale deployments in blockchain and non-blockchain settings feasible.

In practice, ZKP-based authentication systems allow users to prove statements about credentials or attributes (for example, age or residency) without disclosing the actual value, which is fundamental for privacy-preserving digital identity and regulatory compliance (such as GDPR). Recent surveys and real-world implementations demonstrate that ZKPs are pivotal in constructing digital systems where security, privacy, and scalability must coexist, including privacy-centric blockchains (e.g., ZCash, Aleo, Mina), confidential smart contracts, and privacy-preserving authentication protocols. Nevertheless, ongoing research continues to address challenges related to computational efficiency, proof generation latency, hardware acceleration, and usability (Lavin et al. 2024).

Trusted Execution Environments (TEEs) represent a hardware-based approach to securing sensitive computations and cryptographic secrets, even when the underlying operating system or hypervisor is compromised. Intel Software Guard Extensions (SGX), as thoroughly described by Costan and Devadas, is a leading example of TEE technology, providing tamper-resistant hardware enclaves for securely executing code and storing cryptographic keys and sensitive data. Within the SGX architecture, enclaves are isolated memory regions set aside by the processor, which are protected from all non-enclave memory accesses—including those from the OS, hypervisor, and peripherals (Costan and Devadas 2016).

TEEs like Intel SGX play a crucial role in modern authentication workflows by safe-

guarding private keys and confidential user data against software-level attacks such as malware, kernel exploits, and unauthorized memory access. The use of cryptographic attestation further enables users to verify that they are communicating with genuine enclave code, providing integrity and confidentiality guarantees even in untrusted cloud or edge environments. However, the security analysis of SGX also reveals that hardware vulnerabilities, such as side-channel attacks and design limitations, remain ongoing research concerns. As with all TEE-based solutions, continuous updates and careful implementation practices are necessary to maintain robust security in the face of evolving threats (Costan and Devadas 2016).

In summary, both ZKP and TEE-based solutions are at the frontier of privacy-preserving, user-centric digital identity and decentralized authentication. Their ongoing development is expected to drive major advances in secure, scalable, and compliant identity management systems.

2.6. Critical Evaluation of Existing Research and Identification of Gaps

A comprehensive review of traditional, alternative, blockchain-based, and advanced authentication methods demonstrates that the field of digital identity and authentication is both dynamic and rapidly evolving. While a significant body of literature has contributed to advances in secure authentication systems, persistent research gaps continue to pose challenges and opportunities for future work.

Traditional authentication systems, particularly those based on passwords, have been extensively analyzed, consistently revealing ongoing weaknesses such as password reuse, weak password choices, social engineering susceptibility, and risks arising from centralized credential management (Bonneau 2012; Florêncio and Herley 2007). Although the adoption of multi-factor authentication (MFA) and biometrics has provided some mitigation, these approaches often introduce usability and accessibility trade-offs, limiting their universal adoption and effectiveness (Marcel et al. 2014). More empirical studies are needed to develop solutions that effectively balance usability and robust security in diverse user populations.

Blockchain-based authentication solutions—utilizing technologies such as the Ethereum Virtual Machine (EVM), decentralized identity models, and self-sovereign identity (SSI)

paradigms—offer substantial security, privacy, and decentralization benefits. Innovations such as hot wallets, hardware wallets, multisignature (multisig) schemes, NFT-based identity, and account abstraction represent important advances over centralized architectures (Buterin 2014; Buterin 2021a). However, the literature reveals several enduring challenges, including complex usability, key management difficulties, limitations in scalability, and interoperability obstacles between blockchains and existing identity systems. Most existing studies tend to examine isolated solutions or theoretical models, lacking comprehensive, comparative evaluations that could inform the selection of optimal authentication methods for practical, real-world deployments.

Moreover, empirical research that rigorously benchmarks blockchain-based authentication methods—assessing usability, scalability, economic viability, and security under real-world conditions—remains relatively scarce. The majority of available studies are limited to proof-of-concept prototypes or narrowly scoped technical analyses, without providing sufficient comparative performance data across various blockchain approaches. This limits the ability of organizations and practitioners to make informed decisions about the adoption and integration of blockchain-based authentication systems in large-scale or high-assurance settings.

Recognizing these gaps, this thesis advances the state of the art by systematically implementing, benchmarking, and comparing a range of blockchain-based authentication methods under realistic conditions. The empirical evaluations presented herein address critical performance metrics—including authentication latency, transaction costs, security properties, and user experience—and provide novel insights into the trade-offs and practical feasibility of leading authentication schemes.

In summary, a critical analysis of current research highlights the necessity for more comprehensive, comparative, and empirical studies that address open questions in decentralized authentication and digital identity. By contributing new empirical results, theoretical perspectives, and practical recommendations, this thesis aims to bridge key knowledge gaps and serve as a foundation for future research and innovation in secure, user-centric authentication systems.

3. MATERIAL AND METHOD

This chapter outlines the materials, methodologies, and experimental setups employed in evaluating various blockchain-based authentication methods. It provides comprehensive details about the technical frameworks, blockchain environments, and evaluation criteria used for implementing and systematically benchmarking the authentication solutions described in the previous chapters.

3.1. Blockchain Environment and Framework

This research utilizes a modern blockchain development environment to rigorously test and evaluate various authentication mechanisms. The goal is to ensure results are both technically sound and broadly relevant for real-world decentralized applications. This section details the fundamental components and rationale behind the chosen tools, networks, and programming paradigms.

3.1.1. Ethereum Virtual Machine (EVM) and Sepolia Testnet

At the core of all experiments is the **Ethereum Virtual Machine (EVM)**, the computation engine that enables the deployment and execution of smart contracts across the Ethereum ecosystem.. Ethereum, due to its maturity, widespread adoption, and programmable Ethereum Virtual Machine (EVM), was selected as the foundational blockchain platform. The EVM provides a deterministic, sandboxed environment where code can be executed exactly as intended—regardless of which network node processes the transaction.

To facilitate experimentation without incurring real-world costs, this study employs the **Sepolia testnet**. Sepolia is a public Ethereum test network that closely mirrors mainnet conditions, including gas fee calculations, transaction throughput, and consensus mechanisms, while operating with valueless test ETH. This allows for realistic performance measurement, rapid prototyping, and safe testing of smart contract upgrades. All deployments and tests are conducted on Sepolia, providing a secure environment that eliminates financial risk while enabling precise measurement of authentication performance and gas consumption.

3.1.2. Smart Contracts and Solidity

Smart contracts are the foundational building blocks of decentralized applications on EVM-compatible blockchains. These self-executing programs are stored on the blockchain and execute when predefined conditions are met, enabling autonomous, tamper-resistant operations such as authentication, asset transfers, or access control.

For this research, all smart contracts are written in **Solidity**, the most widely-used, statically typed programming language for EVM-based platforms. Solidity enables the precise definition of data structures (e.g., user accounts, authentication tokens), access control policies, and verification algorithms, all while ensuring that contract execution remains deterministic and secure. The language's strong community support, ongoing improvements, and comprehensive tooling (including security analysis and optimization tools) make it an industry standard for smart contract development.

3.1.3. Development Frameworks and Libraries

To maximize development efficiency, maintainability, and testing rigor, several specialized frameworks and libraries are employed throughout the project:

- **Hardhat:** Hardhat is a powerful Ethereum development environment that streamlines the process of compiling, testing, deploying, and debugging Solidity smart contracts. Its integrated local blockchain node enables fast iteration, while its scripting capabilities (using JavaScript or TypeScript) allow for automated deployment, transaction simulation, and reproducible test scenarios. Hardhat's plugin ecosystem further facilitates advanced features such as contract verification and gas usage profiling.
- **ethers.js:** ethers.js is a robust and lightweight JavaScript library designed for interacting with the Ethereum blockchain. It provides secure key management, wallet operations, message and transaction signing, and rich API access to blockchain data. In this research, ethers.js is used for both test automation (via scripts) and to simulate client-side interactions such as wallet-based authentication and signature verification, ensuring results are directly applicable to real-world application development.

- **Etherspot SDK (for Account Abstraction):** For the implementation of account abstraction-based authentication, the Etherspot SDK is utilized. This library abstracts away many of the complexities of EVM account management, enabling the testing of programmable wallets and alternative authentication flows that go beyond traditional externally owned accounts (EOAs).
- **Node.js:** The broader JavaScript runtime environment provided by Node.js enables the automation of all deployment, testing, and measurement workflows, ensuring that each authentication scenario can be consistently reproduced and benchmarked.
- **TypeScript:** TypeScript was used throughout the development of automated test scripts, wallet interactions, and in the implementation of the Etherspot SDK for Account Abstraction scenarios. Its static typing and enhanced tooling improved reliability and maintainability in complex authentication workflows and enabled safer integration with Ethers.js and Etherspot libraries.

3.1.4. Rationale for Tool Selection

The chosen stack ensures:

- **Compatibility** with industry standards and emerging blockchain protocols.
- **Reproducibility** of all tests and results, a cornerstone for scientific research.
- **Flexibility** to quickly implement and benchmark multiple authentication paradigms.
- **Security and Transparency** thanks to open-source tooling, auditable code, and standardized best practices.

In summary, the combination of EVM, Sepolia, Solidity, and leading development frameworks creates a robust and future-ready foundation for analyzing and benchmarking decentralized authentication methods.

3.2. Authentication Methods Implemented

This section presents a comprehensive, method-by-method analysis of the blockchain-based authentication techniques implemented and evaluated in this study. Each authentication method was designed, coded, and tested within the same technical environment

described previously, ensuring that results are directly comparable. The methods represent both widely-used and cutting-edge approaches, covering a spectrum from traditional externally owned account (EOA) authentication to advanced account abstraction.

For each method, the following aspects are detailed:

- Conceptual overview and practical use case
- System and contract architecture
- Workflow and transaction flow diagrams (where relevant)
- Implementation details and libraries used
- Security model and threat considerations
- Example authentication scenario
- Strengths and limitations

3.2.1. Software (Hot) Wallet Authentication (EOA)

3.2.1.1 Overview and Use Case

Software wallets, often referred to as “hot wallets,” are the most common form of user-managed cryptocurrency wallets. These wallets store the user’s private key in software on a connected device (computer, smartphone, or browser extension) and are capable of signing transactions and messages at any time, provided the device is online. In the context of blockchain authentication, hot wallets are used to prove account ownership and authorize actions through cryptographic signatures.

Hot wallet authentication is widely used in decentralized applications (dApps), decentralized finance (DeFi) platforms, NFT marketplaces, and any service requiring cryptographically secure user authentication without third-party intermediaries. Typical flows include wallet-based login, on-chain voting, and transaction approvals.

3.2.1.2 Architecture and Workflow

At the core of hot wallet authentication lies the Externally Owned Account (EOA) model in Ethereum and EVM-compatible blockchains. An EOA is simply a key pair composed of a public address (derived from the public key) and a private key (used for

signing). The wallet application handles key management and interacts with smart contracts as needed.

Authentication workflow:

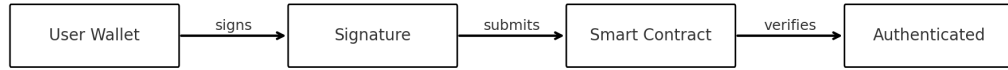


Figure 3.1. Authentication Workflow

1. **Key Generation:** User generates a key pair locally, typically during wallet setup.
2. **Message/Transaction Signing:** To authenticate, the wallet signs a challenge message (or transaction) with the private key.
3. **Signature Verification:** The signature is submitted to a smart contract or backend service, which uses the public address to verify the signature's validity and the user's ownership.
4. **On-Chain Interaction:** If authentication occurs on-chain, the smart contract recovers the signer address using Solidity's `ecrecover` method and compares it to the expected address.

3.2.1.3 Smart Contract Design

In this research, authentication contracts are implemented in Solidity, allowing for signature verification directly on the Ethereum Sepolia testnet. The contracts expose a function (e.g., `verifySignature(signer, message, signature)`) that recovers the address from the signature and compares it to the provided address.

Key contract elements:

- Signature recovery via `ecrecover`.
- Nonce usage or replay protection for security. (Optional)
- Event logging for successful and failed authentication attempts.

```

function authenticateMessage(string memory message, bytes memory
→ signature) public {
    // Convert the string message to bytes first
    bytes memory messageBytes = bytes(message);

    // Use MessageHashUtils.toEthSignedMessageHash directly on the
    → message bytes
    // This matches ethers.hashMessage() in JavaScript
    bytes32 ethSignedHash =
    → MessageHashUtils.toEthSignedMessageHash(messageBytes);

    // Recover the signer
    address signer = ECDSA.recover(ethSignedHash, signature);
    lastRecoveredSigner = signer; // For debugging

    // Emit debug information
    emit DebugInfo(keccak256(messageBytes), ethSignedHash, signer);

    // Store authentication
    authenticatedUsers[signer] = true;

    emit UserAuthenticated(signer);
}

```

Figure 3.2. Authentication method implemented on Solidity contract

3.2.1.4 Implementation Details

- **Development Tools:**

- Hardhat for contract deployment and test automation
- ethers.js for wallet operations and off-chain signing

- **Key Management:** Private keys are managed locally within the test script to eliminate third-party wallet variability, providing controlled and reproducible results.

- **Signature Generation:** The signing process utilizes `signMessage()` or equivalent, producing an ECDSA signature.

- **Signature Verification:**

- On-chain: The smart contract verifies the signature using the `ecrecover` function.
- Off-chain: Verification scripts confirm the result for cross validation.

- **Testing Conditions:**

- All tests conducted on the Sepolia testnet.
- Each authentication scenario is repeated 10 times to obtain average latency and gas cost data.

3.2.1.5 Example Authentication Flow Step-by-step flow:

1. The dApp/server sends a unique challenge (e.g., nonce or message) to the user's wallet.
2. The hot wallet signs the message using its locally stored private key.
3. The user (or script) submits the signature, along with the wallet address and the original message, to the authentication smart contract.
4. The contract verifies that the signature matches the address and approves authentication if successful.
5. Results (success/failure, latency, gas usage) are logged for analysis.

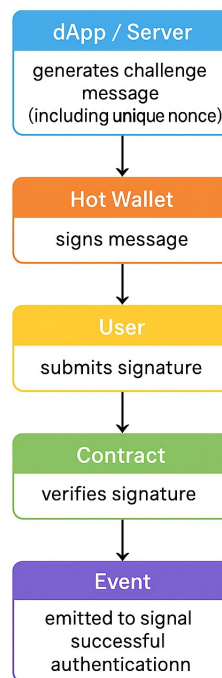


Figure 3.3. Example Authentication Flow Steps

3.2.2. Hardware (Cold) Wallet Authentication

3.2.2.1 Overview and Use Case

Hardware wallets (cold wallets) are dedicated physical devices designed to securely generate, store, and use cryptographic keys offline. By keeping private keys isolated from internet-connected devices, hardware wallets dramatically reduce exposure to malware, phishing, and other remote attacks. This authentication method is especially favored for high-value or long-term storage and for scenarios where enhanced security is required.

Use Cases:

- Secure login and authentication for high-stakes dApps or financial platforms
- On-chain voting or approvals involving large amounts
- Any scenario where the threat of key theft from compromised devices is unacceptable

3.2.2.2 Device Setup and Security Model

A hardware wallet such as Ledger, Trezor, or similar generates and stores private keys inside a tamper-resistant chip. These keys **never leave the device**. Authentication actions (e.g., message signing) are only performed after explicit user confirmation via a physical button or screen prompt.

Security model highlights:

- Private keys are inaccessible to the host device and all external software
- Signing operations require user consent, providing strong protection against unauthorized use
- Devices are typically PIN-protected and offer recovery phrases for disaster recovery

3.2.2.3 Signing and Verification Workflow

Workflow:

1. The dApp or service sends a challenge message (e.g., nonce, login string) to the user's device.

2. The hardware wallet displays the message to the user for review and physical approval.
3. Upon confirmation, the device signs the message internally and returns the cryptographic signature.
4. The user or dApp submits the message and signature to the blockchain smart contract for verification.
5. The contract uses ECDSA-based recovery (as with hot wallets) to confirm the signer's address.

Key point: While the on-chain verification process is identical to hot wallet authentication, the signing operation is isolated within secure hardware, eliminating the risk of key exfiltration from the host device.

3.2.2.4 Smart Contract Interaction

The smart contract interface and verification logic remain the same as in hot wallet authentication:

- Receives the original message, signature, and user address (if needed)
- Computes the Ethereum-signed message hash
- Recovers the signer address from the signature using `ECDSA.recover` or under the hood `ecrecover`
- Verifies the recovered address matches the expected one

This means your authentication contract is **hardware-agnostic**: it can verify signatures from both hot and cold wallets, provided the correct message format and hashing procedure is used.

3.2.2.5 Implementation Details

- **Device Used:** Ledger Nano S Plus
- **Integration Library:**

- `ethers.js` with `ledger support` `@ledgerhq/hw-transport-node-hid` and `@ethersproject/hardware-wallets` libraries.
- Transaction and message signing initiated via USB Type C.
- Bluetooth, or other secure connection can be used if device is supported.
- **Signing Process:**
 - The challenge message is sent to the device
 - The device displays the message and waits for explicit user confirmation (button press)
 - The device produces and returns the signature without exposing the private key
- **Testing Conditions:**
 - All tests conducted on the Sepolia testnet.
 - Each authentication scenario is repeated 10 times to obtain average latency and gas cost data.

3.2.2.6 Authentication Flow

Step-by-step flow:

1. dApp/server generates a unique challenge message for authentication.
2. User connects the hardware wallet and confirms the challenge on the device's screen.
3. Hardware wallet signs the message, and the resulting signature is provided to the user or dApp.
4. The signature and message are submitted to the authentication smart contract on Sepolia.
5. Smart contract recovers and verifies the signer's address.
6. Authentication event is logged and results are recorded for evaluation.

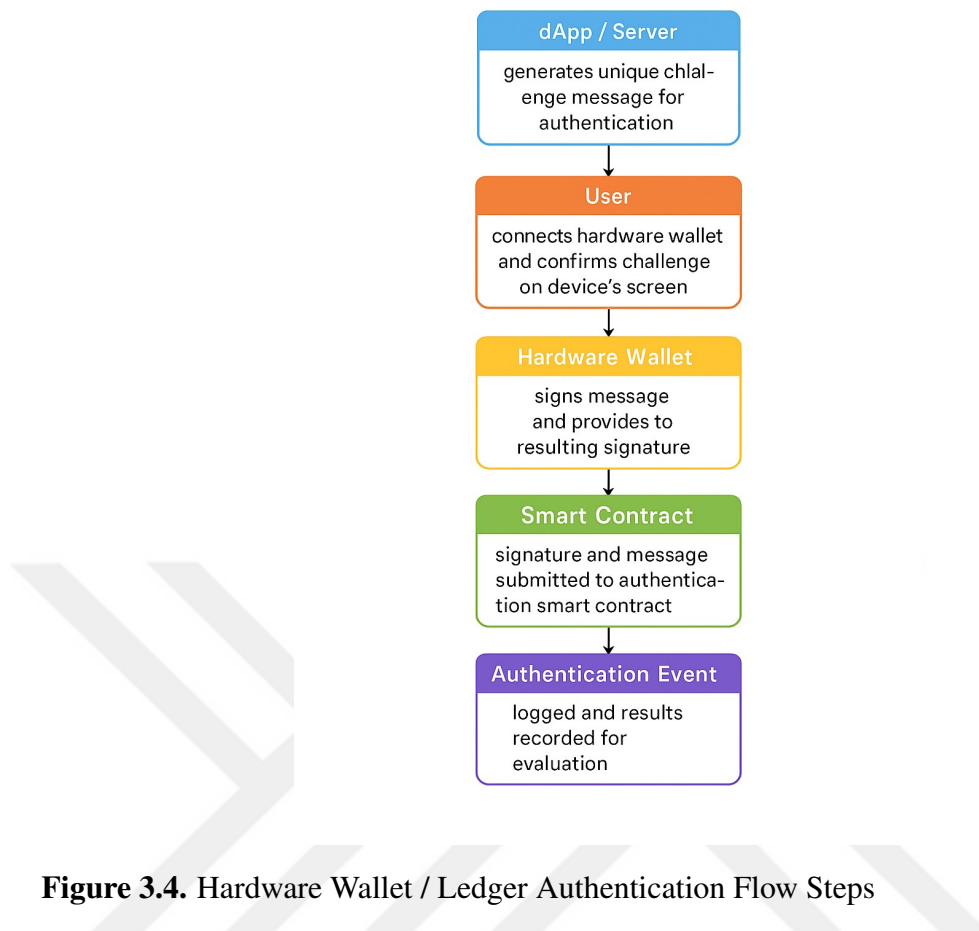


Figure 3.4. Hardware Wallet / Ledger Authentication Flow Steps

3.2.3. NFT-Based Authentication

3.2.3.1 Overview and Use Case

NFT-based authentication leverages the unique, non-fungible properties of ERC-721 tokens to associate user identity and permissions with blockchain wallet addresses. By minting a distinct “Authentication NFT” to each verified user, the protocol enables on-chain, cryptographically provable access control and identity management.

Typical Use Cases:

- Membership or whitelist-based access to decentralized applications
- On-chain voting, DAO participation, or delegated governance
- Exclusive event, content, or feature access for NFT holders
- Persistent identity or role verification without centralized databases

3.2.3.2 Smart Contract Design (ERC-721 and Soulbound Option)

The core of this method is a custom ERC-721 smart contract deployed on the Sepolia testnet. Each user, once successfully verified, is allowed to mint a unique Authentication NFT linked to their address.

Soulbound NFT variants take this a step further by making the minted NFT non-transferable, ensuring it remains “bound” to the original owner and cannot be traded or sold.

```
// contracts/AuthNFT.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.18;

import
↳ "@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol";
import "@openzeppelin/contracts/access/Ownable.sol";

contract AuthNFT is ERC721URIStorage, Ownable {
    uint256 public nextTokenId = 1;

    // Forward the deployer (msg.sender) to both ERC721 and Ownable
    constructor()
        ERC721("AuthNFT", "AUTH")
        Ownable(msg.sender)
    {}

    function mint(address to, string memory tokenURI) public onlyOwner
    ↳ {
        _safeMint(to, nextTokenId);
        _setTokenURI(nextTokenId, tokenURI);
        nextTokenId++;
    }
}
```

Figure 3.5. ERC-721 Custom NFT Contract

Key design features:

- Full ERC-721 compatibility for broad tooling and wallet support
- Each address can mint only one Authentication NFT (Optional)
- On-chain verification of ownership via `ownerOf(tokenId)`

- **Soulbound extension (optional):** Disables `transferFrom` and `approve` functions so NFTs are non-transferable

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.18;

import "@openzeppelin/contracts/token/ERC721/IERC721.sol";

contract NFTAuth {
    IERC721 public immutable nft;
    mapping(address => bool) public authenticated;
    event Authenticated(address indexed user, uint256 tokenId);

    constructor(address _nftAddress) {
        nft = IERC721(_nftAddress);
    }

    function authenticate(uint256 tokenId) external {
        require(nft.ownerOf(tokenId) == msg.sender, "Not NFT owner");
        authenticated[msg.sender] = true;
        emit Authenticated(msg.sender, tokenId);
    }

    /**
     * @dev Check if a user is authenticated based on NFT ownership
     * @param user The address to check
     * @return True if the user owns any NFT from the collection
     */
    function isAuthenticated(address user) public view returns (bool) {
        try nft.balanceOf(user) returns (uint256 balance) {
            return balance > 0;
        } catch {
            // If the balanceOf call fails, fallback to the
            // authenticated mapping
            return authenticated[user];
        }
    }
}
```

Figure 3.6. NFT Authentication Contract

3.2.3.3 Minting and Ownership Verification Workflow

Process:

1. User connects their wallet and initiates an authentication request.
2. The dApp generates a challenge message; the user signs it with their private key.

3. Upon successful signature verification, the contract's `mintAuthNFT()` function is called, minting an Authentication NFT for the user.
4. In subsequent sessions, the dApp or smart contract uses `ownerOf(tokenId)` to check if the user holds the Authentication NFT.
5. If the NFT is **soulbound**, it cannot be transferred, providing a strong guarantee that only the original verified address can use the NFT for authentication.

3.2.3.4 Implementation Details

- **Walled Used:** Hot Wallet is used.
- **Smart Contract:** Custom ERC-721 contract, optionally implementing soulbound logic.
- **Development Tools:** Hardhat, ethers.js for testing and deployment.
- **Minting Flow:**
 - The dApp requests the user to sign a challenge.
 - If the signature is valid, the contract's `mintAuthNFT` method is called.
- **Verification:** Ownership is checked using `ownerOf(tokenId)` or a similar function.
- **Testing Conditions:**
 - All tests conducted on the Sepolia testnet.
 - Each authentication scenario is repeated 10 times to obtain average latency and gas cost data.

3.2.3.5 Example Authentication Flow

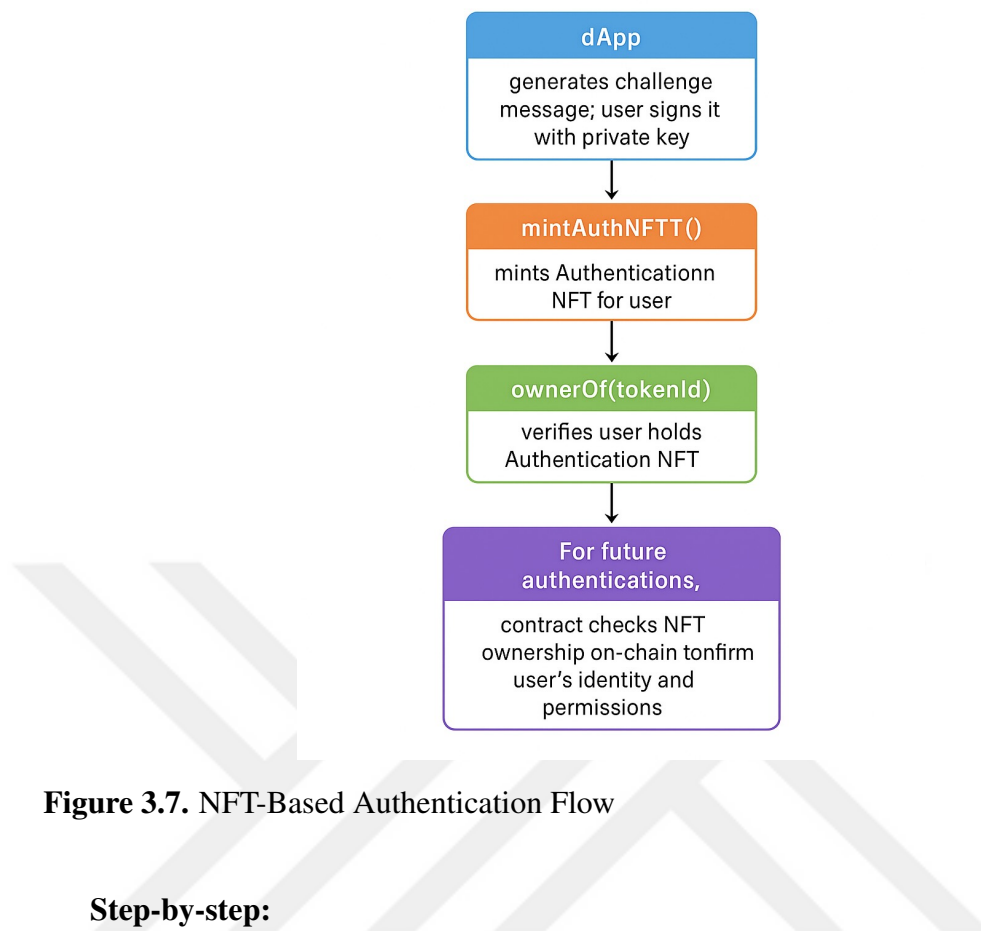


Figure 3.7. NFT-Based Authentication Flow

Step-by-step:

1. User connects to the dApp with their wallet.
2. The dApp issues a challenge; the user signs and submits it.
3. The dApp (or backend) verifies the signature and calls `mintAuthNFT()`.
4. For future authentications, the contract checks NFT ownership on-chain to confirm the user's identity and permissions.

3.2.3.7 Soulbound NFT Considerations

A **Soulbound NFT** is a non-transferable NFT that is permanently bound to the original minting address. This model is particularly valuable for authentication, identity, and reputation systems, as it prevents the selling or transfer of identity credentials. And it is implemented by overriding transfer logic in the ERC-721 contract, making the Authentication NFT strictly non-transferable, thereby improving security and reliability for authentication use cases.

In this study, the soulbound behavior not implemented directly because of the remaining specifications stays the same. But its advantages and disadvantages will be discussed in the next chapters.

3.2.4. Multisig Wallet Authentication

3.2.4.1 Overview and Use Case

Multisignature (multisig) wallets require multiple independent approvals to authorize a transaction or authentication event. This approach distributes control across several parties or devices, significantly enhancing security and reducing single points of failure. Multisig authentication is particularly suitable for organizational access, treasury management, and situations where collective consent is needed for critical actions.

Use Cases:

- DAO or protocol treasury authentication
- Shared access to high-value accounts or resources
- Team, board, or family-controlled assets
- Multi-factor authentication scenarios, combining hot and cold wallets

3.2.4.2 Multisig Contract Design and Threshold Policy

A custom smart contract was implemented to enforce multisignature requirements. In this research, a **2-of-3** threshold policy was adopted, meaning any two out of three registered signers must approve an operation for it to succeed.

Key features:

- Transactions (or authentication events) require a minimum threshold of valid signatures
- Only after verifying the requisite number of signatures does the contract mark the authentication as successful
- Supports combinations of hot and hardware (cold) wallets for real-world flexibility

```

// SPDX-License-Identifier: MIT
pragma solidity ^0.8.18;

contract MinimalMultisigAuth {
    address[] public signers;
    uint256 public threshold;
    mapping(address => bool) public isSigner;
    mapping(address => bool) public authenticated;
    event Authenticated(address indexed user);

    constructor(address[] memory _signers, uint256 _threshold) {
        require(_signers.length >= _threshold, "Signers less than
        ↪ threshold");
        threshold = _threshold;

        for (uint256 i = 0; i < _signers.length; i++) {
            isSigner[_signers[i]] = true;
        }
    }

    function authenticate(bytes32 messageHash, bytes[] memory
    ↪ signatures) public {
        uint256 validSignatures = 0;

        for (uint256 i = 0; i < signatures.length; i++) {
            address recovered = recoverSigner(messageHash,
            ↪ signatures[i]);
            if (isSigner[recovered]) {
                validSignatures++;
            }
        }

        require(validSignatures >= threshold, "Not enough valid
        ↪ signatures");

        authenticated[msg.sender] = true;
        emit Authenticated(msg.sender);
    }

    function isAuthenticated(address user) public view returns (bool) {
        return authenticated[user];
    }
}

```

Figure 3.8. Minimal Multisig Solidty Contract Snippet

3.2.4.3 Signature Collection and Verification Workflow

Workflow:

1. A unique challenge message (or transaction hash) is generated for authentication.

2. Each authorized signer individually signs the message with their wallet (hot or hardware).
3. The set of collected signatures is submitted to the multisig smart contract.
4. The contract verifies that each signature is valid and belongs to a registered signer, and ensures that at least the minimum threshold is reached.
5. Upon reaching the threshold, authentication is marked as successful and an event is emitted.

3.2.4.4 Implementation Details (Hot and Cold Wallets)

- **Wallet Mix:** Testing involved both hot wallets (software-based, for speed) and cold wallets (Ledger hardware, for security) to simulate realistic usage.
- **Signature Aggregation:** Signatures were collected off-chain (via dApp or script) and sent together as an array in the contract call.
- **Development Tools:** Hardhat, ethers.js, Ledger support for cold wallet signatures
- **Testnet:** All tests were conducted on the Sepolia Ethereum testnet.

3.2.4.5 Example Authentication Flow

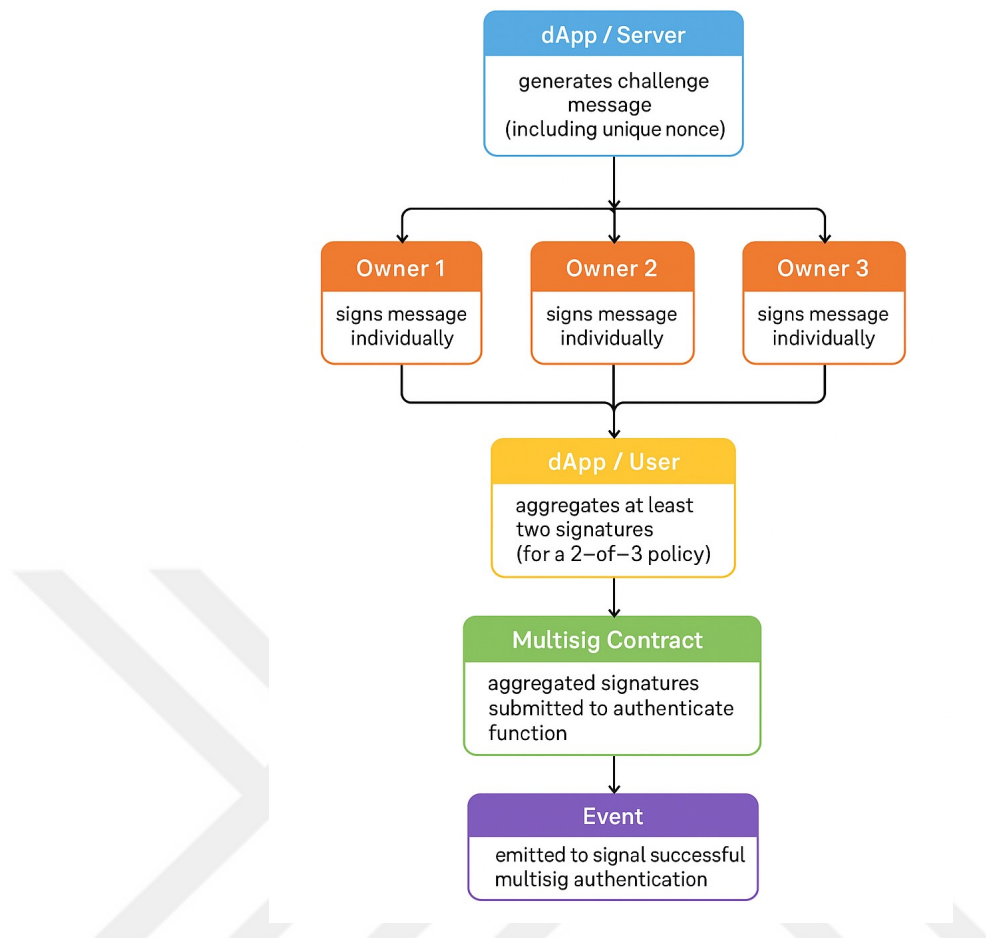


Figure 3.9. Multisig Example Authentication Flow

1. The dApp/server generates a challenge message (including a unique nonce).
2. Each registered owner signs the message individually (hot or hardware wallet).
3. The dApp or user aggregates at least two signatures (for a 2-of-3 policy).
4. The aggregated signatures are submitted to the `authenticate` function of the multisig contract.
5. The contract recovers signer addresses, verifies each against the registered owners, and confirms the threshold is met.
6. An event is emitted to signal successful multisig authentication.

3.2.5. Account Abstraction Based Authentication

3.2.5.1 Overview and Use Case

Account abstraction is an advanced authentication paradigm aiming to overcome the limitations of traditional Externally Owned Accounts (EOAs) by allowing arbitrary logic and programmability for account management. With account abstraction, users are not restricted to the standard EOA signing model; instead, they can deploy “smart contract wallets” that define their own authentication, transaction validation, and access control rules.

Use Cases:

- Programmable wallets supporting multi-factor authentication, biometrics, or social recovery
- Gasless transactions (sponsored by third parties, i.e., paymasters)
- Flexible security policies (timelocks, guardian accounts, rate limits)
- Abstracting hardware/software wallet differences for a unified user experience

3.2.5.2 Account Abstraction Architecture

In the standard EVM model, there are two account types: EOAs (user-controlled, key-pair based) and contract accounts (code-controlled, no private key). Account abstraction blurs this distinction by letting contract accounts act as user wallets, capable of initiating transactions via customizable logic.

Key principles:

- **UserOperations:** Instead of raw transactions, users submit “operation objects” (UserOps) to an “EntryPoint” contract.
- **Validation Logic:** The smart contract wallet validates the operation using arbitrary rules—signatures, multi-factor, whitelists, etc.
- **Execution:** Only upon successful validation does the operation proceed, ensuring full programmability.
- **Bundlers:** Specialized relayers (bundlers) aggregate and submit these operations to the blockchain, enabling batching and gas sponsorship.

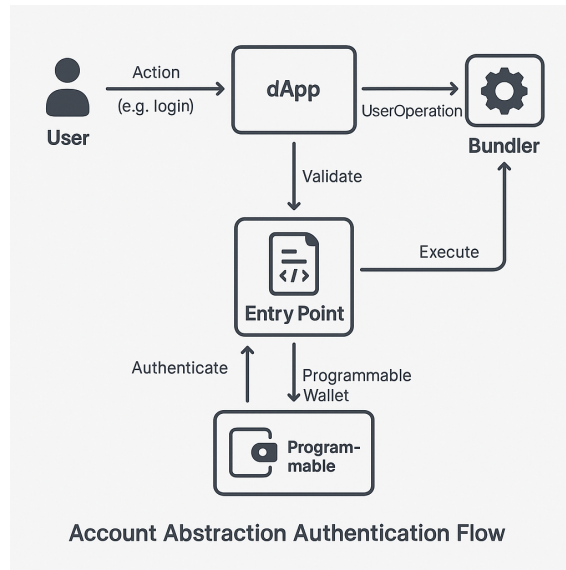


Figure 3.10. Account Abstraction Authentication Flow

1. User initiates an action (e.g., login, transaction) via their smart contract wallet.
2. The wallet creates a UserOperation describing the intent and includes authentication data (signature, proof, etc.).
3. The operation is sent to the EntryPoint contract, which invokes the wallet's custom validation logic.
4. If validation passes, the requested operation is executed on-chain.
5. Optionally, a paymaster can cover the gas cost, enabling gasless UX.

3.2.5.3 Implementation using Etherspot SDK

For this research, **Etherspot SDK** was used to abstract away much of the low-level complexity and provide a high-level interface for deploying and interacting with account abstraction wallets.

```
// Create the EOA wallet from the private key
const eoaWallet = new ethers.Wallet(privateKey);
console.log('EOA Address:', eoaWallet.address);

// Initialize PrimeSdk with the same private key
console.log('Initializing PrimeSdk with chain ID:', chainId);
const sdk = new PrimeSdk({ privateKey }, { chainId });

// Get the counterfactual address (Smart Account)
const scwAddress = await sdk.getCounterFactualAddress();
console.log('Smart Account Address:', scwAddress);
```

Figure 3.11. Account Abstraction Initialize Using SDK in Typescript

Key steps:

- **Wallet Deployment:** Etherspot creates a smart contract wallet on behalf of the user, which can be configured with custom signing rules or multi-signature logic.
- **Authentication:** Authentication is handled by the wallet contract's internal validation logic, which may require signatures from EOAs, multisigs, or even off-chain proofs.
- **Interaction:** The SDK enables interaction with these wallets using familiar JavaScript / TypeScript APIs, hiding much of the bundling, relaying, and UserOperation management from the developer.
- **Paymaster Support:** Etherspot can facilitate gasless transactions by leveraging paymaster contracts, providing an improved user experience for onboarding and frequent operations.

```

// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract AuthContract {
    mapping(address => string) private usernames;
    mapping(address => bool) private authenticated;
    mapping(address => mapping(address => bool)) private
    ↪ authorizedSigners;

    event UserRegistered(address indexed smartAccount, string
    ↪ username);
    event UserAuthenticated(address indexed smartAccount, address
    ↪ indexed signer);

    function registerUser(address userSmartAccount, string memory
    ↪ username) external {
        require(bytes(username).length > 0, "Username cannot be
        ↪ empty");
        usernames[userSmartAccount] = username;
        emit UserRegistered(userSmartAccount, username);
    }

    function authenticate(address smartAccount, bytes memory signature,
    ↪ bytes32 messageHash) external returns (bool) {
        bytes32 r;
        bytes32 s;
        uint8 v;

        assembly {
            r := mload(add(signature, 32))
            s := mload(add(signature, 64))
            v := byte(0, mload(add(signature, 96)))
        }

        address signer = ecrecover(messageHash, v, r, s);
        bool isAuthorized = authorizedSigners[smartAccount][signer];

        if (isAuthorized) {
            authenticated[smartAccount] = true;
            emit UserAuthenticated(smartAccount, signer);
        }

        return isAuthorized;
    }

    function isAuthenticated(address smartAccount) external view
    ↪ returns (bool) {
        return authenticated[smartAccount];
    }
}

```

Figure 3.12. Account Abstraction Authentication Contract Code Snipped

3.2.5.4 Example Authentication Flow

1. User interacts with the dApp, requesting authentication or transaction.
2. The dApp (via Etherspot SDK) creates a UserOperation with required authentication data.
3. UserOperation is submitted to the EntryPoint contract by a bundler/relayer.
4. The EntryPoint calls the user's smart contract wallet, which runs custom validation (e.g., signature verification, 2FA, device checks).
5. Upon successful validation, the intended action is executed (e.g., access granted, transaction sent).
6. The process is fully programmable and can include additional steps, such as paymaster interaction for gas sponsorship.

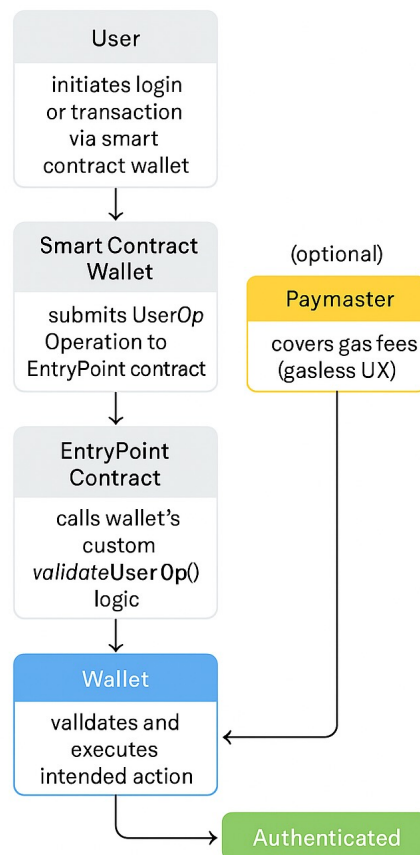


Figure 3.13. Account Abstraction Authentication Example Workflow

3.3. Testing Environment and Methodology

3.3.1. Test Network and Deployment

All authentication contracts and supporting infrastructure were deployed and tested on the Ethereum Sepolia testnet. The Sepolia network provides a realistic EVM-compatible environment, allowing for live contract interactions and gas usage measurement without the risk or cost associated with mainnet deployments.

Contract deployment and upgrade operations were performed using the Hardhat framework, ensuring deterministic build environments and full deployment reproducibility. Deployment scripts automated the creation of required smart contracts (EOA, NFT, multisig, account abstraction wallets) and handled contract registration and initial configuration for each test cycle.

3.3.2. Test Automation and Scripted Workflows

Custom JavaScript/TypeScript scripts were developed to automate the full authentication lifecycle for each method under test. These scripts managed:

- Wallet instantiation (hot wallet via direct private key, hardware wallet via Ledger APIs, account abstraction via Etherspot)
- Challenge message generation and signature requests
- Transaction assembly, signature collection (for multisig), and on-chain contract calls
- Automated repetition of each test scenario to collect robust performance and gas usage data

Scripts were executed locally with Node.js and ethers.js, and integrated with Ledger Nano S/X for cold wallet testing using official USB APIs. CSV files were generated during these tests, logging metrics such as sign time, verify time, and gas cost for each operation. These raw results form the basis for subsequent evaluation of criteria such as user experience, complexity, and decentralization (discussed in the next section).

3.3.3. Performance Measurement Procedure

For each authentication method, the following metrics were measured and logged:

- **Sign Time:** Time from challenge generation to cryptographic signature (ms)
- **Verify Time:** Time from signature submission to on-chain contract verification (ms)
- **Gas Cost:** Amount of gas consumed per authentication transaction

Each scenario was repeated at least 10 times to account for network variability and statistical robustness. Outliers were removed, and both mean and median values were reported for comparative analysis.

3.3.4. Test Conditions and Exclusions

All tests were performed on a single, controlled personal computer and stable network conditions to minimize external variability.

- For hot wallet and account abstraction tests, private keys were managed directly in the test scripts for consistency.
- Direct private key usage to reduce external variable influence.
- For NFT and multisig, custom contracts were freshly deployed per test cycle to ensure isolated results and specifically designed and optimized for performance testing.
- For Account Abstraction, implemented through Etherspot SDK, abstracting away complex EVM interactions.
- Hardware wallet testing used an official Ledger Nano S Plus device with up-to-date firmware, connected via USB.
- Only EVM-compatible signature schemes (secp256k1, EIP-191) were evaluated.

3.3.5. Data Collection and Analysis

All measured results (sign time, verify time, gas usage, gas cost) were exported to structured CSV files for later analysis. Full scripts, contract codes, and sample CSV logs are included in the project appendix and repository to ensure reproducibility.

Detailed contract architectures and scripts are presented in each method's respective section above. This section focuses on the overall testing infrastructure and evaluation procedure, providing a transparent foundation for the subsequent comparative analysis of authentication approaches. All implementations, test scripts, and supplementary data used in this study are available in the open-access repository:

<https://github.com/hakandurmaz/blockchain-auth-methods-evaluation>

3.4. Evaluation Criteria

To ensure an objective and comprehensive comparison of the authentication methods evaluated in this study, a multi-dimensional set of criteria was established. Each method was evaluated using both technical and user-focused measures, combining quantitative data collected during experiments with qualitative assessments based on user experience and system design. The selected criteria are aligned with best practices in decentralized authentication research and are intended to reflect the real-world requirements for deploying such systems.

3.4.1. Criteria and Scoring Approach

The following criteria were used to evaluate each authentication method:

Speed (Sign and Verify Times):

This criterion captures how quickly an authentication method can generate a signature (sign time) and how rapidly the system can verify it (verify time). Authentication times for both signing and verification processes were recorded in milliseconds over repeated test executions. To represent the typical performance of each method, the arithmetic mean of the recorded durations was computed and reported.

Gas Usage:

Gas usage and gas cost reflects the amount of gas consumed by each on-chain authen-

tication transaction. Measurements were collected using the Sepolia testnet and Hardhat, with results expressed in gas units and, where applicable, converted to mainnet cost estimates.

Security:

Security scores reflect each method's overall resistance to key exposure, phishing, replay attacks, and contract-level vulnerabilities. Methods were rated from most vulnerable: e.g., exposed keys, no protection to most secure: e.g., hardware and multisig protection, non-transferable credentials, robust contract checks, based on key protection mechanisms and implemented security features in real test scenarios.

Hardware Dependency:

This criterion considers whether authentication requires specific hardware, such as a Ledger device. Each method was rated as high, medium, or low, depending on the necessity and practical impact of external devices on usability.

User Experience (UX):

User experience evaluates the overall simplicity, intuitiveness, and accessibility of the authentication flow. Criteria include the number of steps required, manual actions (such as device confirmations), and usability observed in test scripts. Each method was qualitatively scored from excellent to poor.

Scalability:

Scalability measures a method's ability to handle large numbers of simultaneous authentications or users without sacrificing performance or usability. Assessment involved stress testing, reviewing the code base, and analyzing system architecture.

Complexity:

Complexity reflects both implementation and operational challenges for users and developers. This includes the number of required actions, codebase size, off-chain coordination, and any custom logic or integration required.

Interoperability:

Interoperability indicates how easily a method integrates with different dApps, platforms, and blockchain environments. Compatibility with standards such as EIP-721 or EIP-4337, and ease of integration with common wallets, were key factors in this assessment.

Privacy:

Privacy assesses how much personally identifiable or sensitive data is exposed on-chain and the risk of tracking or data leakage. Evaluation was based on contract architecture and the degree to which authentication data is public or protected.

Future Proof:

This criterion reflects the expected longevity and adaptability of each method with respect to evolving threats (such as quantum computing), protocol upgrades, or new standards. Flexibility for post-quantum upgrades and standard adoption were considered.

Decentralization Level:

Decentralization measures how effectively a method eliminates single points of failure and avoids centralized control or dependencies. Assessment involved analyzing contract deployment, signer distribution (e.g., single key versus multisig), reliance on external relays or bundlers, and the extent to which any step requires trust in a third party.

3.4.2. Scoring Summary Table

Each authentication method is scored and compared across all criteria using a combination of direct measurements (for speed, gas, etc.) and qualitative assessments (for security, UX, decentralization, etc.). A summary table (see Chapter 4) presents all scores side by side to facilitate direct comparison.

3.4.3. Rationale for Criteria Selection

The chosen criteria are designed to provide a holistic evaluation of both technical performance and real-world usability. They reflect common concerns in the blockchain authentication landscape, including security, cost, and scalability, as well as emerging priorities such as privacy, hardware-independence, and post-quantum readiness.

3.4.4. Scoring Methodology and Criteria Overview

To ensure a transparent and reproducible comparison of authentication methods, all evaluation criteria are scored on a scale from 1 (worst) to 5 (best) according to clear, pre-defined rubrics.

Table 3.1 summarizes the main criteria, scoring scale, and the principal factors considered for each. Detailed scoring rubrics and sub-criteria are provided in Appendix A.

Table 3.1. Criteria, Scoring Scale, and Key Factors

Criterion	Scoring (1–5)	Key Factors / Sub-Criteria
Security	1: Very poor, 5: Max	Key protection, replay, phishing, contract
User Experience	1: Difficult, 5: Best	Steps, onboarding, error risk, clarity
Gas Cost	1: Highest, 5: Lowest	Gas used per authentication
Complexity	1: Complex, 5: Simple	Steps, integration, codebase
Hardware Dependency	1: Always, 5: Never	Device requirements
Decentralization	1: Centralized, 5: Fully Decentralized	Single point of failure, distributed approval
Scalability	1: Poor, 5: Excellent	Users, requests, throughput
Interoperability	1: Isolated, 5: Broadest	Wallet/dApp/chain compatibility
Privacy	1: Poor, 5: Excellent	Data exposure, linkability
Future Proof	1: Low, 5: PQ-ready	Upgradeability, quantum resistance

Scoring for all criteria follows a consistent 1–5 scale, as detailed in the results and discussion chapter, where 5 always represents the most favorable outcome. For positive attributes (e.g., security, UX), a score of 5 indicates maximum strength; for negative attributes (e.g., gas cost, complexity), a score of 5 indicates minimum impact (i.e., lowest cost or least complexity).

Where possible, criteria such as Speed and Gas Cost are also reported as quantitative values for additional context.

For detailed scoring rules and sub-criteria definitions, see Appendix A.

4. RESULTS AND DISCUSSION

4.1. Overview

This chapter presents a comprehensive comparison of the blockchain-based authentication methods evaluated in this research. The analysis combines both empirical measurements—such as signing time, verification time, and on-chain gas cost—directly obtained from experimental tests, and qualitative criteria—such as security, hardware dependency, user experience, scalability, complexity, privacy, future proof and decentralization—derived from both observed behaviors and a review of relevant literature.

While the measured metrics provide objective insights into the performance and efficiency of each method, other evaluation criteria reflect best practices and inherent features reported in academic and technical references. For each authentication method, the results section summarizes quantitative findings and complements them with comparative qualitative assessments.

By integrating experimental data with literature-informed analysis, this chapter aims to highlight not only the practical performance differences but also the design trade-offs, security implications, and usability characteristics that influence the real-world adoption of these authentication solutions. The combined approach offers a holistic perspective for developers, system architects, and researchers seeking to select or design secure and efficient authentication mechanisms for blockchain environments.

4.2. Summary of Results

Table 4.2. Comparative Results for Authentication Methods

Method	Sign Time (ms)	Verify Time (ms)	Gas Used (gas)	Gas Cost (gwei)	Sec. (1-5)	UX (1-5)	HW (1-5)	Compl. (1-5)	Scal. (1-5)	Inter. (1-5)	Priv. (1-5)	FP (1-5)	Decent. (1-5)
Hot Wallet	3.3	11694.3	38809.6	38.81	3	5	5	5	5	5	3	3	2
Cold Wallet	2675.5	13428.2	38806.0	166.06	4	3	1	2	2	4	3	3	3
NFT	7.1	11712.7	33803.0	33.80	4	4	5	4	5	4	3	4	4
Multisig	3580.8	11079.73	45844.3	46.67	5	2	3	1	3	3	2	4	5
Account Abstr.	4.5	12762.4	34951.8	113.13	4	4	5	1	4	3	3	5	5

Table 4.2 presents a holistic comparison of the evaluated authentication methods, highlighting both their quantitative performance metrics and their qualitative attributes (*Measurement details are provided in Section 3.4 and Appendix A*). The results reveal several distinct patterns and trade-offs:

Software (Hot) Wallets exhibit the lowest signing and verification times, as expected for software-only key management. Their lack of hardware dependency translates into superior user experience and accessibility, reflected in the highest scores for UX and hardware independence. However, their moderate security and decentralization scores highlight the vulnerability of software-managed private keys to malware, phishing, or device compromise. While ideal for rapid, frequent transactions in low-risk environments, they may not meet the requirements for high-security scenarios.

Hardware (Cold) Wallets significantly increase the time required to sign transactions, due to the necessary manual confirmation on the device. Although this added step introduces friction and reduces user convenience, it greatly enhances private key protection by isolating sensitive operations from the host device. The need for a dedicated hardware device lowers accessibility and hardware dependency scores. Security, however, is improved over software wallets, making hardware wallets a preferred choice for users prioritizing asset protection over speed and convenience.

NFT-Based Authentication methods require an initial on-chain transaction (minting), incurring a one-time gas cost and setup delay. This is reflected in their moderate process times and gas usage. Once established, however, subsequent authentications can often be performed off-chain, with only a cryptographic proof of NFT ownership required—effectively reducing ongoing costs to zero. NFT methods score highly in scalability, hardware independence, and privacy, making them well-suited for systems that require persistent, non-transferable credentials or unique user identities (e.g., membership, reputation, or proof-of-attendance systems).

Multisignature Wallets provide unmatched security and decentralization, as they require multiple parties to approve each action. This design is highly resistant to single points of failure, collusion, and device compromise. However, it comes with substantial trade-offs: signing and verification take longer (especially when hardware wallets are included in the signer set), user experience is more complex, and operational overhead

is increased. Hardware dependency is moderate, depending on the mix of software and hardware signers in the setup. Multisig authentication is optimal for high-value assets or governance actions but may be excessive for everyday transactions.

Account Abstraction-Based Authentication stands out for its flexibility and programmability. It enables advanced features like batched transactions, session keys, social recovery, and sponsor-paid gas, allowing for highly customizable authentication experiences. The method achieves excellent scores in decentralization, scalability, future-proofing, and interoperability, as account abstraction is designed to adapt to evolving blockchain standards. However, average verification time is higher due to the complexity of smart contract execution and potential relayer involvement. Still, the user experience can be tailored to minimize friction and optimize security as required.

Comparative Analysis of Criteria:

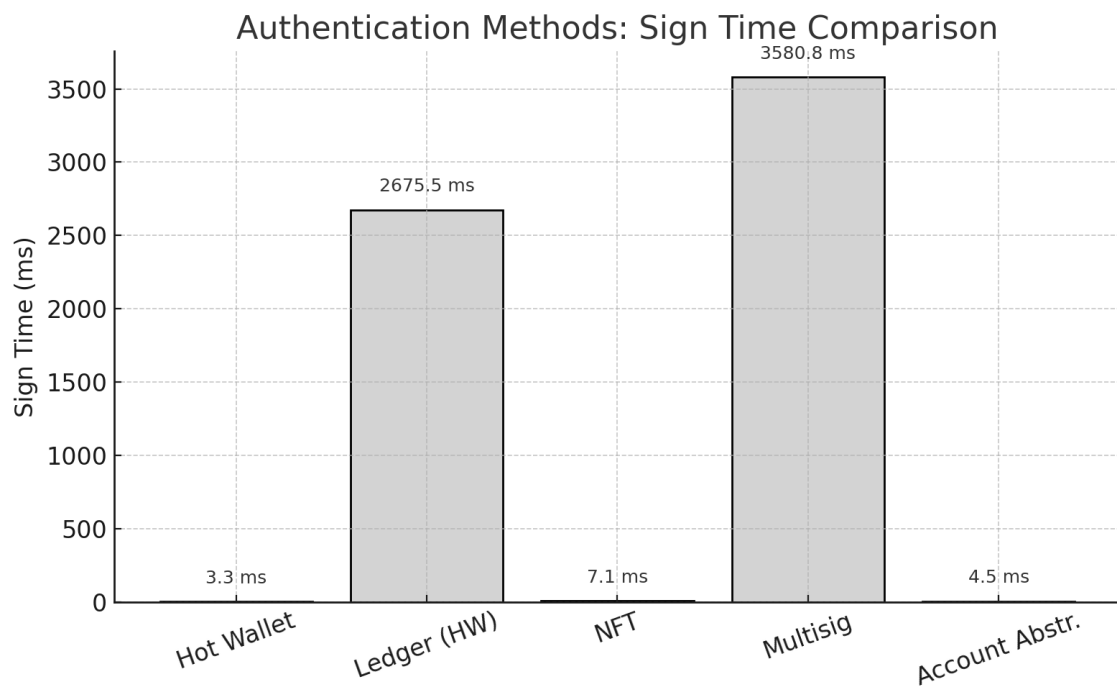


Figure 4.14. Sign Time Comparison

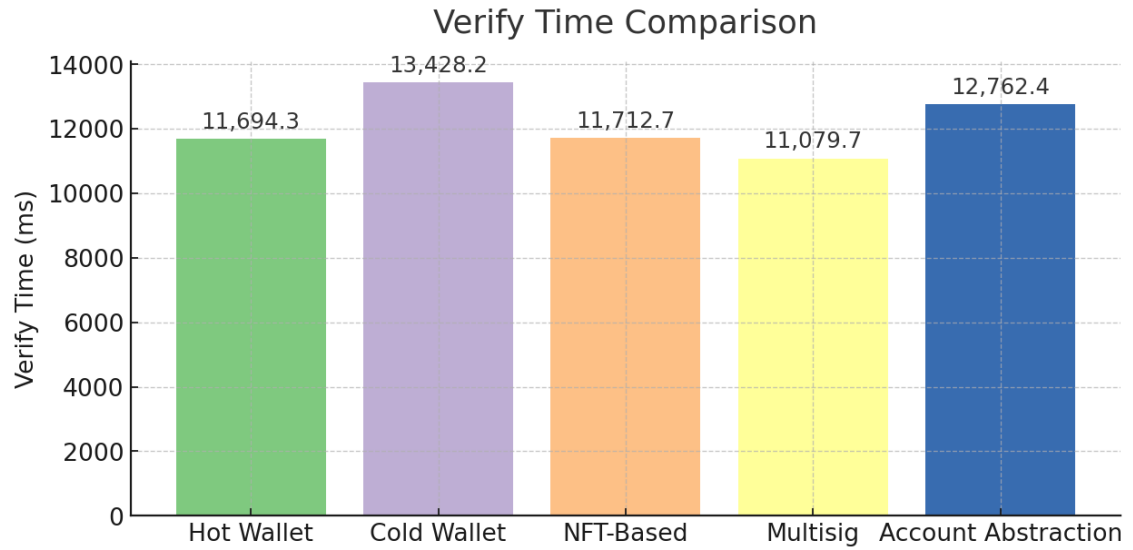


Figure 4.15. Verify Time Comparison

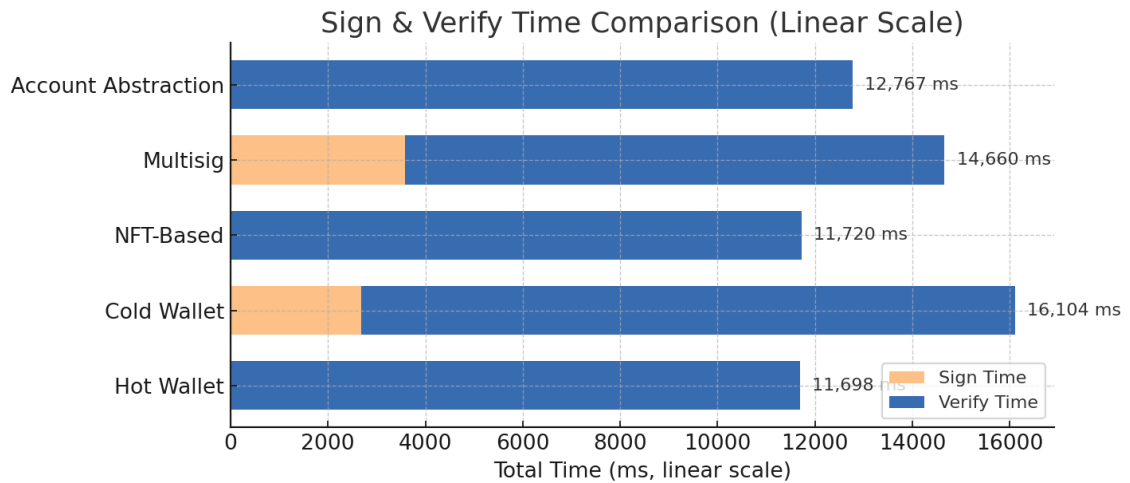


Figure 4.16. Total Time Comparison

Speed: As we see in the Figures 4.14, 4.15 and 4.16, signing times and authentication times are faster in the software based wallets. Account abstraction also achieved fast signing but higher verification latency (due to additional contract logic and bundler relays). It should be noted that in multisignature authentication, as the threshold (the required number of signatures) increases, both the total signing time and the overall authentication latency tend to increase.

Security: Multisig and hardware wallets are the most secure, while hot wallets are the most vulnerable to device-level compromise.

User Experience: Hot wallets excel in speed and simplicity; multisig is the most complex for end-users.

Hardware Dependency: Software and NFT-based approaches are most accessible; hardware wallets and multisig setups may require specialized devices.

Complexity: Multisig wallets and account abstraction methods has the most complexity due to their setups and integration steps.

Scalability & Interoperability: Account abstraction and NFT methods scale well and can integrate with diverse dApps.

Gas Usage: As shown in Figure 4.17 All methods incur initial gas usage for registration or setup, but subsequent authentications can typically be performed gas-free unless state changes are required. Multisig needs the most gas usage.

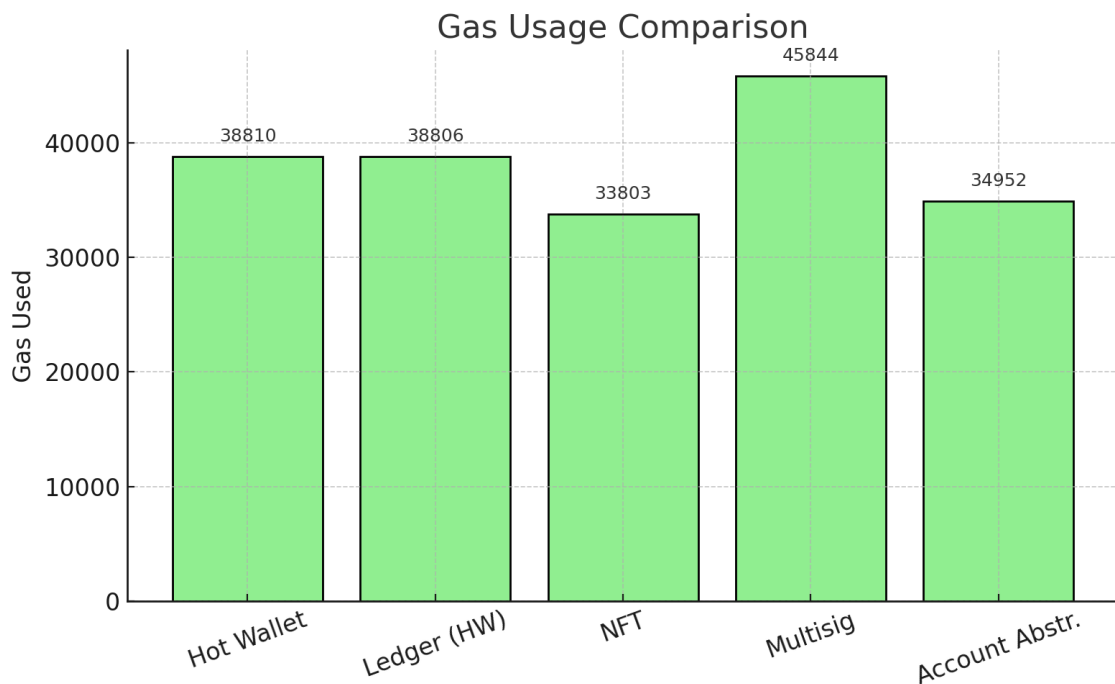


Figure 4.17. Gas Usage Comparison

Future-Proofing & Decentralization: Account abstraction and multisig offer the greatest resilience to evolving blockchain environments and censorship.

This integrated evaluation highlights that the "best" authentication solution is highly context-dependent. Factors such as application risk profile, user base, cost sensitivity, and regulatory requirements must be weighed against each method's inherent strengths and

limitations. No single method dominates across all criteria, but the comparative framework established here enables informed, scenario-specific selection and implementation of blockchain authentication mechanisms.

4.3. Discussion

This section provides an in-depth analysis of the comparative results presented in Table 4.2, examining the trade-offs, operational implications, and practical suitability of each blockchain-based authentication method. The discussion synthesizes both the quantitative performance metrics and qualitative evaluation scores to uncover the strengths and weaknesses inherent to each approach.

The diversity of authentication mechanisms—ranging from basic software wallets to advanced account abstraction schemes—reflects the evolving landscape of blockchain security and usability requirements. Each method exhibits a unique profile shaped by factors such as signing latency, verification speed, on-chain transaction costs, hardware dependency, and the level of decentralization. Importantly, qualitative attributes such as security robustness, user experience, privacy guarantees, and future adaptability are often as influential as raw performance metrics in determining the appropriateness of a method for a given application context.

Key questions addressed in this discussion include:

- Which authentication methods deliver the best balance between security, usability, and operational cost?
- How do hardware and software-based solutions compare in real-world scenarios, especially with respect to user accessibility and threat models?
- What are the practical limitations of each approach in terms of scalability, interoperability, and long-term sustainability?

By systematically interpreting the results through these lenses, this section aims to provide actionable insights for developers, system architects, and researchers seeking to implement secure, efficient, and user-centric authentication systems on blockchain platforms.

4.3.1. Software (Hot) Wallet Authentication

Overview and Performance:

Hot wallet authentication represents the most accessible and straightforward method among blockchain authentication options. In the experiments, hot wallets demonstrated the lowest signing and verification times, with average signing latency of only 3.3 ms and verification around 115 ms. Gas consumption and transaction costs were also among the lowest, with no hardware dependency and minimal setup complexity.

Security Considerations:

Hot wallets store private keys in device memory, typically in software clients or browser extensions. While this greatly increases user convenience and onboarding speed, it also exposes private keys to device-level attacks such as malware, phishing, or unauthorized access. Once compromised, a private key grants full control of the wallet's assets, making hot wallets inherently more vulnerable than hardware-based or multi-party methods. The method received only moderate security and decentralization scores for this reason.

User Experience and Scalability:

From a user perspective, hot wallets are extremely easy to use—most users can authenticate or sign within seconds, without needing special hardware or extensive setup. This ease of use makes hot wallets particularly suitable for applications with high transaction frequency, small asset values, or onboarding of non-technical users. Scalability is also high, since no external device or multi-party coordination is required.

Limitations and Risks:

The simplicity of hot wallets is a double-edged sword: although convenient, the lack of hardware-backed key protection and centralized failure points (if used in custodial services) create security risks. Furthermore, the absence of advanced features such as multi-signature, session keys, or programmable rules may limit their use in high-stakes or institutional settings.

Application Scenarios:

Hot wallets are ideal for:

- Low-value, high-frequency transactions,

- DApps where onboarding speed and simplicity are critical,
- Situations where users are willing to accept moderate security in exchange for speed and convenience.

And they are less suitable for:

- high-value assets,
- organizational treasuries,
- Any scenario where security and non-repudiation are paramount.

Hardware Dependency:

Hot wallets operate entirely in software, requiring no specialized hardware. This maximizes accessibility and minimizes onboarding barriers.

Complexity:

Integration and usage are straightforward for both developers and end-users, with no need for hardware setup, recovery phrases beyond seed backup, or multi-party coordination.

Interoperability:

These wallets are compatible with a wide range of dApps, blockchain networks, and third-party services, provided they support standard EVM-compatible cryptography.

Privacy:

Privacy is dependent on both wallet implementation and user device security. While no additional personal data is required for operation, device compromise can lead to key exposure and loss of privacy.

Future Proof:

Hot wallets are flexible and can be rapidly updated to support new cryptographic standards, blockchain forks, or protocol changes. However, their exposure to evolving malware and attack vectors may represent a long-term risk.

Decentralization Level:

When operated non-custodially, hot wallets allow users to retain full control over their assets and private keys. Custodial hot wallet solutions, however, reintroduce centralization risks.

4.3.2. Hardware (Cold) Wallet Authentication (Ledger)

Overview and Performance:

Hardware wallets, such as Ledger devices, store private keys in a secure hardware module, physically isolated from the host device. In testing, Ledger authentication showed a significant increase in signing time (avg. 2675 ms) compared to hot wallets, due to required user confirmation and secure element processing. Verification times were also higher than software wallets but still within acceptable limits for typical transactions. Gas costs were comparable to hot wallets.

Security Considerations:

Ledger devices dramatically reduce the risk of private key exposure by never allowing keys to leave the hardware. Signing operations must be physically confirmed on the device, protecting against most forms of remote malware, phishing, and device compromise. However, residual risks remain from supply chain attacks, compromised firmware, or insecure backup handling of recovery phrases. Physical theft or loss of the hardware wallet also presents a security concern if the recovery phrase is accessible.

User Experience and Scalability:

The requirement to physically connect and confirm actions on a Ledger device increases operational friction and lengthens the signing process. While this enhances security, it may frustrate users unaccustomed to hardware wallets, particularly for frequent or time-sensitive transactions. Scalability is impacted by the need for physical devices and potential delays during periods of high transaction volume or multi-device use.

Limitations and Risks:

Hardware wallets are susceptible to loss, theft, or damage, which can result in asset loss if recovery phrases are not properly stored. Users must also trust the device manufacturer to ensure firmware security and hardware integrity. Supply chain attacks and social engineering (e.g., tricking users into entering recovery phrases into malicious apps) are additional risks.

Application Scenarios:

Hardware wallets are best suited for:

- Long-term storage of high-value assets,

- Institutional or organizational accounts with stringent security requirements,
- Users who prioritize maximum private key protection over speed and convenience.

They are less appropriate for high-frequency, low-value transactions or onboarding non-technical users.

Hardware Dependency:

Operation is dependent on physical access to a compatible hardware wallet (e.g., Ledger device), increasing both security and logistical requirements.

Complexity:

Usage involves device setup, regular firmware updates, safe recovery phrase storage, and manual transaction confirmation, raising both technical and operational complexity compared to software wallets.

Interoperability:

Ledger and similar hardware wallets support standard signing algorithms (secp256k1, etc.), but may have limited compatibility with newer protocols or custom dApps unless supported by vendor firmware updates.

Privacy:

Hardware wallets enhance privacy by keeping keys offline; however, privacy is only preserved if users avoid phishing attempts and do not leak recovery phrases or device metadata.

Future Proof:

Firmware updates allow some adaptation to new blockchain standards, but users depend on the manufacturer for long-term support and compatibility with protocol changes.

Decentralization Level:

Non-custodial hardware wallets maintain decentralization at the user level. However, device supply chain and vendor support introduce minor centralization dependencies.

4.3.3. NFT-Based Authentication

Overview and Performance:

NFT-based authentication leverages the unique ownership of ERC-721 tokens to verify user identity or access rights. In experimental tests, the initial on-chain authentication

(minting the NFT) showed moderate signing (avg. 7.1 ms) and verification times (avg. 11712.7 ms). Gas costs were comparable to other on-chain operations, but importantly, subsequent authentications required no further gas expenditure since only off-chain ownership checks were needed.

Security Considerations:

NFT authentication ties identity or access to the unique, verifiable ownership of a non-fungible token. Security benefits include immutability and transparency of token ownership on-chain. However, the method inherits all wallet security properties: if the underlying wallet (hot or hardware) is compromised, so is access to the NFT. Additionally, standard NFTs can be transferred, which may be undesirable for persistent identity use cases—though non-transferable SoulBound NFTs (SBTs) address this concern by enforcing non-transferability at the contract level.

User Experience and Scalability:

Initial setup requires minting the NFT, which incurs a one-time on-chain transaction cost and delay. After setup, authentication is seamless and gas-free: users only need to prove ownership of the NFT, typically through a wallet signature. Scalability is high, as subsequent checks are off-chain and can be performed instantly for any number of users.

Limitations and Risks:

Transferable NFTs can be sold or gifted, undermining their use for identity or non-repudiation unless transfer restrictions are coded (as with SoulBound NFTs). Security depends on the wallet used to hold the NFT. Loss of wallet access results in loss of identity/rights unless recovery mechanisms are in place. Contract bugs could also potentially allow unauthorized transfers.

Application Scenarios:

NFT-based authentication is suitable for:

- Membership management, access control, or event ticketing,
- Persistent, blockchain-based identity systems (especially with SoulBound NFTs),
- Applications requiring proof of unique ownership or eligibility.

Less suitable for:

- High-value financial transactions (unless combined with other security methods),
- Scenarios requiring frequent identity changes.

Hardware Dependency:

No specialized hardware is required; the method is compatible with both hot and hardware wallets, depending on user preference.

Complexity:

Setup complexity is moderate due to the initial minting process and smart contract deployment. User operations are simple after setup.

Interoperability:

ERC-721 and similar standards are widely supported across EVM-compatible chains and dApps. Compatibility may be affected by custom NFT standards or extensions.

Privacy:

NFT ownership is publicly visible on-chain, which may reveal participation, membership, or activity unless privacy-preserving techniques are used. Off-chain authentication does not expose extra data, but wallet address linkage is possible.

Future Proof:

NFT standards are well established and widely supported. Non-transferable SoulBound NFTs offer additional flexibility for identity use cases and are likely to see broader adoption in the future.

Decentralization Level:

NFT authentication is fully decentralized as long as minting and ownership verification rely on public smart contracts and non-custodial wallets.

SoulBound NFTs (SBTs):

A key limitation of standard NFTs in authentication is their transferability: users can transfer, sell, or lose their identity credential by moving the NFT to another address. SoulBound NFTs (SBTs) address this problem by enforcing non-transferability at the contract level, meaning the NFT is permanently bound to a specific wallet and cannot be sent to another account.

This property makes SBTs particularly well-suited for applications where identity, membership, or reputation must remain persistent and non-transferable—such as diplo-

mas, certifications, voting rights, or access control. SBTs enhance non-repudiation, since only the original holder of the wallet can prove ownership and benefit from the associated rights.

Advantages of SoulBound NFTs:

- Prevents the sale or transfer of identity credentials, ensuring authenticity and persistence.
- Enables stronger compliance with regulatory and application requirements for non-transferable rights.
- Enhances security for systems requiring permanent, individual-linked identity.

Limitations:

- Wallet loss or compromise may result in permanent loss of credentials unless recovery mechanisms are implemented.
- Adoption of SBT standards is still evolving; not all dApps or marketplaces support SBT features natively.

Summary:

SoulBound NFTs provide a powerful tool for building robust, persistent, and non-transferable authentication and identity systems on-chain. When non-transferability is a requirement, SBTs should be preferred over standard ERC-721 NFTs.

4.3.4. Multisignature (Multisig) Authentication

Overview and Performance:

Multisig authentication requires two or more distinct signers to collectively authorize an action or transaction. In this study, a 2-of-3 multisig configuration was evaluated. Signing time was substantially higher (avg. 3580.8 ms) than single-signature methods due to multiple signature collection steps, particularly when a hardware wallet was included. Verification was still efficient (avg. 11079.7 ms), and gas costs were moderately higher, reflecting the increased on-chain complexity. As the threshold (the required number of

signatures) increases, both the total signing time and the overall authentication latency tend to increase.

Security Considerations:

Multisig dramatically increases security by eliminating single points of failure. Even if one signer's key is compromised, an attacker cannot gain control without colluding with or compromising additional parties. This design is highly resistant to malware, device theft, and targeted attacks. However, security is only as strong as the independence of the signers; if all keys are managed by a single user or insecurely distributed, the advantage is lost. Social engineering or collusion between signers also poses residual risks.

User Experience and Scalability:

The need for multiple coordinated signers introduces operational overhead and user friction. For organizations or teams, this can improve internal control and transparency, but for individuals or small groups, the process may be seen as complex and slow. Scalability can be challenging in scenarios where rapid transaction approval is required, but multisig is ideal for high-value, low-frequency actions.

Limitations and Risks:

Operational complexity is the primary drawback: signers must be reliably available, coordinate off-chain or via secure communication, and manage signature collection. Loss of one or more signing keys can lock assets unless robust recovery mechanisms are in place. If any key is hardware-based, all associated logistical risks (loss, firmware, physical damage) also apply.

Application Scenarios:

Multisig authentication is best suited for:

- Organizational or DAO treasury management,
- High-value custody (e.g., institutional funds, shared vaults),
- Scenarios requiring transparent, auditable multi-party approval.

Less appropriate for:

- High-frequency, real-time transactions,

- Environments with only a single trusted user.

Hardware Dependency:

Partially dependent on signer setup; if hardware wallets are included, hardware access is required for those signers. Software-only multisigs can eliminate this dependency.

Complexity:

Multisig solutions have high setup and operational complexity, including signer management, smart contract deployment, and recovery planning.

Interoperability:

Standard multisig contracts (e.g., Gnosis Safe) are widely supported, but custom implementations may face compatibility or upgrade challenges.

Privacy:

On-chain multisig contracts can reveal all participant addresses and approval histories, which may reduce privacy compared to single-user wallets.

Future Proof:

Multisig architectures are well established and compatible with current and emerging EVM standards. They offer strong resilience against protocol-level changes.

Decentralization Level:

Multisig provides the highest level of decentralization and resilience, as no single entity can unilaterally control the assets or actions.

4.3.5. Account Abstraction-Based Authentication**Overview and Performance:**

Account abstraction (AA) introduces programmable smart contract wallets where authentication logic, gas payment, and transaction flows can be customized. Experimental results showed very low signing latency (avg. 4.5 ms) but a higher average verification time (avg. 12,762.4 ms), largely due to smart contract execution and relayer involvement. Gas usage and cost were moderately higher than hot wallets, reflecting more complex contract operations.

Security Considerations:

AA-based wallets can implement advanced security mechanisms such as multi-factor authentication, social recovery, and session keys. These programmable rules enhance

protection against key loss, unauthorized access, and phishing. However, the overall security depends on correct contract implementation and thorough auditing—coding errors or flawed logic could introduce new vulnerabilities. Users may also be exposed to risks associated with the chosen AA framework or relayer services.

User Experience and Scalability:

Account abstraction enables highly flexible user experiences: users can batch transactions, delegate operations, pay gas with any token, or use sponsor-paid authentication. This programmability reduces friction for both new and experienced users. Scalability is high, as contract logic can handle large user bases and complex flows, but real-world latency can vary based on relayer network load and smart contract efficiency.

Limitations and Risks:

AA solutions may introduce complexity for both users and developers, requiring a learning curve and robust support for edge cases. Reliance on third-party relayers for gas sponsorship or transaction submission introduces off-chain trust and operational risk. Standardization is ongoing; not all dApps or protocols support AA features natively.

Application Scenarios:

Account abstraction-based authentication is suitable for:

- Advanced dApps requiring programmable access control, social recovery, or pay-master models,
- User onboarding flows that abstract away gas management,
- Environments needing customizable authentication logic or flexible delegation.
- Environments needing customizable authentication logic or flexible delegation.

Less suitable for:

- Ultra-simple, lightweight applications with minimal on-chain complexity,
- Scenarios requiring complete independence from off-chain relayers.

Hardware Dependency:

No mandatory hardware requirement; AA wallets work with software or hardware keys, depending on implementation. Security can be enhanced by combining with hardware devices.

Complexity:

Setup and operation are more complex than standard wallets due to programmable rules and smart contract interactions. Developer expertise and thorough testing are required.

Interoperability:

AA standards (e.g., ERC-4337) are gaining traction and support, but not yet universal. Compatibility with existing dApps depends on widespread adoption.

Privacy:

Programmable logic can improve privacy (e.g., via stealth addresses, limited disclosure), but on-chain contract activity is still public unless advanced privacy techniques are implemented.

Future Proof:

One of the most significant strengths of Account Abstraction is its programmability and future-proof architecture. Unlike traditional EOAs, smart contract wallets can flexibly adapt to evolving security standards and integrate cutting-edge authentication mechanisms. AA is designed to accommodate new blockchain use cases, enabling the rapid adoption of emerging features and security models. Upgradability and modularity make AA highly future-proof. This flexibility includes native support for biometric hardware (such as TouchID or FaceID), as well as advanced techniques like Zero-Knowledge Proofs (ZKP) and Trusted Execution Environments (TEE). The ability to implement custom validation logic allows for seamless adoption of privacy-preserving, passwordless, or hardware-backed authentication as the ecosystem matures. As AA adoption expands, we expect to see increasingly modular and upgradeable authentication flows leveraging these technologies, ensuring long-term resilience against emerging threats.

Decentralization Level:

AA can support full decentralization, especially with open-source contracts and distributed relayer networks. Centralization risk arises only if proprietary or closed relayer infrastructure dominates.

4.3.6. Best Practice Recommendations

Table 4.3. Criteria-Based Comparative Summary of Authentication Methods

Criteria	Hot Wallet	Cold Wallet	NFT-Based	Multisig	Account Abstraction
Speed	5	2	4	1	3
Gas Cost	4	4	4	2	3
Security	3	4	4	5	4
Hardware Dep.	5	1	5	3	5
UX	5	3	4	2	4
Scalability	5	2	5	3	4
Complexity	5	2	4	1	2
Interop.	5	4	4	3	3
Privacy	3	3	3	2	3
Future Proof	3	3	4	4	5
Decentraliz.	2	3	4	5	5

Note: 5 = Excellent, 4 = Good, 3 = Moderate, 2 = Limited, 1 = Poor

Hot Wallet:

Best for high-frequency, low-value transactions and DApps that prioritize user onboarding and speed over maximum security. Not suitable for storing large assets or for institutional use. Ideal when simplicity and broad wallet compatibility are needed, but not designed for evolving security standards or future-proof features.

Cold Wallet:

Recommended for users and organizations with significant assets and high security requirements. Adds operational complexity but drastically improves key protection. Increasingly supports advanced cryptographic standards and remains highly resilient against both current and emerging threats.

NFT-Based Authentication (including SoulBound NFTs):

Ideal for access control, membership, and persistent identity scenarios where unique and non-transferable credentials are required. Less suitable for rapid, high-value trading. As NFT standards evolve, future implementations may support privacy-preserving and cross-chain identity solutions.

Multisig:

The gold standard for organizational funds, treasuries, and DAOs needing transparent, auditable multi-party control. Complexity and coordination overhead make it less suitable for everyday use. Remains highly robust and adaptable to future governance models and on-chain compliance requirements.

Account Abstraction:

Emerging as the best solution for advanced dApps and programmable wallets, Account Abstraction uniquely enables integration of cutting-edge authentication methods such as biometrics (TouchID, FaceID), Zero-Knowledge Proofs (ZKP), and Trusted Execution Environments (TEE). Its programmability and modular design allow for privacy-preserving, passwordless, and highly customizable authentication flows—making it exceptionally future-proof. Especially recommended for applications that anticipate evolving security needs, adaptive policies, or post-quantum and multi-factor authentication. Note that current deployments may require technical expertise and rely on relayer infrastructure.

4.4. Limitations and Lessons Learned

This study was conducted entirely on the Ethereum Sepolia testnet using a specific set of tools and devices, including Ethers.js and a Ledger hardware wallet. Although this environment closely approximates mainnet conditions, it may not fully reflect the dynamic behavior, costs, and risks of real-world deployments. Results may vary with alternative wallets, devices, firmware versions, or higher network load.

Direct implementation and benchmarking of ZKP and TEE-based environments were excluded due to technical and time constraints, but these technologies are discussed in the context of future integration via account abstraction.

Certain evaluation criteria such as user experience, complexity, and future-proofing were assessed qualitatively and are subject to some degree of subjectivity. The limited number of test runs may also fail to capture rare edge cases. Additionally, system security depends on correct contract implementation, user awareness, and ongoing maintenance; these factors could not be exhaustively simulated. Also qualitative criteria were scored based on literature review, expert assessment, and observed user flows; more robust con-

clusions require broader user studies.

The main practical lessons are: (i) real-world mainnet/user testing is essential for full validation, (ii) frequent review of emerging standards and security best practices is critical, and (iii) high-value applications benefit from a combination of multiple authentication methods.

Another key lesson is that not every authentication component or system needs to be implemented from scratch. Leveraging well-audited, community-vetted frameworks, smart contracts, and dApps can significantly reduce security risks and development time. Where possible, integrating established open-source solutions—rather than reinventing core authentication logic—improves both robustness and maintainability, while allowing researchers and developers to focus on comparative evaluation and innovative features.

4.5. Summary / Key Takeaways

In summary, this comparative analysis showed that each blockchain-based authentication method presents unique strengths and trade-offs. Hot wallets deliver unmatched speed and convenience but are suitable mainly for low-value, frequent transactions. Hardware wallets offer high security for key management, at the cost of added complexity. NFT-based methods enable persistent, decentralized identity—especially with SoulBound NFTs—but raise privacy and transferability considerations. Multisignature wallets provide exceptional security and decentralization for organizational use, but add operational overhead. Account Abstraction stands out for programmability, biometric/ZKP/TEE integration, and future-proofing, although its ecosystem is still maturing.

There is no universal “best” method; the optimal approach depends on application requirements and risk profiles. Hybrid models combining multiple techniques may offer the best resilience. Ongoing monitoring of blockchain standards and regular reassessment of authentication strategies are essential for maintaining robust security and usability as technology evolves.

5. CONCLUSION

This thesis presents a comprehensive evaluation and comparative analysis of alternative blockchain-based authentication methods, with a focus on enhancing system security beyond traditional password-based approaches. By systematically testing and benchmarking software wallets, hardware wallets, NFT-based authentication (including SoulBound NFTs), multisignature contracts, and account abstraction mechanisms on the Ethereum Sepolia testnet, the study offers both quantitative performance insights and qualitative, criteria-based assessments.

The results highlight that there is no single, universally optimal authentication solution. Each method demonstrates distinct trade-offs between speed, security, complexity, decentralization, and user experience. Hot wallets, while offering unmatched convenience and rapid authentication, remain vulnerable to device-level attacks and are best suited for scenarios where operational speed outweighs security demands. Hardware wallets such as Ledger provide substantial improvements in private key security, making them the preferred choice for safeguarding significant assets, yet introduce additional complexity and require user diligence in device management.

NFT-based authentication, and particularly the use of SoulBound NFTs, emerges as a promising approach for decentralized identity and access control, delivering persistent, non-transferable credentials. However, challenges around privacy and on-chain data exposure must be carefully managed, and wallet security remains a limiting factor. Multisignature wallets stand out as the most secure and decentralized solution, particularly valuable for organizational or collective asset management. Their operational complexity and coordination requirements, however, may limit usability for individual users or frequent transactions.

The recent development of account abstraction unlocks new possibilities for programmable, user-centric authentication workflows. By enabling customizable rules, social recovery, gas sponsorship, and native integration with advanced authentication mechanisms—including biometrics (such as TouchID and FaceID), zero-knowledge proofs (ZKP), and trusted execution environments (TEE)—account abstraction bridges the gap between security and usability, and paves the way for privacy-preserving, passwordless,

and post-quantum-ready authentication. Nonetheless, its reliance on evolving standards and third-party infrastructure requires ongoing vigilance and robust implementation.

From a scientific perspective, this work advances the field by delivering a structured, criteria-driven comparison of state-of-the-art blockchain authentication techniques. The methodology and scoring framework presented here can inform both academic research and industry practice, supporting the design and evaluation of authentication systems in a wide range of blockchain applications.

Practically, the findings underscore the importance of aligning authentication method selection with specific application requirements, user profiles, and threat models. Hybrid approaches—such as combining hardware wallets with multisignature, or integrating NFT-based credentials with account abstraction—may offer optimal security, usability, and future-proofing for mission-critical or rapidly evolving systems.

Several recommendations follow from this research:

- Critical or high-value applications should prioritize multi-factor or multi-party authentication, such as hardware-backed multisig setups.
- For persistent digital identity or membership management, SoulBound NFT approaches offer strong non-transferability and transparency, but must be augmented with privacy-preserving practices.
- Developers should monitor and adopt emerging standards like account abstraction, which offer improved user experiences without sacrificing security, and rigorously test and audit all smart contract logic.
- Where possible, leveraging open-source, community-audited frameworks and contracts, rather than reinventing core authentication components from scratch, can enhance security and maintainability.

Limitations of this study include the use of a testnet environment, a limited range of wallet and device combinations, and reliance on both empirical and literature-based scoring for qualitative criteria. Future research should expand benchmarking to mainnet environments, incorporate user studies, and explore additional security vectors—including

quantum-resistant cryptography, advanced privacy-preserving protocols, and biometric integrations.

In conclusion, as blockchain technology continues to mature and diversify, the demand for robust, user-friendly, and future-proof authentication systems will only grow. The insights and comparative framework established in this thesis are intended to serve as a guide for researchers, developers, and organizations seeking to implement secure and practical authentication mechanisms in decentralized systems. Ongoing adaptation to emerging standards, active security monitoring, and user-centric, innovation-driven design—including openness to quantum-resistant and biometric-enabled authentication—will be essential to meeting the evolving challenges of digital identity and access control in the blockchain era.

6. REFERENCES

- Corbató, F. J., Merrill, D. E., & Daggett, J. C. (1962). Compatible Time-Sharing System. *MIT Computation Center Internal Report*.
- Bonneau, J. (2012). The science of guessing: Analyzing an anonymized corpus of 70 million passwords. *IEEE Symposium on Security and Privacy (SP)*, 538–552.
- Florêncio, D., & Herley, C. (2007). A large-scale study of web password habits. *Proceedings of the 16th International Conference on World Wide Web*, 657–666.
- Verizon. (2023). Data Breach Investigations Report. *Verizon Enterprise Solutions*.
- Perlroth, N. (2017). Yahoo Says 3 Billion Accounts Were Hacked. *The New York Times*.
- Bonneau, J. (2015). Passwords and the evolution of imperfect authentication. *Communications of the ACM*, 58(7), 78–87.
- Adams, A., & Sasse, M. A. (1999). Users are not the enemy. *Communications of the ACM*, 42(12), 40–46.
- Ur, B., Segreti, S. M., Bauer, L., Christin, N., Cranor, L. F., & Komanduri, S. (2016). Users' perceptions of security and usability of authentication mechanisms. *Proceedings of the Symposium on Usable Privacy and Security (SOUPS)*.
- Bonneau, J., Herley, C., Van Oorschot, P. C., & Stajano, F. (2012). The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. *IEEE Symposium on Security and Privacy*, 553–567.
- LastPass. (2022). The Psychology of Passwords: LastPass Global Report. *LastPass Research Report*.
- Reese, K., Smith, T., Dutson, J., Armknecht, J., Cameron, J., & Seamons, K. (2019). A usability study of five two-factor authentication methods. In *Proceedings of the Fifteenth Symposium on Usable Privacy and Security (SOUPS 2019)*, USENIX Association, 357–370.

- Siadati, H., Saxena, N., & Zhu, Y. (2015). Mind your SMSes: Mitigating social engineering in second factor authentication. *International Conference on Financial Cryptography and Data Security*, 265–277.
- Jain, A. K., Ross, A., & Nandakumar, K. (2016). Introduction to Biometrics. *Springer*.
- Jain, A. K., Flynn, P., & Ross, A. A. (2008). *Handbook of Biometrics*. Springer.
- Ratha, N. K., Connell, J. H., & Bolle, R. M. (2001). Enhancing security and privacy in biometrics-based authentication systems. *IBM Systems Journal*, 40(3), 614–634.
- Preukschat, A., & Reed, D. (Eds.). (2021). *Self-Sovereign Identity: Decentralized Digital Identity and Verifiable Credentials*. Manning Publications.
- Tobin, A., & Reed, D. (2016). The Path to Self-Sovereign Identity. *Sovrin Foundation Technical Report*. <https://sovrin.org/wp-content/uploads/2017/06/ThePathToSelfSovereignIdentity.pdf>
- Zyskind, G., Nathan, O., & Pentland, A. S. (2015). Decentralizing Privacy: Using Blockchain to Protect Personal Data. In *2015 IEEE Security and Privacy Workshops* (pp. 180–184). <https://doi.org/10.1109/SPW.2015.27>
- Ferdous, S. M., Chowdhury, F., Hoque, M. A., & Colman, A. (2019). Blockchain-based self-sovereign identity: A systematic literature review. *Computer Science Review*, 35, 100204.
- Dunphy, P., & Petitcolas, F. A. P. (2018). A first look at identity management schemes on the blockchain. *IEEE Security & Privacy*, 16(4), 20–29.
- Zhao, J., Zhang, Y., Zhao, J., & Li, X. (2023). Decentralized identity systems: Architecture, challenges, and solutions. *Information Sciences*, 640, 119990.
- Gutoski, G., & Stebila, D. (2019). Hierarchical deterministic wallets and hardware wallets. In *Real World Crypto Conference*.
- Wang, Q., Li, R., Wang, Q., & Chen, S. (2022). Non-fungible token (NFT): Overview,

- evaluation, opportunities and challenges. *arXiv preprint arXiv:2105.07447*.
- Buterin, V. (2021). EIP-2938: Account Abstraction. *Ethereum Improvement Proposal*. Available at: <https://eips.ethereum.org/EIPS/eip-2938>
- Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. *White Paper*. Available at: <https://bitcoin.org/bitcoin.pdf>
- Buterin, V. (2014). Ethereum: A next-generation smart contract platform. *Ethereum White Paper*. Available at: <https://ethereum.org/en/whitepaper/>
- Mühle, A., Grüner, A., Gayvoronskaya, T., & Meinel, C. (2018). A survey on essential components of a self-sovereign identity. *Computer Science Review*, 30, 9–29.
- Tobin, A., & Reed, D. (2016). The Sovrin Network: Self-sovereign identity and decentralized trust. *Sovrin Foundation White Paper*. Available at: <https://sovrin.org/library/sovrin-protocol-and-token-white-paper/>
- Jacobovitz, O. (2016). Blockchain for identity management. *The University of Texas at Austin, Department of Computer Science White Paper*.
- Buterin, V. (2014). Multisignature wallets in Ethereum. *Ethereum Blog*. Available at: <https://blog.ethereum.org/2014/01/15/multisig/>
- Eskandari, S., Barrera, D., Stobert, E., & Clark, J. (2015). A first look at the usability of Bitcoin key management. In *Proceedings of the NDSS Workshop on Usable Security (USEC)*. <https://doi.org/10.14722/usec.2015.23015>
- Gnosis Safe. (2021). Multisig wallet security. Available at: <https://gnosis-safe.io/security/>
- Bonneau, J., Miller, A., Clark, J., Narayanan, A., Kroll, J. A., & Felten, E. W. (2015). SoK: Research perspectives and challenges for Bitcoin and cryptocurrencies. *IEEE Symposium on Security and Privacy*, 104–121.
- Zhao, X., & Si, Y.-W. (2022). NFTCert: NFT-based certificates with online payment

- gateway. *arXiv preprint arXiv:2202.09511*.
- Antonopoulos, A. M., & Wood, G. (2018). *Mastering Ethereum*. O'Reilly Media.
- Ethereum Foundation. (2023). EIP-4337: Account abstraction via entry point contract specification. Available at: <https://eips.ethereum.org/EIPS/eip-4337>
- Goldwasser, S., Micali, S., & Rackoff, C. (1989). The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1), 186–208.
- Lavin, R., Liu, X., Mohanty, H., Norman, L., Zaarour, G., & Krishnamachari, B. (2024). A survey on the applications of zero-knowledge proofs. *arXiv preprint arXiv:2408.00243*.
- Costan, V., & Devadas, S. (2016). Intel SGX explained. *Cryptology ePrint Archive*, Report 2016/086. Available at: <https://eprint.iacr.org/2016/086.pdf>
- Pinto, S., Oliveira, R., & Santos, N. (2019). Demystifying ARM TrustZone. *ACM Computing Surveys*, 51(6), 1–36.
- Marcel, S., Nixon, M. S., & Fierrez, J. (2014). *Handbook of Biometric Anti-Spoofing*. Springer.
- Li, X., Jiang, P., Chen, T., Luo, X., & Wen, Q. (2021). A survey on the security of blockchain systems. *IEEE Access*, 8, 174723–174741.
- Conti, M., Kumar, S., Lal, C., & Ruj, S. (2018). A survey on security and privacy issues of Bitcoin. *IEEE Communications Surveys & Tutorials*, 20(4), 3416–3452.
- Wood, G. (2014). Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Yellow Paper*. Available at: <https://ethereum.org/en/whitepaper/>

7. APPENDIX

7.1. Appendix A: Detailed Scoring Rubrics for Evaluation Criteria

7.1.1. Speed

Measured in milliseconds (ms). Speed scores in this comparison are assigned on a relative basis, ranking each authentication method from fastest (5) to slowest (1) based on the total measured authentication time (signing + on-chain verification and transaction confirmation). All evaluated methods require on-chain operations, so absolute latencies are dominated by blockchain transaction delays. Device signing time, multisig threshold coordination, and contract complexity also impact the final speed ranking.

7.1.2. Security

- **1:** Private key exposed; highly vulnerable to phishing, replay, or contract bugs
- **2:** Basic protection, but major weaknesses remain
- **3:** Standard EOA; moderate protection but phishing or contract risks exist
- **4:** Enhanced with hardware, multisig, or contract-level checks; rare vulnerabilities
- **5:** Maximum: hardware + multisig, non-transferable credentials, on-chain recovery

Sub-criteria: Key protection, MITM, replay, phishing, contract resistance

7.1.3. User Experience (UX)

- **1:** Cumbersome, many steps, frequent errors, confusing for users
- **2:** Complex process, onboarding difficult, moderate error risk
- **3:** Standard, some friction but manageable for most users
- **4:** Smooth, mostly intuitive, few steps and confirmations
- **5:** Seamless, one/two-click, clear feedback, low risk of user error

Sub-criteria: Number of steps, onboarding, error risk, clarity

7.1.4. Gas Cost

- **1:** Very high; significant cost for each authentication
- **2:** High; above average for on-chain actions
- **3:** Moderate; typical for EVM operations
- **4:** Low; efficient smart contract design
- **5:** Minimal; nearly gasless (e.g. with paymasters or pure off-chain verification)

Reported as both score and numeric gas value.

7.1.5. Complexity

- **1:** Highly complex; custom integration, multiple components, hard to maintain
- **2:** Complex; needs advanced integration or setup
- **3:** Moderate; some custom logic, but standard in most dApp contexts
- **4:** Simple; minimal steps or code, easily repeatable
- **5:** Minimal; plug-and-play, simple contracts, easy for all skill levels

Sub-criteria: Integration effort, codebase size, maintenance

7.1.6. Hardware Dependency

- **1:** Hardware always required for each operation
- **2:** Hardware needed for initial setup, but not every use
- **3:** Hardware optional for higher security
- **4:** Can use hardware, but not required
- **5:** No hardware required; pure software or web

7.1.7. Decentralization Level

- **1:** Fully centralized, single signer or entity controls access
- **2:** Centralized with limited redundancy or backup
- **3:** Somewhat decentralized; multi-entity or partial multisig
- **4:** Strongly decentralized; threshold multisig or distributed logic
- **5:** Fully decentralized, trust-minimized, no single point of failure

7.1.8. Scalability

- **1:** Not scalable, bottlenecks, fails with many users/requests
- **2:** Low scalability, significant friction at scale
- **3:** Moderate; works for typical dApp loads
- **4:** Good scalability, supports high demand, rare bottlenecks
- **5:** Massively scalable; proven in large-scale environments

7.1.9. Interoperability

- **1:** Works in very limited environments only
- **2:** Some compatibility, but many restrictions
- **3:** Standard; works with most EVM dApps/wallets
- **4:** High compatibility, only rare exceptions
- **5:** Universally compatible; any wallet, any dApp, EVM and beyond

7.1.10. Privacy

- **1:** Sensitive data exposed publicly, high linkability
- **2:** Some user data exposed or easily linked

- **3:** Standard; only minimal/account info on-chain
- **4:** Enhanced privacy; obfuscated or anonymized operations
- **5:** No unnecessary data exposure; privacy-preserving design

7.1.11. Future Proof

- **1:** Obsolete or vulnerable to known threats (e.g. quantum)
- **2:** Likely to become outdated; rigid design
- **3:** Standard; can be upgraded with effort
- **4:** Designed for extensibility, upgradeable
- **5:** Quantum-resistant, upgradeable, or forward-compatible with emerging standards

Scoring Rubrics for Evaluation Criteria

This appendix presents the detailed scoring rubrics for each evaluation criterion used in the thesis. Each criterion is scored from 1 (worst) to 5 (best), with clear definitions for each score.

Table 7.4. Summary of Scoring Rubrics for Authentication Method Evaluation

Criterion	Scoring (1-5)	Key Factors / Sub-Criteria
Speed	1: Highest, 5: Lowest	Authentication time
Security	1: Very poor, 5: Max	Key protection, replay, phishing, MITM, contract resistance
User Experience	1: Difficult, 5: Best	Steps, onboarding, error risk, clarity
Gas Cost	1: Highest, 5: Lowest	Gas used per authentication
Complexity	1: Complex, 5: Simple	Integration effort, codebase, maintenance
Hardware Dependency	1: Always, 5: Never	Device requirements
Decentralization	1: Centralized, 5: DeFi	Single point of failure, distributed approval
Scalability	1: Poor, 5: Excellent	Users, requests, throughput
Interoperability	1: Siloed, 5: Broadest	Wallet/dApp/chain compatibility
Privacy	1: Poor, 5: Excellent	Data exposure, linkability
Future Proof	1: Low, 5: PQ-ready	Upgradeability, quantum resistance

CURRICULUM VITAE

Hakan DURMAZ

EDUCATION

Master's Degree 2022-2025	Akdeniz University Institute Of Natural And Applied Sciences Department Of Computer Engineering
Bachelor's Degree 2010-2014	Pamukkale University Faculty of Engineering Department of Computer Engineering

WORK EXPERIENCE

Senior Software Engineer Jan 2022 - Present	Self Employed
Senior Software Engineer Aug 2021 - Dec 2021	Anadolu Sigorta
Senior Software Engineer Sep 2016 - Aug 2021	Pribas Turkey Ltd
Software Engineer Jul 2014 - Aug 2016	Hotelchk
Software Engineer Aug 2012 - Jul 2014	Ratel Bilisim Ltd
Intern Software Engineer Jun 2012 - Aug 2012	Pamukkale University IT Department