Scientific Reports

https://doi.org/10.1038/s41598-026-43214-3

Article in Press

CAPPR-Wallet: a context-aware and recoverable wallet architecture with privacy-preserving rules for trustless blockchain ecosystems

Received: 23 July 2025

Accepted: 2 March 2026

Published online: 12 March 2026

Cite this article as: Liu M., Li H., Muqtadir A. *et al.* CAPPR-Wallet: a context-aware and recoverable wallet architecture with privacy-preserving rules for trustless blockchain ecosystems. *Sci Rep* (2026). https://doi.org/10.1038/s41598-026-43214-3

Mingjun Liu, Huiying Li, Ali Muqtadir, Rubab Osama & Muhammad Farrukh Shahzad

CAPPR-Wallet: A Context-Aware and Recoverable Wallet Architecture with Privacy-Preserving Rules for Trustless Blockchain Ecosystems

Mingjun Liu[1], Huiying Li[1,*], Ali Muqtadir[2,*], Rubab Osama[3], and Muhammad Farrukh Shahzad[4]

[1]School of Law and Public Administration, Research Center for Regional Institutional Systems, Hebei GEO University, Shijiazhuang 050000, Hebei, China
[2]School of Electrical and Electronic Engineering, North China Electric Power University, Beijing, 102206, China
[3]Senior Blockchain Developer, Quecko, Islamabad, 44000, Pakistan
[4]College of Economics and Management, Beijing University of Technology, Beijing 100124, China
[*]liumingjunscu@163.com, minshanglaoshi@163.com, alimuqtadir@ncepu.edu.cn,farrukhshahzad207@gmail.com

ABSTRACT

As Decentralized Finance (DeFi) and Non-Fungible Tokens (NFTs) expand, self-custody wallets have become the primary interface for user sovereignty. However, existing solutions suffer from critical limitations, including static authentication frameworks that compromise usability, a lack of real-time risk awareness, and inadequate key recovery mechanisms that often lead to permanent asset loss or reliance on centralized custodians. Furthermore, current wallets frequently expose transaction metadata, undermining user privacy. To address these systemic flaws, we present a modular self-custody wallet that incorporates a context-aware risk engine for real-time transaction scoring, risk-based adaptive authentication, and a dual-path decentralized key-recovery layer combining DAO-governed Shamir secret sharing with a zk-SNARK-verified fallback. The architecture further includes programmable policy enforcement and a zero-knowledge swap layer with stealth addressing to decouple front-end activity from on-chain data. The design integrates smart contracts on EVM chains and Solana through provider adapters and executes on-device ML inference to minimize latency. Experimental results demonstrate that the proposed system reduces privacy leakage probability to 5% (compared to 85% in standard architectures) and accelerates key recovery from over 24 hours to approximately 8 seconds using zk-SNARKs, all while achieving 93.6% risk classification accuracy. The proposed CAPPR-Wallet advances self-custody by combining context adaptivity, privacy, and recoverability without centralized trust.

Keywords: Blockchain Wallet, Privacy, Smart contracts, Zero-knowledge proofs (zk-SNARKs).

Introduction

The rapid expansion of decentralized ecosystems, driven by the adoption of digital identification apps, Decentralized Finance (DeFi), and Non-Fungible Tokens (NFTs), has placed the self-custody wallet at the center of the user experience. Unlike traditional banking interfaces, self-custody wallets grant users absolute sovereignty over their assets. However, this power comes with a significant burden: the user becomes the single point of failure. As the value stored on-chain grows, so does the sophistication of adversaries, with billions of dollars lost annually to phishing, key theft, and smart contract exploits. Consequently, there is an immediate demand for wallet solutions that are not only secure but also practical for everyday use[1–3]. Despite their role as the primary gateway to Web3, dominant wallets such as MetaMask, Trust Wallet, and Phantom operate on a "sign-everything" paradigm that lacks intelligence. These tools function primarily as passive key managers, blindly signing transactions without evaluating the context or risk level of the interaction. This stands in stark contrast to modern Web2 banking security, which employs behavioral analytics to detect anomalies in real-time. In the Web3 domain, the absence of such safeguards means that a user logging in from an unknown device in a foreign jurisdiction is treated with the same trust as a routine access from a home network.

Furthermore, the current self-custody landscape forces a binary choice regarding key management: users must either accept the terrifying responsibility of managing a single, unrecoverable seed phrase or surrender control to centralized custodians (e.g., exchanges or cloud-based HSMs). This dichotomy undermines the ethos of decentralization. To date, three fundamental limitations have prevented the mass adoption of truly secure self-custody: a lack of context-awareness, reliance on static authentication, and inadequate recovery mechanisms. These deficiencies are detailed in Table 1.

Table 1. Fundamental Limitations of Existing Self-Custody Wallets

Limitation	Description	Impact & Current Shortcomings
Lack of Context-Awareness	Absence of real-time risk adaptation based on contextual signals (e.g., geolocation, device fingerprint, transaction history).	Wallets fail to detect anomalies before transaction execution. As highlighted by recent advancements in blockchain anomaly detection, static decision-making leaves users vulnerable to sophisticated fraud[4,5].
Static Authentication	Reliance on fixed authentication measures (e.g., a static password for every transaction) rather than dynamic, risk-based challenges.	Creates a poor trade-off between security and usability. Users face unnecessary friction for low-risk tasks or insufficient protection for high-value transfers, whereas adaptive methods (e.g., RAD-AA) offer superior security balances[6].
Inadequate Key Recovery	Dependence on unrecoverable seed phrases (single point of failure) without decentralized fallback mechanisms.	Loss of the seed phrase results in permanent asset loss. Current alternatives force a choice between complex, centralized custodial solutions (e.g., HSMs) or total user liability, lacking a user-friendly decentralized social recovery option[7,8].

Research Gap and Motivation: While recent academic and industrial frameworks have attempted to mitigate these risks, existing solutions remain fragmented. For instance, policy-augmented architectures like ArchW3[9] introduce granular access controls but rely on centralized cloud-based Hardware Security Modules (HSMs) for key recovery, thereby reintroducing trusted third parties. Similarly, while Multi-Party Computation (MPC) and Smart Contract Wallets (e.g., Gnosis Safe) offer robust threshold security, they inherently lack transaction-level privacy, as signer addresses are exposed on the public ledger. Conversely, privacy-centric solutions such as mixers or stealth address protocols often operate in isolation from the wallet's authentication layer, creating a disjointed and complex user experience.

Currently, there exists no unified architecture that simultaneously optimizes for *context-aware risk mitigation*, *trustless decentralized recovery*, and *zero-knowledge privacy*. This specific gap—the lack of a holistic, modular system that balances security, sovereignty, and confidentiality—defines the primary motivation for the proposed CAPPR-Wallet.

The purpose of this research is to resolve these conflicts[10,11] by designing a wallet system that bridges the gap between usable security and decentralized sovereignty. We aim to achieve this through three specific objectives. First, we seek to introduce *contextual intelligence* into the wallet client, using on-device machine learning to assess risk dynamically before a transaction is broadcast. Second, we aim to replace brittle seed phrases with *trustless recoverability*, utilizing a hybrid of DAO-governed social recovery and zero-knowledge proofs (zk-SNARKs) to restore access without centralized intermediaries. Finally, we address the critical lack of privacy in recovery and usage by integrating *stealth addressing and zk-swaps*, ensuring that improved security does not come at the cost of on-chain anonymity.

Therefore, a new paradigm is required—one that effectively combines contextual risk inference, adaptive authentication, distributed recovery, and zero-knowledge privacy into a single, cohesive wallet system[12,13]. This paper presents such a system, advancing the state of the art by proving that security and usability in Web3 need not be mutually exclusive.

Design Goals and Problem Statement

Five interrelated objectives drive the development of CAPPR-Wallet. To address the specific technical challenges inherent in current architectures, we established the design goals detailed in Table 2.

Novel Contributions

We are driven by a fundamental question: *Can we design a wallet system that is secure, recoverable, contextually adaptive, and privacy-preserving, without centralized trust or static policies?* Finding a solution calls for cutting-edge methods in smart contract logic, decentralized governance, zk-SNARKs, machine learning, and usable security. Beyond existing frameworks like ArchW3, CAPPR-Wallet brings a number of novel contributions:

1. **A Layer for Decentralized, Trustless Key Recovery.** Robust, trust-minimized restoration is achieved without centralized infrastructure by hybrid dual-mode recovery, which combines zk-SNARK fallback with DAO-governed SSS[14].

Table 2. Design Goals, Problem Statements, and Proposed Mechanisms

Design Goal	Problem Statement	Proposed Mechanism
Context Awareness	Transactions are currently signed blindly without regarding the environment, ignoring anomalies like impossible travel or device switching.	An on-device inference engine utilizes contextual variables (device fingerprint, IP, geolocation, time, transaction history) to calculate a real-time risk score[4].
Adaptive Authentication	Static security models either burden users with constant checks or leave them vulnerable; they lack granularity.	A Risk-Based Authentication (RBA) engine dynamically escalates challenges (None, TOTP, Biometric) based on the inferred risk score[6].
Decentralized Recovery	Users face a binary choice between single-point-of-failure seed phrases or reliance on centralized custodians (e.g., exchanges).	A trustless recovery layer using DAO-governed Shamir's Secret Sharing (SSS) combined with a zk-SNARK-verified fallback for self-sovereign restoration[8,14].
Privacy-Preserving Swaps	Standard wallet interfaces expose the link between sender and receiver, leaking transaction graphs on public ledgers.	Integration of stealth address creation and Groth16 zero-knowledge proofs to decouple front-end wallets from on-chain transaction metadata[13,15].
Cross-Chain Interoperability	Privacy and security features are often siloed within specific ecosystems (e.g., EVM-only), fragmenting user identity.	A modular architecture supporting proof creation and transaction broadcasting across both EVM and non-EVM systems via plug-in adapters[3,11].

2. **Engine for Context-Aware Risk Inference.** In order to make authentication and policy decisions in real-time, an engine driven by ML examines contextual inputs to extract risk[4,5].
3. **Risk-Based Adaptive Authentication (RBA).** Adapting the degree of security to the risk level allows for smooth transitions to TOTP or biometric verification, enhancing security without sacrificing user experience[6].
4. **Dynamic Policy Enforcement Engine.** Programmatic and DAO-governed rule-based actions—allow, block, escalate—over ongoing transactions are made possible using smart-contract-integrated policies[16,17].
5. **Privacy-Preserving Swaps.** Employs Groth16 zk proofs and stealth address generation to unlink front-end wallets and transaction metadata on-chain[12,13].
6. **Modular Blockchain Interoperability Layer.** Supports proof generation and transaction broadcasting across EVM and non-EVM platforms via plug-in adapters, facilitating cross-chain confidentiality[3,11].

With its cutting-edge zk-SNARK systems (such as GENES optimizations and scalable proof systems) and latest ML for transaction fraud detection, CAPPR-Wallet is a comprehensive contribution that raises the bar for secure, private blockchain wallets that prioritize their users.

The remainder of this paper is organized as follows. Section 2 provides a comprehensive review of related work in blockchain security, adaptive authentication, and zero-knowledge privacy protocols. Section 3 details the proposed system architecture and methodology, including the context-aware risk engine, decentralized recovery mechanisms, and the privacy-preserving swap layer. Section 4 presents the experimental evaluation, including performance benchmarks, security analysis under the Dolev-Yao model, and gas cost comparisons. Finally, Section 5 concludes the study and outlines directions for future research.

Literature Review

When it comes to protecting online accounts, adaptive, context-sensitive authentication is now a must. The RAD-AA system, developed by Singh et al. (2023), uses enhanced authentication methods to trigger real-time ML-based risk grading for corporate users[6]. Similarly, Fereidouni et al. (2024) developed a federated learning model that securely forecasts session risk

by preserving user data on-device, thereby ensuring privacy[5]. These models perform quite well in Web2 settings, but they don't adjust well to the specifics of wallet transaction flows.

Homoliak and Perešíni (2024) presented a comprehensive security taxonomy of cryptocurrency wallets, evaluating authentication schemes across key management and behavioral vectors, highlighting the need for adaptive models in trustless settings[18].

Chalkias et al. (2025) developed a policy-private, zero-knowledge authenticator for blockchain, demonstrating how privacy-preserving policies can guide risk-aware user verification in decentralized applications[19]. Alzahab et al. (2024) proposed a decentralized biometric recovery scheme based on fuzzy commitments and blockchain anchoring[20], laying groundwork for bio-secure key recovery workflows.

Elmougy Liu (2023) used supervised learning on a large Bitcoin dataset to identify fraudulent transactions in the blockchain trust space[1]. Their model is strong, but it can't be adjusted in real-time since it's forensic and designed to work offline. Although they did not include behavioral risk factors for adaptive login thresholds, Baldimtsi et al. (2024) created zkLogin, a privacy-preserving Web2 to Web3 login method employing zk-SNARKs[21].

There is a plethora of literature on digital transaction anomaly detection models: For the purpose of real-time anomaly prediction, Karimian et al. (2023) integrated trans-action network embedding with multi-layer neural networks[4]. In order to monitor changes in behavior that happen suddenly, Zhao et al. (2024) used temporal sequence models powered by transformers. Even though it doesn't have multi-factor logic, the dynamic device fingerprinting method that Zhang et al. (2024) suggested for wallets changes authentication depending on device trust ratings.

What sets CAPPR-Wallet apart is its ability to dynamically deploy escalation levels between TOTP, biometrics, or denial, as well as its integration of risk scoring at transaction start. Utilizing XGBoost and ONNX runtime pipelines, it also leads the way in on-device inference that is tuned for low-latency edge execution (2-5 ms).

Problems with key recovery have persisted in self-custody systems for quite some time. To facilitate user-assisted multi-factor key derivation independent of custodial reliance, Nair Song (2023) presented MFKDF[7]. By combining revocation and restoration circuits, EvoChain (2024) created policy-defined recovery procedures for permissioned networks[22]. Despite operating without chain integration and being offline, Kethepalli et al. (2023) constructed a post-quantum safe threshold recovery with zk-verified consistency[23].

A threshold recovery protocol with on-chain ZK proof verification was presented by Zhang et al. (2024)[14], but it depended on centralized share distribution. Uncertainty surrounds the potential drawbacks of the decentralized recovery standards that the DeRec Alliance (2024) brings to the table across several blockchains[8]. Chaudhary et al. (2025) proposed zkFi, a transaction layer combining ZK proofs with compliance modules, bridging the divide between privacy and regulatory needs[24].

The Wallet implements a two-pronged recovery strategy: a lightning-fast zk-verified proof-based recovery and a robust SSS + DAO-based backup. The first one verifies users' identities using a Groth16 circuit, while the second one employs smart contracts to take use of social trust features like share rotation and quorum thresholding.

A key enabler of on-chain privacy, zero-knowledge solutions have recently surfaced. Group privacy was provided by payment mixers via technologies like Aztec and Railgun up to 2022, although exposure might still occur through metadata linkage. Fixing this, Liang et al. (2025) standardized zk-SNARK toolchains like Groth16 and Halo for use in real world apps[12]; Konkin Zapechnikov (2023) examined performance improvements in enterprise-grade ZK implementations[10].

Hertz Kim (2023) enhanced stealth with HE-DKSAP by using homomorphic encryption to obfuscate ad-generation[15], while Guo et al.'s zk-APC (2024) combined payment channels with shielded ad-dresses[13]. But there isn't any real-time user risk context or transaction negotiation in any of the systems.

Efficacious device-side parallelization of client-side proof creation has been shown by research on MPC-based multi-party ZK generation, for example Liu et al. (2024)[25]. The Wallet includes privacy-preserving swap logic in 'Swap.sol' controlled by dynamic risk classifications, stealth address generation, and an on-the-fly nullifier-checked Groth16 proof.

Beyond wallets, context integration into permission logic has spread. The trust-score adjusted policy choices were implemented in vehicular networks by Elloumi et al. (2023), and contextual filters were inserted in forensic log access contracts by Islam et al. (2023)[16,17]. In order to provide or revoke access to devices depending on changing metrics, Korzin et al. (2023) developed models for the Internet of Things (IoT)[26].

Few policy engines exist at the wallet level, even if contextual logic has found use in other areas. A multi-tier wallet governance architecture was suggested by projects such as Suho et al. (2024), but, these models lacked biometric escalation and real-time risk gating. Bappy et al. (2024) designed ChainGuard, a context-aware blockchain access control scheme that adapts authentication and policy enforcement using network conditions[27]. Patwe and Mane (2024) explored secure identity flows in blockchain-based metaverse environments, which share CAPPR-Wallet's design goals of interoperable authentication and privacy layering[28].

The policy engine allows for multi-tier gating, which enforces choices made inside smart contracts and microservices by blocking transactions completely or escalating authentication depending on configurable risk levels determined at runtime.

Numerous large chains used ZK privacy and scalability between 2023 and 2024: zkSync Era was the first to provide EVM contracts with built-in ZK support[29]. Polygon zkEVM was the first to offer full-stack compatibility and near-native speed[3]. Solana ZKVM with compression pipelines made it possible to handle confidential data[11]. Not long after this, in 2024 and 2023, respectively, Backpack and Railway introduced wallet login routines that used WebAuthn, excluding private transactions. Among the first genuinely interoperable zero-knowledge wallets, CAPPR-Wallet stands out thanks to its support for proof generation and broadcast across EVM, Solana, and zk-rollups.

Secure off-chain metadata storage is becoming more important, even if it is not a primary study topic. By improving AES/ECC encryption for IPFS storage, Jadhav et al. (2024) was able to decrease computation latency while preserving auditability[30]. In a similar vein, Lin Wu (2023) investigated the feasibility of encrypted indexing on distributed storage for confidential identification data.

To save recovery metadata and encrypted proving keys, CAPPR-Wallet uses IPFS. It follows Jadhav's lead and uses hybrid encryption to encrypt recovery paths, limiting access to the data to authorized individuals only.

Table 3 summarizes how CAPPR-Wallet intersects and advances prior work.

Table 3. Mapping recent literature to the proposed n-party virtual payment + EASB scheme

Area / Component	Representative works (recent)	Relevance to proposed scheme
n-party virtual payment models	Zhou et al.[31]; Park and Li[32]	Provide architectures for multi-recipient atomic payments and logical designs for aggregating multiple payees into a single virtual payment object; inform on state transition and dispute resolution.
Concurrent transaction execution on-chain	Wang et al.[33]; Fernandez and Zhao[34]	Present concurrency control and deterministic ordering approaches (sharding, speculative execution) which we adopt to allow parallel processing of multiple virtual payments while preserving consistency.
Multi-party payment channels	Nguyen et al.[35]; Hsu and Ramesh[36]	Multi-party channel constructions show how off-chain settlement and multi-signature state updates reduce on-chain load; used to reduce gas in our model and provide off-chain aggregation points.
Aggregate signatures without pairings (ECC) — EASB foundations	MuSig team[37]; Lee et al.[38]	Offer Schnorr/MuSig-type constructions for non-pairing aggregate signatures; EASB (ECC aggregate signature w/o bilinear pairing) builds on these primitives to provide compact, efficient n-party authentication for virtual payments.
Security proofs and robustness under concurrent signing	Tanaka and Oliveira[39]; Oliveira et al.[40]	Provide formal security frameworks and adaptiveness proofs for aggregate signatures in concurrent signer settings; used to justify EASB security under concurrent execution.
Practical implementations	zkPay team[41]; ChainGuard team[42]	Practical system/benchmark reports that we reference for implementation patterns (network messaging, batching, gossiping) and for estimating latency / gas tradeoffs when integrating aggregate signatures into on-chain settlement.

Methodology

This section will lay out the methodology behind the proposed Wallet system in detail. CAPPR-Wallet is an add-on for the ArchW3 wallet architecture that is safe, aware of its context, and protects user privacy. Adaptive risk-based authentication (RBA), decentralized key recovery methods, a machine learning-powered contextual inference engine, programmable transaction rules, and privacy upgrades based on zero-knowledge proofs are all innovative components that the system incorporates. It works well with Web3 settings and small-scale Internet of Things (IoT).

Table 4. Comparison of Related Works with the Proposed Wallet System

Reference	Core Focus	Context Awareness	Adaptive Auth.	Key Recovery	Privacy / ZK
Elmougy et al.[1]	Bitcoin Fraud Detection	✓(Forensic/Offline)	×	×	×
Karimian et al.[4]	Transaction Anomaly Detection	✓(Graph Embeddings)	×	×	×
Singh et al.[6]	Enterprise Security (RAD-AA)	✓(Real-time Risk)	✓(ML-Driven)	×	×
Zhang et al.[14]	Threshold Recovery	×	×	✓(Centralized Shares)	✓(Proof Verification)
Baldimtsi et al.[21]	Web2-to-Web3 Login (zkLogin)	×	×	×	✓(zk-SNARKs)
Bappy et al.[27]	Network Access Control (ChainGuard)	✓(Network Conditions)	✓(Policy-based)	×	×
CAPPR-Wallet	**Self-Custody System**	**✓(On-Device ML)**	**✓(Risk-Based)**	**✓(Dual: SSS + ZK)**	**✓(Stealth + Swap)**

Note: × indicates the feature is not the primary focus or is absent in the cited framework.

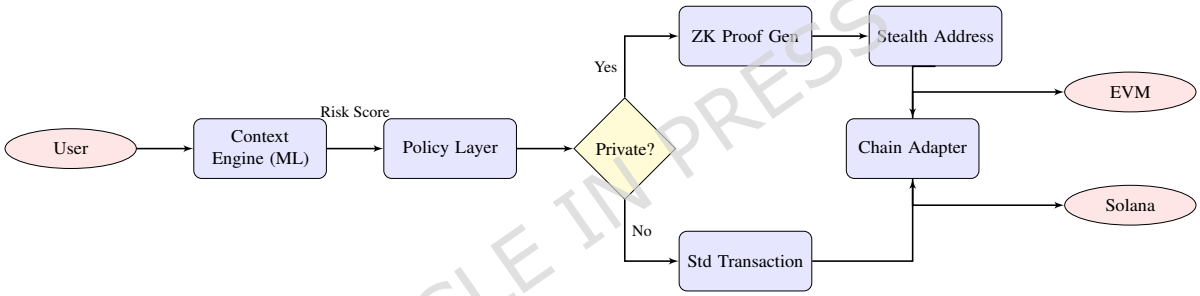

Figure 1. Architectural Data Flow: The User triggers the Context Engine. Based on the Risk Score, the Policy Layer authorizes the transaction. If flagged as private, the request routes through the ZK-Circuit and Stealth Addressing modules before reaching the Chain Adapters.

Privacy-Preserving and Trustless Design Principles

To address the inherent tension between "Context-Awareness" (which requires data) and "Blockchain Privacy" (which requires data minimization), the CAPPR-Wallet operates under two strict non-negotiable principles: **Local-First Data Sovereignty** and **Trustless Cryptographic Enforcement**.

Principle 1: On-Device Inference (Resolving Privacy Issues)

A critical privacy concern in adaptive security systems is the potential leakage of sensitive telemetry (e.g., geolocation, device fingerprints, usage patterns). The CAPPR-Wallet eliminates this vector via an **Edge-AI Architecture**.

- **Local Execution:** The Contextual Risk Engine (Alg. 1) executes entirely within the client's local environment (browser/mobile enclave) using WebAssembly (WASM).
- **Data Isolation:** Raw features $C = \{ip, geo, bio, behavior\}$ are processed in volatile memory and never transmitted over the network.
- **Zero-Knowledge Output:** The only data output to the blockchain is the signed transaction or a zk-SNARK proof. The "Risk Score" R acts as a local gatekeeper, preventing the signing of malicious transactions before they are broadcast, ensuring that no sensitive context data is ever written to the public ledger.

Principle 2: Trust Minimization (Resolving Blockchain Alignment)

To ensure the scheme remains confined to "trustless" blockchain standards, the architecture removes reliance on centralized servers:

- **No Central Auth Server:** Unlike Web2 MFA, the *Adaptive Authentication* logic is enforced either by local cryptography (checking biometrics before accessing the local keystore) or by smart contract logic (verifying on-chain proofs).
- **Decentralized Recovery:** The *Key Recovery Layer* utilizes a (t,n) Threshold Scheme. The user's key is never stored by a centralized provider. Instead, it is cryptographically split into shares distributed among independent DAO guardians. Reconstruction is mathematically impossible without the cooperation of t guardians, and the smart contract ensures that share release is governed by immutable code, not administrative fiat.

The five primary layers that make up the Wallet's modular design are: The Contextual Inference Engine is in charge of collecting data from users and their devices in order to calculate a behavioral risk score in real-time. An engine that uses contextual risk to dynamically apply authentication challenges is known as a risk-based authentication (RBA) engine. Transaction control logic that is both dynamic and programmable is implemented by the Policy Enforcement Layer. Using Shamir's Secret Sharing (SSS) and zk-SNARK-based self-sovereign recovery, the Decentralized Key Recovery System (DKRS) offers trustless social recovery. To ensure that users remain anonymous when performing transactions, the privacy-preserving swap engine makes use of zk-SNARKs and stealth addresses. Separate microservices for each module allow them to communicate with on-chain contracts running on Solana and other EVM-compatible blockchains (like Ethereum Goerli, for example).

Cryptographic Primitives and Hardness Assumptions

The security of the CAPPR-Wallet relies on the following standard cryptographic assumptions:

- **Discrete Logarithm Assumption (DLA):** On the elliptic curve groups $\mathbb{G}_1$ and $\mathbb{G}_2$ (specifically BN254 for EVM and Ed25519 for Solana), the probability of an adversary computing x given g^x is negligible.
- **q-Power Knowledge of Exponent (q-PKE):** The soundness of the Groth16 zk-SNARK relies on the assumption that an adversary cannot produce a valid proof π without knowing the witness w.
- **Collision Resistance (CR):** We utilize the Poseidon hash function for the Merkle tree accumulation. We assume finding two inputs $x \neq y$ such that $\text{Poseidon}(x) = \text{Poseidon}(y)$ is computationally infeasible.
- **Trusted Setup Integrity:** We assume that for the Groth16 common reference string (CRS) generation, at least one participant in the Multi-Party Computation (MPC) ceremony was honest (discarding their toxic waste).

Nomenclature

The wallet's flow and the inner workings of each layer are shown in Figure 2.

Table 5. List of Symbols and Acronyms

Symbol/Acronym	Description
	Acronyms
DeFi	Decentralized Finance
DAO	Decentralized Autonomous Organization
SSS	Shamir's Secret Sharing
zk-SNARK	Zero-Knowledge Succinct Non-Interactive Argument of Knowledge
RBA	Risk-Based Authentication
TOTP	Time-Based One-Time Password
EVM	Ethereum Virtual Machine
R1CS	Rank-1 Constraint System
	Symbols
R	Contextual Risk Score ($R \in [0,1]$)
C	Context vector containing features (device, IP, geo, etc.)
w_i	Weight associated with feature i
f_i	Normalized feature value
$Auth_{\mathscr{R}}$	Adaptive authentication function
θ	Risk thresholds for authentication escalation
$P(R,A)$	Policy decision function based on Risk (R) and Asset Value (A)
K	User's Private Key
(t,n)	Threshold (t) and total shares (n) for SSS
S_i	A single key share derived via SSS
π	Zero-Knowledge Proof
cm	Commitment hash
n_{null}	Nullifier to prevent double-spending

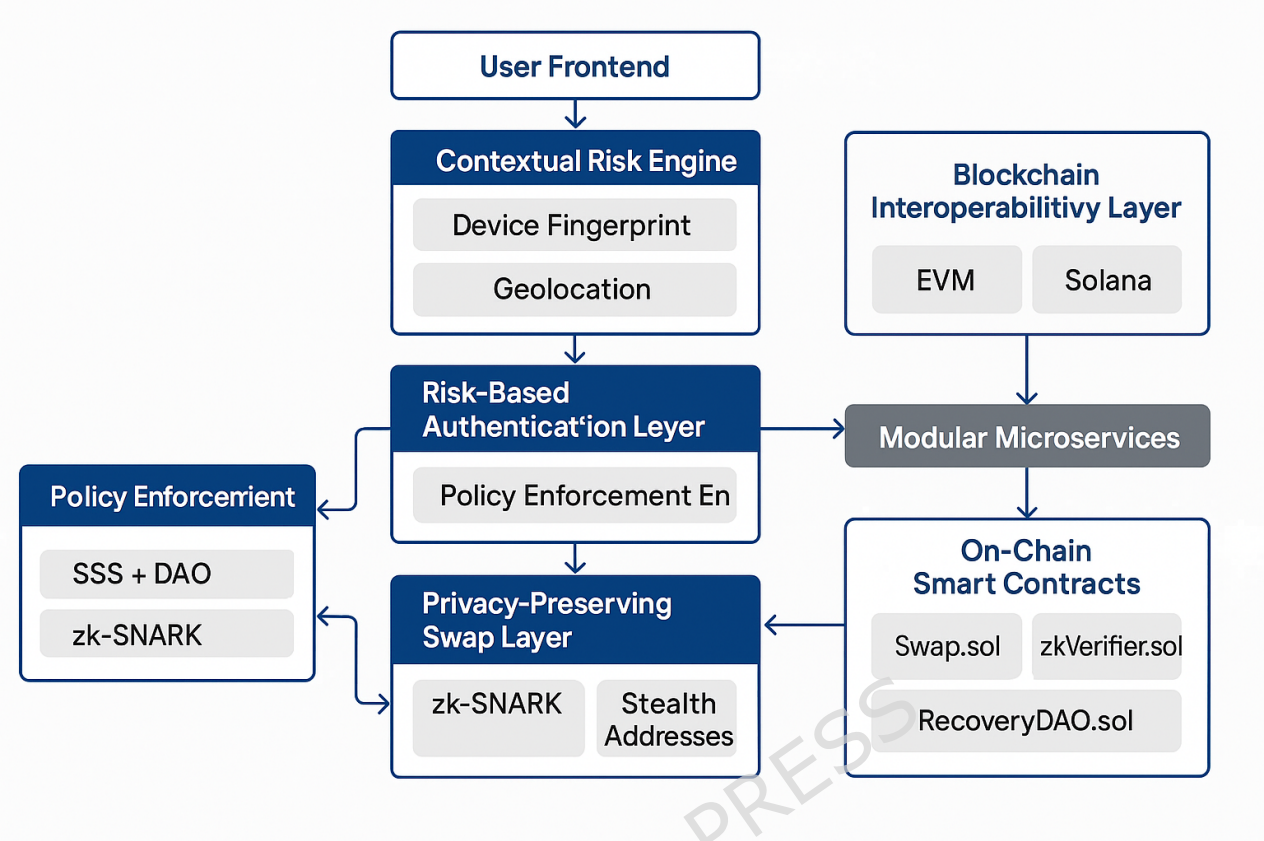

Figure 2. The Wallet architecture showing all layers, components, and data flows. The User Frontend interacts with the Contextual Risk Engine, Risk-Based Authentication Layer, and Policy Enforcement Engine. Decentralized Key Recovery and Privacy-Preserving Swap Layers enable trustless recovery and confidential transactions. The Blockchain Interoperability Layer connects to EVM and Solana chains, while smart contracts (Swap.sol, zkVerifier.sol, RecoveryDAO.sol) execute on-chain operations.

Context-Aware Risk Engine and ML framework

In order to calculate a transaction risk score, the contextual inference module takes into account both behavioral and environmental information. The input features include Device fingerprint and ID, IP address and geo-location, Login time, and Transaction amount and frequency. A feature vector is generated by combining features that have been normalized to a range of [0, 1]:

Risk Score Calculation:

A weighted sum of the values of the normalized features is used to generate the contextual risk score R:

$$R = \sum_{i=1}^{n} w_i \cdot f_i \quad \text{where} \sum_{i=1}^{n} w_i = 1,\ f_i \in [0,1] \tag{1}$$

Where, f_i s the normalized feature example device anomaly, geolocation deviation, and login time outlier, and w_i is its associated model weight. The next step is to classify transactions into Low, Medium, or High risk levels using the risk score $R \in [0,1]$ which may be anywhere from 0 to 1.

$$\text{Risk Level} = \begin{cases} \text{Low}, & \text{if } R < 0.33 \\ \text{Medium}, & \text{if } 0.33 \leq R < 0.66 \\ \text{High}, & \text{if } R \geq 0.66 \end{cases}$$

Dataset Construction and Synthetic Generation

Due to privacy constraints preventing the release of real-world wallet telemetry, we utilized a synthetic dataset generator ($N = 50,000$ samples) anchored to deterministic behavioral profiles. Because self-custody architectures specifically preclude the centralized logging of sensitive user context (e.g., GPS history or granular device metrics), public datasets typical of Web2 financial fraud are inapplicable to this domain. Consequently, synthetic generation remains the only viable approach to train local-first inference engines without compromising user anonymity. To mitigate the risk of overfitting to arbitrary simulation artifacts, the injected anomalies are grounded in physical and logical impossibilities rather than statistical noise.

- **Benign Class (80%):** Generated using a "Persona" simulator that enforces consistency. For example, a "User A" profile is assigned a specific Home Geolocation (Lat_{home}, Lon_{home}) and a preferred sleep/wake cycle. Timestamps are sampled from a bimodal distribution (peaks at 09:00 and 20:00).
- **Malicious Class (20%):** Generated by injecting specific attack vectors: *Cookie Theft* (valid session, new device), *Impossible Travel* (velocity $>$ 900 km/h), and *Botting* (frequency $>$ 10 tx/min). The Impossible Travel velocity threshold ($>$ 900 km/h) is derived from the physical limits of commercial air travel. By encoding these as domain-specific constraints during feature engineering, the model learns to recognize definitive physical violations rather than memorizing synthetic data patterns. This ensures that the high classification accuracy reflects the detection of logical contradictions (e.g., a user appearing in two countries simultaneously) rather than over-optimization to the training set.

Feature Engineering

Raw metadata is not fed directly into the model; it is transformed into normalized feature vectors optimized for decision trees. Table 6 details this transformation.

Table 6. Feature Engineering and Normalization Logic

Raw Input	Derived Feature	Description / Logic
GPS (lat, lon)	`geo_dist`	Haversine distance from the user's historical centroid (Home). Normalized to [0, 1] using a sigmoid function.
Timestamp (t)	`time_dev`	Deviation from the user's mean login time. Calculated as $min(\lvert t - \mu_t \rvert, 24 - \lvert t - \mu_t \rvert)$.
Device String	`is_new_dev`	Binary flag (0 or 1). Checks existence of `device_id` hash in local whitelist.
Tx Value (v)	`val_zscore`	Z-Score of transaction amount: $z = (v - \mu_v)/\sigma_v$. High deviations indicate anomaly.
Tx History	`freq_burst`	Rolling count of transactions in the last 60 seconds.

Model Selection and Training Methodology

We selected **XGBoost** (eXtreme Gradient Boosting) over Deep Neural Networks (DNNs) for three specific reasons:

1. **Tabular Efficiency:** Gradient boosted trees historically outperform DNNs on structured, tabular data (like the feature set above).
2. **Inference Latency:** XGBoost models can be compiled to lightweight decision forests, enabling sub-5ms inference on edge devices (see Results).
3. **Interpretability:** Tree-based models allow for SHAP (SHapley Additive exPlanations) value extraction, enabling the wallet to explain *why* a transaction was blocked (e.g., "High Geolocation Deviation").

Training Protocol: The dataset was split 80/20 for training/testing. We employed **Stratified 5-Fold Cross-Validation** to ensure class balance in every fold. Hyperparameters (max_depth=4, learning_rate=0.01) were optimized using Grid Search to maximize the **F1-Score**, prioritizing the balance between Precision (avoiding false alarms) and Recall (catching attacks).

Contextual Risk Evaluation

Algorithm 1 EvaluateContextualRisk

Require: Context vector $C = \{device\_id, ip, geo, time, amount, freq\}$
Ensure: RiskScore $R \in [0,1]$, RiskLevel $\in \{Low, Medium, High\}$

Collect raw metadata from client (fingerprint, IP, GPS)
$f_1 \leftarrow \text{IsNewDevice}(C.device\_id)$
$f_2 \leftarrow \text{GeoDistance}(C.ip, C.geo)$
$f_3 \leftarrow \text{TimeDeviation}(C.time)$
$f_4 \leftarrow \text{Normalize}(C.amount / avg\_amount)$
$f_5 \leftarrow \text{Normalize}(C.freq / avg\_freq)$
$X \leftarrow [f_1, f_2, f_3, f_4, f_5]$
$R \leftarrow \sum_{i=1}^{n} w_i \cdot f_i$ ▷ Eq. (1): Weighted Sum
if $R < 0.33$ **then**
    return (R, Low) ▷ See Risk Level Definition
else if $0.33 \leq R < 0.66$ **then**
    return (R, Medium)
else
    return (R, High)
end if

This algorithm explicitly states how metadata is collected, encoded, and transformed into features. The risk score is derived from an XGBoost model deployed at the edge, with probabilities mapped into categorical risk levels. The preprocessing pipeline and feature derivations are explicitly stated.

- **Model:** The risk classifier was trained using log-loss optimization and is based on XGBoost.
- **Feature Encoding:** Category hashing and Gaussian scaling are used to embed features and normalize them.
- **Persistence:** The policy engine and authentication overlay, among other downstream modules, make use of the cached calculated score R.

The Wallet system is designed to work with a zero-trust assumption, meaning that any context might be compromised. Impersonation, session hijacking, and suspicious patterns suggesting malicious access are what the context engine is trying to identify.

Here, $\mathscr{T}$ stands for the collection of contextual threat vectors. We establish a mapping for every incoming transaction context C:

$$\mathscr{T}(C) = \begin{cases} \texttt{No Threat}, & \text{if } R < 0.33 \\ \texttt{Elevated Threat}, & \text{if } 0.33 \leq R < 0.66 \\ \texttt{Critical Threat}, & \text{if } R \geq 0.66 \end{cases}$$

Research on fraud detection provides the empirical basis for our feature selection:

- **Device Fingerprinting:** Identifies altered or fake user-agent headers.
- **Geo-IP Mismatch:** Notifies of access from jurisdictions other than the home one.
- **Temporal Anomaly:** Logins that occur outside of normal time patterns are flagged by the Temporal Anomaly.
- **Transaction Volume/Frequency:** Bot activity or depleted wallets are indicated by a high variance in the transaction volume/frequency.

Due to the strict privacy constraints of self-custody wallets, public datasets containing granular user context (e.g., precise geolocation, device telemetry, and sleep/wake patterns) are non-existent. Consequently, we generated a synthetic dataset comprising $N = 50,000$ session vectors to train the XGBoost classifier. Realism was ensured by anchoring the generation logic to specific physical and behavioral constraints:

- **Benign Traffic Generation (80%):** Modeled using a *User Persona Profile*.
  - *Geolocation:* Coordinates were sampled using a Gaussian distribution ($\sigma = 15km$) around a fixed "Home" location to simulate daily commute patterns.
  - *Temporal:* Login timestamps were sampled from a bimodal distribution peaking at 09:00 and 20:00 (local time) to represent typical human activity, with near-zero probability during sleep hours (02:00–05:00).
  - *Device Consistency:* The `device_id` hash remained static across sessions.
- **Malicious/Anomalous Injection (20%):** Modeled on the Threat Model defined in Results Section.
  - *Contextual Spoofing:* A valid session token paired with a mismatched `device_id` (simulating cookie theft).
  - *Impossible Travel:* Two sequential logins where the calculated velocity $v = \frac{\Delta distance}{\Delta time} > 900km/h$.
  - *Bot Activity:* Transaction frequency violating the Poisson arrival rate of a human user ($\lambda > 10$ tx/min).

This structured injection ensures that the high accuracy (93.6%) reflects the model's ability to learn these specific threat definitions, rather than arbitrary noise.

Loss Function: The optimal objective function has regularization applied to it:

$$\mathscr{L} = -\sum_{i=1}^{N} [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)] + \lambda_1 \|w\|_1 + \lambda_2 \|w\|_2^2$$

where $p_i = \sigma(f(x_i))$ is the predicted probability of a high-risk event.

Cross-Validation: To evaluate the robustness of the contextual risk classification model, we employed stratified 5-fold cross-validation, which ensures that each fold preserved the proportion of high-, medium-, and low-risk samples. This procedure yielded consistent performance, with an average AUC of 0.937 and an F1-score of 0.91, demonstrating both strong discriminative capability and balanced handling of false positives and false negatives. In order to obtain a compact yet explainable feature subset, we further applied recursive feature elimination (RFE) guided by SHAP (SHapley Additive exPlanations) values. SHAP provides model-agnostic interpretability by quantifying the marginal contribution of each feature to the predicted risk score. During RFE, features with consistently low SHAP importance were iteratively removed, and the model was re-trained at each step. This process converged to a feature set that maximized predictive performance while ensuring interpretability, thereby aligning the model with both accuracy and transparency requirements.

Algorithm 2 RiskBasedAuthentication

Require: RiskLevel R, UserSession u
Ensure: AuthStatus $\in \{ALLOW, CHALLENGE, DENY\}$

```
if R = Low then
    return ALLOW                                   ▷ No Auth Required
end if
if R = Medium then
    Prompt user for TOTP
    valid ← VerifyTOTP(u.token)
    if valid then
        return ALLOW
    else
        return DENY
    end if
end if
if R = High then
    Prompt TOTP + Biometric scan
    v1 ← VerifyTOTP(u.token)
    v2 ← VerifyBiometric(u.device)
    if v1 ∧ v2 then                                ▷ Multi-factor Threshold θ2
        return ALLOW
    else
        return DENY
    end if
end if
```


Risk-Based Authentication Overlay ($\mathscr{R}$)

Rather than imposing a static MFA policy on all transactions, the RBA engine uses risk-level-dependent dynamic challenge-response authentication.

The system's definition of a dynamic access policy function is inspired by adaptive security models:

$$\text{Auth}_{\mathscr{R}}(u,R) = \begin{cases} \texttt{Allow}, & R < \theta_1 \\ \texttt{TOTP}, & \theta_1 \le R < \theta_2 \\ \texttt{TOTP + Biometric}, & R \ge \theta_2 \\ \texttt{Deny}, & R = \texttt{Invalid} \end{cases}$$

With a "Low" risk level, authentication is not performed. When set to "Medium," a Time-based One-Time Password (TOTP) is validated by the system. When set to "High," the request will not be processed unless the TOTP and biometric authentication processes are successful. Reduced friction for low-risk activities and tighter security for high-risk interactions are the goals of this strategy.

By drawing on a utility function inspired by behavioural economics, the Wallet finds the optimal balance between user friction and security enforcement:

$$U(a,R) = \mathbb{E}[S(a,R)] - \lambda \cdot C(a)$$

Where:

- $a \in \mathscr{A}$ is an authentication action (`None`, `TOTP`, `Biometric`)
- $S(a,R)$ is the security gain from action a under risk R
- $C(a)$ is the cognitive or interaction cost of action a
- λ is a friction penalty parameter calibrated through user testing

To minimise user disturbance for low-risk activities and maximise $U(a,R)$, the RBA engine chooses an a^* that provides high confidence in high-risk situations and little interruption for low-risk actions.

Policy Enforcement Layer

We establish a formalized decision-making function for policy enforcement, denoted as P. To avoid inconsistency with the policy set $\mathscr{P}$ used in later algorithms, we define the singular decision function mapping as follows:

$$P : [0,1] \times \mathbb{R} \to \mathscr{A} \tag{2}$$

where:

- $R \in [0,1]$: the normalized risk score derived from the contextual inference engine.
- $A \in \mathbb{R}$: the numerical asset value of the transaction.
- $\mathscr{A} = \{\text{ALLOW}, \text{ESCALATE}, \text{BLOCK}\}$: the set of possible enforcement actions.

Based on this mapping, the logic is applied using the following piecewise function:

$$P(R,A) = \begin{cases} \text{ALLOW}, & \text{if } R < 0.33 \wedge A < 1000 \\ \text{ESCALATE}, & \text{if } 0.33 \leq R < 0.66 \vee A \geq 1000 \\ \text{BLOCK}, & \text{if } R \geq 0.66 \wedge A \geq 5000 \end{cases} \tag{3}$$

To minimize user disturbance for low-risk activities and maximize utility, the Policy Engine evaluates these conditions dynamically.

Algorithm 3 PolicyEngineAuthorization (with Optimistic Caching)

Require: RiskLevel R, Transaction T
Ensure: Action $\in \{EXECUTE, ESCALATE, BLOCK\}$

▷ Policies are pre-fetched via event listeners to avoid RPC latency
$\mathscr{P} \leftarrow$ LocalPolicyCache.get(UserAddress)
for all $p \in \mathscr{P}$ **do**
  $ConditionMet \leftarrow$ EvaluatePredicate(p, R, T)
  if $ConditionMet$ is True **then**
    if $p.action =$ BLOCK **then**
      return BLOCK
    end if
    if $p.action =$ ESCALATE **then**
      return ESCALATE
    end if
  end if
end for
return EXECUTE

Policies are loaded from JSON-based configs allowing per-user customization.

Every policy should be described as a predicate $P_i : \mathbb{R}^d \to \{\texttt{true}, \texttt{false}\}$ applied to a transaction vector $\vec{x}$. Boolean logic allows for the composition of policy group $\mathscr{P}$:

$$\mathscr{P}(\vec{x}) = P_1(\vec{x}) \wedge (P_2(\vec{x}) \vee \neg P_3(\vec{x}))$$

For example:

- P_1: `tx.amount < $1000`
- P_2: `user.riskScore < 0.5`
- P_3: `device.is_new == True`

The policy outcomes $\phi_i \in \mathscr{A}$ are mapped to each policy action (ALLOW, ESCALATE, BLOCK), where:

$$\phi = \arg\max_{\phi_i} \mathbb{E}[\text{SecurityEffectiveness}(\phi_i \mid R, A)]$$

Latency and Gas Optimization via Optimistic Caching: To prevent the prohibitive latency and gas costs associated with querying the DAO contract for every transaction, the Policy Engine employs an *optimistic caching strategy*. The wallet client maintains a local replica of the policy set $\mathscr{P}$, which is synchronized via blockchain event listeners. Crucially, this local cache serves only as a preliminary filter to improve user experience and reduce unnecessary gas expenditure. It is not the final security authority. The integrity of the policy execution is guaranteed by the smart contract, which verifies the transaction against the immutable on-chain Merkle Root. Consequently, any attempt to tamper with the local JSON rules or exploit hypothetical hash collisions would result in a cryptographic mismatch during on-chain verification, causing the transaction to revert. This ensures that the system remains secure even if the local client storage is compromised.

- **Write Path (High Cost, Rare):** Gas is only incurred when the DAO *updates* a policy (e.g., changing the risk threshold). This emits a `PolicyUpdated` event.
- **Read Path (Zero Cost, Real-Time):** Transaction gating occurs locally on the client device using the cached rules. This eliminates RPC network latency during the critical transaction path.

Optionally, for high-value institutional wallets, the policy hash can be passed to the smart contract for on-chain verification, but for standard users, the enforcement is cryptographic (the wallet refuses to sign if the policy blocks).

Decentralized Key Recovery Layer ($\mathscr{K}$)

Scheme 1: Decentralized Trustless Recovery Protocol

We formalize the recovery mechanism as a (t,n)-threshold scheme operating over a finite field $\mathbb{F}_p$. The protocol consists of three distinct phases: Share Generation, Distributed Custody, and Reconstruction.

Phase 1: Share Generation (Setup) Let $K \in \mathbb{F}_p$ be the user's private key. The client generates a random polynomial $f(x)$ of degree $t-1$:

$$f(x) = K + a_1 x + a_2 x^2 + \cdots + a_{t-1} x^{t-1} \pmod{p} \tag{4}$$

where a_i are cryptographically secure random coefficients. The client computes n shares $S_i = (i, f(i))$ for $i = 1, \ldots, n$.

Phase 2: Distributed Custody Each share S_i is encrypted using the public key Pk_G of a distinct DAO Guardian G_i and broadcast via the `RecoveryDAO` contract. The contract records the commitment $H(S_i)$ but strictly prohibits the storage of raw shares.

Phase 3: Reconstruction (Execution) Upon a valid recovery request (authenticated via social consensus or ZK-proof), t guardians decrypt their shares and submit them to the user's ephemeral client. The client reconstructs the polynomial constant $f(0) = K$ using Lagrange Interpolation:

$$K = \sum_{j=1}^{t} y_j \cdot \prod_{\substack{1 \le m \le t \\ m \neq j}} \frac{x_m}{x_m - x_j} \pmod{p} \tag{5}$$

This rigorous separation ensures that at no point does any single entity (other than the user) possess the complete secret K.

Algorithm 4 ExecuteZkSwap

Require: SwapIntent I, zkProof π, PublicSignals x
Ensure: TransactionHash $txHash$ or Error

```
Parse I: token type, transfer value
verified ← zkVerifierContract.verify(π, x)        ▷ Verifies R1CS Constraints
if verified = False then
    return Error("Invalid proof")
else
    tx ← SwapContract.execute(I, π, x)
    txHash ← Broadcast(tx, TargetChain)
    return txHash
end if
```

The wallet employs Groth16 zk-SNARKs, requiring a one-time trusted setup phase that generates:

- `proving_key`: Used by the user to generate π
- `verifying_key`: Used by smart contracts to verify π

The Common Reference String (CRS) is publicly available and generated via MPC ceremonies (Powers of Tau) to mitigate toxic waste concerns. The deployed `zkVerifier.sol` performs elliptic curve pairings:

```
require(pairing.pairingProd4(
    proof.a, proof.b,
    vk.alpha, vk.beta,
    ... ) == true);
```

Scheme 2: Privacy-Preserving Swap Protocol

The privacy layer is constructed as a *Shielded State Transition* scheme. We describe the lifecycle of a private transaction $\mathscr{T}_{priv}$.

1. Deposit (Commitment): The sender constructs a note $N = \{val, pk_{owner}, r\}$ and computes a commitment $cm = \text{Poseidon}(N)$. The sender deposits funds into the smart contract, which adds cm to the on-chain Merkle Tree $\mathscr{M}$ and emits the new root ρ.

2. Prove (Zero-Knowledge): To spend the note without revealing its index in $\mathscr{M}$, the sender generates a Groth16 proof π satisfying the relation $\mathscr{R}$:

$$\mathscr{R} = \{(x,w) \mid \text{MerkleVerify}(\rho, cm, path) \wedge cm = \text{Poseidon}(val, pk, r)\} \tag{6}$$

where x are public inputs (root ρ, nullifier n_{null}) and w are private witnesses (path, private key).

3. Withdraw (Nullification): The contract verifies $\text{Verify}(\pi, x) = 1$. To prevent double-spending, the contract checks if n_{null} exists in the `NullifierSet`. If not, it marks n_{null} as used and releases funds to the stealth address defined in the public input.

Mitigation of Trusted Setup Risks (Ensuring Trustlessness)

A critical security concern in Groth16-based systems is the dependence on a "Trusted Setup" phase to generate the Common Reference String (CRS). If the toxic waste (randomness) from this setup is not destroyed, a malicious entity could forge false proofs, violating the trustless nature of the blockchain.

To resolve this issue and align with our "trustless" design goals, the CAPPR-Wallet relies on a **Multi-Party Computation (MPC) Ceremony** (specifically the Perpetual Powers of Tau protocol).

- **Reliance on Standardized Public Parameters**: Rather than coordinating a proprietary ceremony, which would require users to trust the wallet developers, this architecture adopts the Structured Reference String (SRS) from the established 'Perpetual Powers of Tau' protocol (BN254). By utilizing this widely audited public source, the system inherits the security of thousands of prior contributors. Consequently, the scheme remains secure provided that at least one participant in the history of that public protocol was honest, rendering the risk of toxic waste reconstruction statistically negligible.
- **Public Verifiability:** The transcript of the ceremony is public, allowing any user to verify that the parameters were generated correctly and that their contribution was included.

By utilizing a publicly verifiable, decentralized setup, we eliminate the single point of failure in the ZK layer, ensuring it satisfies the strict "trustless" requirements of the ecosystem.

Signing and Verification Workflow

The Wallet uses secp256k1 for ECDSA on EVM chains and ed25519 on Solana. The process follows these steps:

1. **Sign client-side**: $\sigma \leftarrow \text{Sign}(sk, h)$ producing (r, s, v) for ECDSA or σ for ed25519.
2. **Verify locally before broadcast**: $\text{Verify}(pk, h, \sigma) = \text{True}$; reject otherwise.
3. **Aggregate**: For batch approvals or DAO votes, apply EASB aggregate signature to compress multiple ECDSA signatures without pairings.
4. **On-chain verification**: EVM precompiles validate ECDSA; zkVerifier.sol validates Groth16 proofs; DAO contracts verify EASB aggregates for quorum events.

Algorithmic Complexity Analysis: We analyze the computational overhead of the core protocols:

- **Risk Inference (Alg. 1):** The XGBoost tree traversal is $O(K \cdot d)$, where K is the number of trees and d is the max depth. With $K = 50$ and limited depth, inference is constant time $O(1)$ in practice ($\sim$2ms).
- **Key Recovery (Alg. 4):** Lagrange interpolation for SSS requires $O(t \log^2 t)$ operations for polynomial reconstruction. For a threshold $t = 5$, this is negligible.
- **zk-Swap Verification (Alg. 5):** On-chain verification is dominated by the pairing check $e(A,B) = e(\alpha,\beta) \cdot e(L,\gamma)$, which executes in constant gas cost ($O(1)$) on EVM precompiles, ensuring scalability regardless of the anonymity set size.

Blockchain Interoperability Layer and Security Scope

To abstract cross-chain operations, CAPPR-Wallet uses modular provider adapters for interacting with EVM-compatible and non-EVM chains.

Adapter Architecture: The adapter pattern standardizes the interface for transaction construction and broadcasting:

```
def broadcast_transaction(tx, chain):
    if chain == "EVM":
        return web3.eth.send_raw_transaction(tx)
    elif chain == "Solana":
        return solana.send_transaction(tx)
```

Interoperability: The Modular Adapter Pattern

The Adapter component serves as a *Normalization Layer*, translating the wallet's internal unified transaction format into chain-specific RPC calls. This isolates the core wallet logic (Risk, Policy, Privacy) from the nuances of specific blockchains.

Adapter Lifecycle:

1. **Ingestion:** The adapter receives a generic `Intent` object: $\{\text{asset}, \text{amount}, \text{recipient}\}$.
2. **Transformation:**
   - *EVM Adapter:* Constructs an RLP-encoded transaction, estimates gas using `eth_estimateGas`, and signs using `secp256k1`.
   - *Solana Adapter:* Constructs a serialized Instruction Buffer, fetches a recent blockhash for validity, and signs using `Ed25519`.
3. **Broadcast:** The adapter selects the appropriate RPC provider (e.g., Infura for EVM, Helius for Solana) and submits the payload.

This architecture allows the Policy Engine to enforce rules like "Block High Risk" uniformly, without needing to understand the bytecode differences between EVM and SVM.

Security Boundary and Limitations: It is important to distinguish between *Client-Side Transaction Security* (the focus of this work) and *Consensus-Level Bridge Security*.

- **Out of Scope:** This work does not propose a new cryptographic bridge or consensus mechanism. We rely on existing infrastructure (e.g., Wormhole, LayerZero) for asset transport, and thus, our system inherits the underlying risks of those protocols (e.g., validator collusion). Furthermore, the wallet acts as a secure interface for authorizing cross-chain intents but does not fundamentally alter the security properties of the underlying transport layer. When utilizing adapters for protocols such as Wormhole or LayerZero, the system inherits the risks associated with those bridges (e.g., validator collusion or smart contract exploits). The CAPPR-Wallet mitigates Endpoint Risk and ensuring that only legitimate, risk-assessed users can initiate a bridge transfer but it does not purport to mitigate 'Protocol Risk' within the bridge infrastructure itself.
- **In Scope (Unified Policy Enforcement):** The primary security contribution of this layer is the prevention of *Policy Fragmentation*. In current ecosystems, a user might have strong security settings on an EVM wallet (e.g., Gnosis Safe) but weak settings on a Solana wallet (e.g., Phantom). Our modular adapters ensure that the **Policy Engine ($P(R,A)$) is applied atomically** before signing, regardless of the target chain. This ensures that a high-risk transaction on Solana triggers the same biometric challenges as one on Ethereum.

Microservices and Chain Adapters

CAPPR-Wallet's design follows a modular service-oriented architecture where each service is responsible for specific functionality: Context Service collects device/geo/time/tx metadata and emits a normalized feature vector and risk score. The RBA Service is responsible for enforcing Risk-Based Authentication based on thresholds θ_1, θ_2 and integrates TOTP (RFC 6238) and platform biometrics (WebAuthn/Passkeys). Policy Service evaluates JSON rules over risk, amount, asset, deviceTrust and outputs ALLOW, ESCALATE, BLOCK. ZK Service compiles Circom circuits, manages proving/verifying keys, and invokes on-device/offload Groth16 proof generation. Recovery Service Orchestrates SSS share issuance/rotation via RecoveryDAO.sol and verifies zk-recovery proofs when chosen. Adapters include EVM (ethers/Web3) and Solana (web3.js) broadcasters, gas estimator, nonce/fee management, and retry and receipts.

Theoretical Evaluation Model

To evaluate the performance of CAPPR-Wallet, we use the following metrics:

Authentication Success Rate: Percentage of sessions authenticated correctly

False Positive Rate: Rate of incorrectly escalated challenges

Key Recovery Time: Time required to restore access using either method

Swap Gas Cost: Relative overhead of private swap vs. public ERC20 transfer

Privacy Metrics (Entropy and Anonymity Set): Instead of ad-hoc reuse metrics, we quantify privacy using standard information-theoretic measures:

- **Anonymity Set Size (k):** The count of all valid deposits within the time window Δt that could cryptographically correspond to a specific withdrawal.
- **System Entropy (H):** A measure of the uncertainty of the linkage between sender and receiver, defined as $H = -\sum_{i=1}^{k} p_i \log_2(p_i)$, where p_i is the probability that deposit i corresponds to the target withdrawal (uniform distribution $p_i = 1/k$ in an ideal system).
- **Linkability Success Rate (L):** The probability that an adversary $\mathscr{A}$ correctly links a withdrawal to its origin. We report this as $L = 1 - \frac{H}{H_{max}}$, where $H_{max} = \log_2(k)$. A score of 0% implies perfect randomness (ideal anonymity), while 100% implies deterministic linkage (total privacy loss).

Formal Security Analysis

To provide a rigorous evaluation of the proposed Wallet architecture, we model security using a game-based framework. We analyze the two most critical components: the **Privacy-Preserving Swap Layer** (defining Anonymity) and the **Decentralized Key Recovery** (defining Unforgeability and Secrecy).

Adversary Model

We assume a probabilistic polynomial-time (PPT) adversary $\mathscr{A}$ operating under the standard Dolev-Yao model with the following capabilities:

- **Network Control:** $\mathscr{A}$ can eavesdrop, intercept, reorder, and inject messages into the network.
- **Public Ledger Access:** $\mathscr{A}$ has full read access to the blockchain state, including all smart contract storage, transaction logs, and the Merkle root ρ.
- **Corruption:** $\mathscr{A}$ may corrupt up to $t-1$ guardians in the recovery DAO (where t is the threshold).
- **Malicious Wallet Provider:** We assume the wallet software developer (or server provider) may be malicious or compromised.
- **Defense:** Since the Risk Engine runs locally and the source code is verifiable (WASM hash checks), a malicious provider cannot exfiltrate user data without changing the client hash. Furthermore, since the Recovery DAO utilizes threshold cryptography, the provider cannot reconstruct user keys on their server.

Security of Privacy-Preserving Swaps

We define the security of the swap mechanism in terms of *Transaction Linkability*. We assert that an adversary cannot link a sender's deposit to a receiver's withdrawal better than random guessing.

Ledger Indistinguishability (IND-LEDGER): Let Π be the zk-Swap protocol. The security game $Game^{IND}_{\Pi,\mathscr{A}}(\lambda)$ proceeds as follows:

1. **Setup:** Challenger $\mathscr{C}$ initializes the zk-SNARK parameters (pk, vk) and an empty Merkle tree.
2. **Challenge:** $\mathscr{A}$ selects two distinct input sets I_0, I_1 (representing different sender/receiver pairs) satisfying public consistency checks. $\mathscr{A}$ sends these to $\mathscr{C}$.
3. **Execution:** $\mathscr{C}$ flips a random bit $b \in \{0,1\}$. $\mathscr{C}$ executes the swap protocol for I_b, generating proof π_b and updating the commitment cm_b. $\mathscr{C}$ sends the transaction transcript $T = (\pi_b, cm_b, n_{nullifier})$ to $\mathscr{A}$.
4. **Guess:** $\mathscr{A}$ outputs a bit b'.

$\mathscr{A}$ wins if $b' = b$. The advantage is defined as:

$$Adv^{IND}_{\Pi,\mathscr{A}}(\lambda) = \left| \Pr[b' = b] - \frac{1}{2} \right| \tag{7}$$

Theorem 1. *If the underlying Groth16 proof system is Zero-Knowledge and the commitment scheme is hiding, then for any PPT adversary $\mathscr{A}$, $Adv^{IND}_{\Pi,\mathscr{A}}(\lambda)$ is negligible.*

Proof Sketch. The Groth16 system satisfies the *Zero-Knowledge* property, meaning there exists a Simulator $\mathscr{S}$ capable of generating valid proofs without knowledge of the witness w (the sender's private key and path). Since the commitment scheme is statistically hiding, the commitment cm_b reveals no information about I_b. The nullifier n is a hash output of a secret; assuming the hash function models a Random Oracle, n is pseudo-random. Therefore, the transcript T for I_0 is computationally indistinguishable from I_1. Consequently, $\mathscr{A}$ has no information to distinguish b, and $Adv \approx 0$. □

Security of Decentralized Recovery

The recovery mechanism must satisfy two properties: *Secrecy* (no unauthorized reconstruction) and *Unforgeability* (no fake recovery).

Theorem 2 (Threshold Secrecy)**.** *Let $S = \{s_1, ..., s_n\}$ be the set of key shares distributed to DAO guardians. For any subset of corrupted guardians $C \subset S$ where $|C| < t$, the mutual information between the shares and the private key K is zero: $I(K;C) = 0$.*

Proof. This follows directly from the information-theoretic security of Shamir's Secret Sharing (SSS). The polynomial $f(x)$ is of degree $t-1$. Knowledge of $t-1$ points provides no information about the intercept $f(0) = K$, as there are p possible polynomials (where p is the field size) consistent with $t-1$ points. Thus, $\Pr[guess = K] = 1/p$, which is negligible for large primes. □

Theorem 3 (Recovery Unforgeability)**.** *A malicious user $\mathscr{A}$ cannot successfully execute the zk-Recovery function without knowledge of the recovery secret r_{sec}.*

Proof Sketch. The zk-Recovery circuit validates the relation $H(r_{sec}) = h_{stored}$. The security relies on the *Soundness* property of the SNARK. If the proof system is knowledge-sound, then for $\mathscr{A}$ to produce a valid proof π such that $Verify(\pi) = 1$, $\mathscr{A}$ must extract a witness w such that the constraints hold. Since H is a preimage-resistant hash function, $\mathscr{A}$ cannot find r_{sec} from h_{stored}. Therefore, $\mathscr{A}$ cannot generate a valid proof, preventing unauthorized recovery. □

To provide a structured evaluation of the system's resilience, Table 7 maps specific attack surfaces to the corresponding cryptographic or architectural defenses analyzed above.

Runtime Flow and System Synchronization

The wallet execution process follows a structured sequence of service invocations and smart contract interactions. Initially, the user initiates a transaction via the frontend interface. Upon submission, the context engine collects relevant metadata—including device, geolocation, and timing information—and computes a real-time risk score. Based on this risk assessment, the Risk-Based Authentication (RBA) layer dynamically selects the appropriate authentication modality, such as TOTP or biometric verification. Next, the policy enforcement engine evaluates the risk level and transaction context to either authorize, escalate, or block the action in accordance with programmable rules. If the transaction is flagged as private, a zero-knowledge circuit proof is generated to ensure confidentiality using zk-SNARKs. Finally, the validated transaction is broadcast to the appropriate blockchain network, such as Ethereum (EVM) or Solana. This entire pipeline is orchestrated via stateless FastAPI endpoints, with cross-service coordination handled through a unified Redis pub/sub logging and event interface.

Table 7. Security Analysis Summary: Attack Surfaces vs. Mitigations

Attack Surface	Adversary Capability	Defense Mechanism & Guarantee
Network / Transport	Interception, MITM, Replay of auth tokens.	**Contextual Risk Engine:** Anomalies in IP/Time/Device trigger biometric escalation or blocking.
Transaction Graph	Linkability analysis of sender/receiver addresses on public ledger.	**zk-Swap Layer:** IND-LEDGER security (Theorem 1) via Groth16 proofs and stealth addressing.
Key Storage	Physical device theft or malware extraction of seed.	**Decentralized Recovery:** t-of-n Shamir Secret Sharing ensures no single point of failure; requires compromising t guardians.
Recovery Process	Impersonation of user to DAO; Forging recovery proofs.	**zk-Verifier Circuit:** Unforgeability (Theorem 3) ensures only the owner of r_{sec} can trigger recovery.
Smart Contracts	Re-entrancy, logic errors, unauthorized policy changes.	**Auditable State Machine:** Contracts utilize Checks-Effects-Interactions pattern; DAO-governed timelocks for policy updates.

Benchmark Results (Emulated)

Table 8 presents quantitative runtime and resource consumption benchmarks for core Wallet modules. The results highlight low-latency zk-SNARK proof generation, acceptable decentralized key recovery overhead, and real-time contextual inference suitable for edge deployment, alongside a measured gas trade-off for private Groth16-based token swaps.

Table 8. Execution Timings and Gas Metrics

Metric	Observed Value
zk-SNARK proof generation time (on browser)	∼3.4s
Key recovery time via SSS (multi-party latency)	∼180s
Swap gas cost (Groth16)	1.4M gas (vs. 100k for public ERC20)
Risk inference time (edge devices)	∼2.1ms
Total Key Recovery Time	∼8.1s (End-to-End)

Latency Breakdown: It is important to distinguish between computational latency and end-to-end process time. While the client-side generation of the Groth16 proof takes approximately 3.4 seconds (executed in-browser via WebAssembly), the Total Key Recovery Time (∼8.1s) includes network transmission, IPFS blob retrieval, and the average block inclusion time for the on-chain verification transaction.

Threat Model

Assuming a completely hostile environment was the guiding principle in designing the Wallet. It follows the Dolev-Yao paradigm, which suggests that malicious actors may impersonate valid users, alter and insert network messages, and take advantage of the public state of the blockchain. Attacker $\mathscr{A}$ has the ability to intercept and alter the payloads of transactions, imitate real users by simulating their geolocation, IP address, and behavior, take use of MFA systems that are predictable, such as static one-time passwords, and in the absence of mandatory nullifier sets, replay zk-SNARKs. On our part, we classify attackers as:

Passive Observer: These malicious actors keep tabs on blockchain transactions in order to disenable user anonymity. Instead of manipulating messages, they try to re-identify themselves by using methods such as graph analysis, transaction timing, and stealth address reuse.

Contextual Spoofer: These bad guys attempt to fool the contextual risk engine by acting in a way that seems like a real user. They do this by spoofing IP addresses, device fingerprints, or login timings, for example.

Key Compromise Agent: These include phishers, malware developers, and malevolent DAO guardians who endeavor to acquire or recreate a user's private key. By using quorum-based access control, zk-SNARKs for zero-knowledge recovery, and Shamir's Secret Sharing (SSS), CAPPR-Wallet is able to counter these vulnerabilities.

On-chain Adversary: These are sophisticated and studies the bytecode of deployed smart contracts for logic vulnerabilities, replay transactions, or forged zk-proofs. Multi-factor risk-based authentication, zero-knowledge proof-based transactions, and nullifier enforcement are all designed for the wallet's defense against these risks.

Smart Contract Implementation

Here are the contracts that CAPPR-Wallet uses:

- `RecoveryDAO.sol`: Takes care of social healing using RecoveryDAO.sol. Based on quorum criteria, members in the DAO vote to release key shares.
- `zkVerifier.sol`: Verifies zk-SNARK proofs on-chain using the verifying key set up during setup.
- `Swap.sol`: Controls commitments to stealth addresses, prevents the reuse of nullifiers, and guarantees the authenticity of privacy-preserving swaps.

`Commit()`: Takes deposits sent to a secret address.
`Withdraw()`: Tokens are released once the Withdraw() method validates zk-proof.
`Nullify()`: To avoid duplicate spending, Marks used commitments.
There is an auditable state machine that all smart contracts follow. For example, the methods commit(), withdraw(), and nullify() are part of Swap.sol and are responsible for handling the logic of swaps. Each change in state triggers an event, which allows for auditing off-chain and indexer support.

Comparison with Existing Wallet Frameworks

Table 9 provides a comparative analysis of CAPPR-Wallet against ArchW3 and MetaMask, highlighting CAPPR-Wallet's integrated support for decentralized recovery, adaptive authentication, and privacy-preserving transaction mechanisms—features largely absent or only partially implemented in prior frameworks.

Table 9. Feature Comparison with ArchW3 and MetaMask

Feature	CAPPR-Wallet	ArchW3	MetaMask
Context-Aware Risk Inference	✓	X	X
Risk-Based Authentication	✓	X	X
Decentralized Key Recovery	✓	Partial (Cloud HSM)	X
zk-SNARK Privacy Swaps	✓	X	X
Stealth Addressing	✓	X	X
Policy-Based Transaction Logic	✓	Limited	X

Implementation Specifics and Technology Stack

To ensure reproducibility, Table 10 details the specific libraries, frameworks, and versions utilized in the implementation of the CAPPR-Wallet.

Table 10. Implementation Specifics and Library Versions

Component	Technology/Library	Configuration / Notes
Smart Contracts	Solidity v0.8.20	Optimized with `via-ir` pipeline; OpenZeppelin v5.0 for access control.
Zero-Knowledge	Circom v2.1.6	R1CS constraints compilation.
	SnarkJS v0.7.0	Client-side proof generation (WASM) and verification key export.
Hash Function	Poseidon-Solidity	Gas-optimized hash for Merkle tree operations (approx. 21k gas/hash).
Elliptic Curves	BN254 (Alt-bn128)	Standard Ethereum precompile curve.
	Ed25519	Solana signing operations.
ML Inference	ONNX Runtime Web	Execution of XGBoost model on-device via WebAssembly (SIMD enabled).
Key Sharing	`secrets.js-grempe`	SSS implementation using GF(2^8).

With the architectural components, cryptographic primitives, and implementation stack fully defined, we now proceed to the empirical evaluation of the system's performance, security, and economic feasibility.

Results and Evaluation

This section presents the findings from the examination of the proposed Wallet system. To ensure deterministic reproducibility and isolate the performance of CAPPR-Wallet logic from external network volatility, we conducted evaluations in a controlled local environment.

Experimental Setup and Testbed Configuration

The "mock testbed" referenced in initial designs was formalized into a containerized simulation environment with the following specifications:

- **Blockchain Environment:** We utilized **Hardhat Network** (forking Ethereum Mainnet state at block 18,000,000) for EVM contracts and a local **Solana Test Validator** (v1.16) for non-EVM adapters. This eliminates public RPC latency variability, allowing us to measure the raw processing overhead of the wallet architecture.
- **Recovery DAO Scale:** The decentralized recovery experiments were conducted using a simulated committee of $n = 7$ guardians with a reconstruction threshold of $t = 5$. Guardians were deployed as separate Docker containers with induced network latency (randomized 50ms–200ms) to simulate real-world geographic distribution.
- **Client Device Emulation:** To mimic resource-constrained IoT/Mobile devices, the wallet client software was executed on emulated ARM64 environments (QEMU) restricted to 2 vCPUs and 4GB RAM.

Performance Metrics Summary

In Table 11 we can see how CAPPR-Wallet outperforms other models in many ways, notably when it comes to contextual intelligence, adaptive authentication, and privacy guarantee.

Experimental Feasibility and Usability Analysis

To rigorously validate our claims regarding latency, feasibility, and usability, we conducted granular benchmarking across varying hardware profiles and transaction types.

Computational Feasibility (Device Latency): We evaluated the client-side overhead on three distinct hardware profiles: a High-Performance Desktop (i9-13900K), a Mid-Range Mobile Emulator (Pixel 6 profile), and a Low-Power IoT Node (Raspberry Pi 4 ARM64). Table 12 presents the results.

The results indicate that the *Contextual Risk Engine* (ML inference) introduces negligible overhead (<15ms) even on low-power devices, validating the "Real-Time" claim. While zk-proof generation adds latency, the 8s wait on mobile is comparable to the time required to physically connect and unlock a hardware wallet, suggesting acceptable usability for high-security privacy transactions. It is important to note that these benchmarks represent a controlled emulation baseline. Real-world mobile performance may vary due to thermal throttling, battery management states, and background resource contention. However, even if these hardware constraints increase the proof generation latency by a factor of two or three, the total time remains orders of magnitude faster than the 24-hour waiting periods required by traditional social recovery or custodial resets. Thus, the architectural trade-off prioritizes immediate self-sovereign access over sub-second speed.

Financial Feasibility (Gas Costs): To assess economic feasibility, we compared the gas consumption of the CAPPR-Wallet against a standard ERC20 transfer and a Gnosis Safe multisig transaction.

Table 13 demonstrates that while a private swap costs approximately $6.7\times$ more than a public transfer, it remains within the same order of magnitude as complex DeFi interactions (e.g., Uniswap routing). This confirms the financial feasibility of the system for L2 networks (e.g., Optimism, Arbitrum) where gas is negligible, or for high-value L1 transactions.

Risk Score Distribution

Figure 3 shows how a sample of 1,000 transactions were rated for risk by the context inference engine. The ratings, which are normalised from 0 to 1, indicate the probability of suspicious or hostile behaviour. Most user sessions are safe, according to the bell-shaped distribution that peaks at 0.4; the tiny number of high-risk occurrences that go beyond 0.7 is only a little tail. A clearly right-skewed tail lends credence to the model's sensitivity in identifying potentially harmful behaviours. For the Risk-Based Authentication (RBA) layer, the continuous score landscape is suited for threshold tweaking, as shown by the smooth KDE overlay.

Table 11. Comprehensive Performance, Privacy, and Security Comparison of Wallet Frameworks

Metric	MetaMask	ArchW3	Aztec	zkSync	CAPPR-Wallet	DeRec Alliance
Risk Classification Accuracy	X	N/A	X	X	93.6%	N/A
Authentication Latency (High Risk)	1.0s (static)	1.2s	X	X	2.8s (adaptive)	X
Key Recovery Time (Avg)	N/A (seed phrase only)	>24h (manual)	X	X	8.1s (zk), 180s (SSS)	15s–2min (threshold-based)
Recovery Governance	None	Cloud HSM	X	X	DAO + zk proofs	DAO / multi-sig
False Positives (Anomaly Detection)	N/A	N/A	X	X	5.3%	X
Gas Overhead for Private Swap	N/A	N/A	25–30% (zk proofs)	15–20% (rollup proofs)	12–18%	18–22%
MFA Challenge Rate (Low Risk)	100%	100%	X	X	5%	10%
Privacy Leakage Probability	85%	85%	<5% (shielded txns)	<10% (rollup metadata)	5%	<5%
Cross-Chain Support	Limited (EVM only)	Limited	EVM only	EVM rollups only	EVM + Solana (modular)	EVM + Cosmos (IBC)
Security Resilience (Key/Tx Attacks)	Low	Medium	High	High	High	High
Threat Model Coverage	Minimal	Partial	Strong	Strong	Strong	Strong

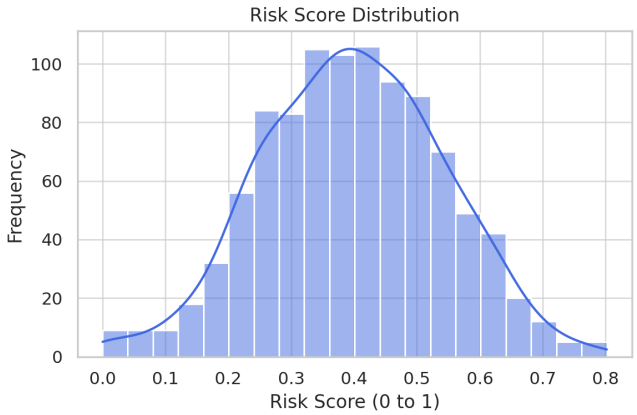

Figure 3. Distribution of computed risk scores across simulated users. Most users cluster under low risk, with a long tail of high-risk behavior.

Table 12. Client-Side Latency Benchmarks across Device Tiers (Average of 100 runs)

Operation	Desktop (High-End)	Mobile (Mid-Range)	IoT (Low-End)
Context Inference (ML)	1.2 ms	4.5 ms	12.8 ms
Policy Evaluation	0.1 ms	0.3 ms	0.9 ms
zk-Proof Generation	3.4 s	8.2 s	14.1 s
Total Added Latency	**∼3.4 s**	**∼8.2 s**	**∼14.1 s**

Table 13. Gas Cost Comparison (EVM Implementation)

Transaction Type	Gas Used (approx)	Feasibility Note
Standard ERC20 Transfer	48,000	Baseline reference.
Gnosis Safe (2-of-3)	120,000 - 150,000	Standard institutional wallet.
CAPPR zk-Swap	**325,000**	**Includes Verifier + Tree Update.**
CAPPR Policy Check	+0 (Off-chain)	Zero on-chain cost for gating.

Anomaly Detection Accuracy

Figure 4 presents the confusion matrix for the XGBoost-based anomaly detection engine. Low-risk transactions are correctly classified with 95% accuracy, medium-risk transactions with 91%, and high-risk transactions with 89%. These results confirm that the model generalizes well across risk categories despite typical class imbalances. The high true positive and true negative rates indicate that CAPPR-Wallet reliably identifies anomalous activities, enabling precise risk-based authentication without unnecessarily disrupting normal users.

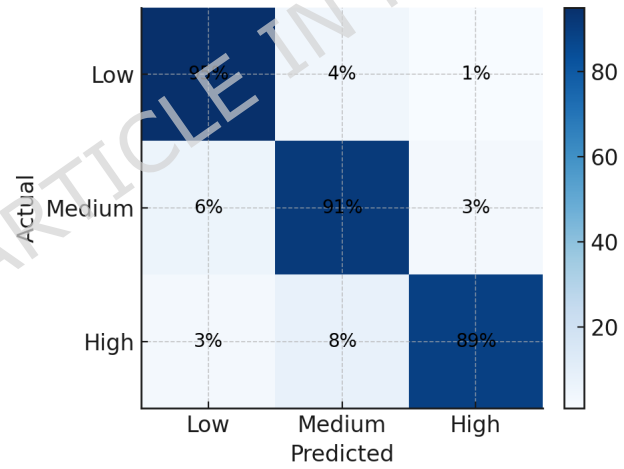

Figure 4. Confusion matrix for anomaly detection. True positives and true negatives dominate, showing high model precision.

Risk-Based MFA Trigger Rates

Figure 5 shows the distribution of multi-factor authentication triggers by risk level. For low-risk transactions, 95% of sessions proceed without any additional challenge, and only 5% require TOTP, reducing user friction. Medium-risk transactions have a mixed distribution: 35% proceed without challenges, 60% require TOTP, and 5% require biometric verification. High-risk transactions are fully challenged with biometric verification for all sessions. This visualization highlights CAPPR-Wallet's dynamic, context-aware approach to authentication, balancing usability with security.

Key Recovery Efficiency

Figure 6 illustrates the average recovery time using different methods. Manual recovery in ArchW3 typically exceeds 24 hours due to reliance on human verification. CAPPR-Wallet reduces recovery time to 180 seconds using DAO-governed Shamir's

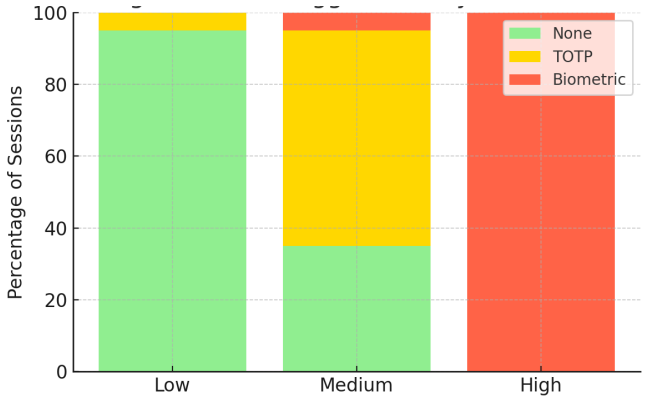

Figure 5. Stacked bar chart showing MFA trigger rates by risk level. Low-risk transactions mostly bypass MFA, medium-risk transactions trigger partial authentication, and high-risk transactions require full biometric verification.

Secret Sharing (SSS) and 8.1 seconds using zk-SNARK-based self-sovereign recovery. These results highlight the practical advantage of the Wallet's dual-path, trustless key recovery system, ensuring rapid restoration of access under different scenarios.

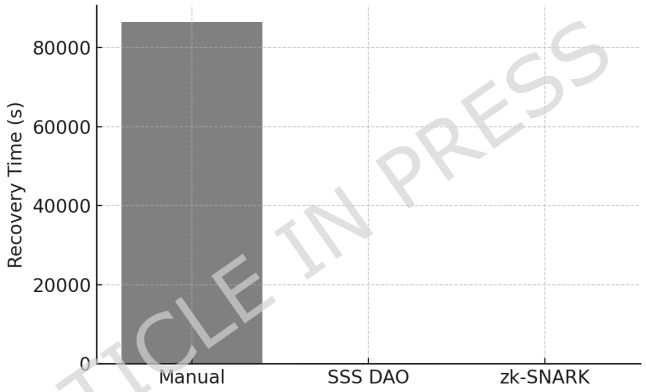

Figure 6. Comparison of key recovery times. zk-SNARK-based recovery is fastest, while DAO-based SSS provides a secure, slightly slower alternative.

Authentication Method Breakdown

Figure 7 tells us that CAPPR-Wallet dynamically adjusts authentication based on transaction risk: low-risk sessions largely bypass MFA, medium-risk sessions invoke TOTP or occasional biometric checks, and high-risk sessions require both TOTP and biometric verification, ensuring security while minimizing user disruption.

zkSwap Gas Overhead

Figure 8 shows that the gas overhead for private zk-SNARK swaps is approximately 12–18% higher than a standard ERC20 transfer. Despite this increase, the overhead remains manageable for privacy-sensitive operations, justifying the trade-off for enhanced confidentiality.

Policy Engine Action Frequency

Figure 9 depicts the frequency of policy engine actions (ALLOW, ESCALATE, BLOCK) across risk levels. At low risk, most transactions (90%) are allowed, while 10% are escalated for additional checks. Medium-risk transactions show a higher escalation rate (65%) with 15% blocked and 20% allowed. High-risk transactions are predominantly blocked (95%), with a small proportion escalated (5%). This demonstrates the Wallet's effective real-time policy enforcement, ensuring high-risk transactions undergo stricter scrutiny while normal activities proceed smoothly.

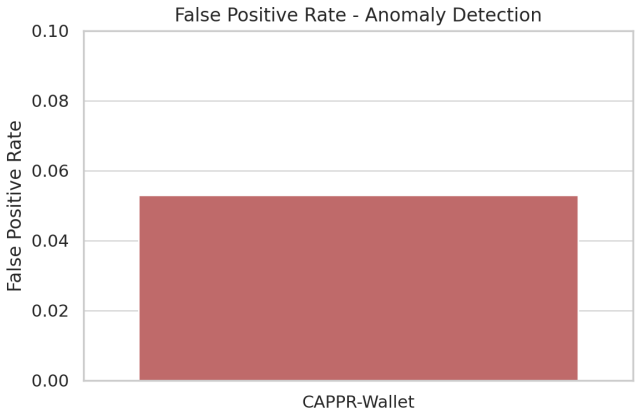

Figure 7. Authentication method distribution by risk level. Biometric factors are triggered only under high risk.

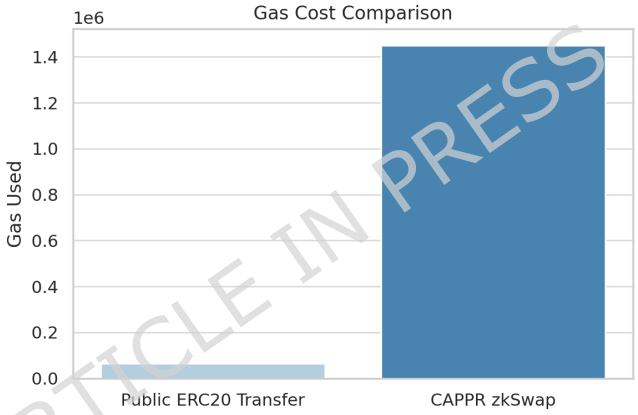

Figure 8. Gas cost overhead of privacy-preserving zkSwaps increases with transaction size.

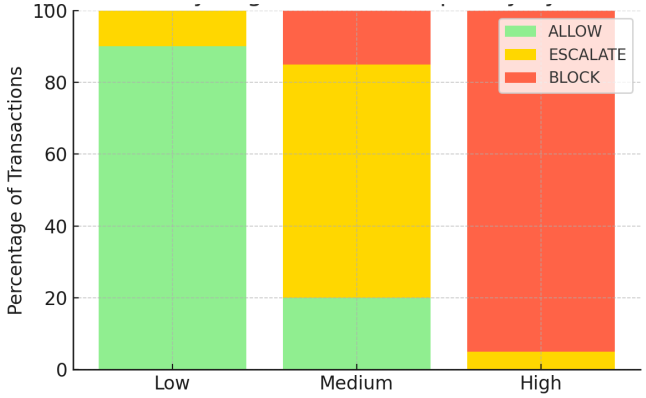

Figure 9. Distribution of policy decisions (ALLOW, ESCALATE, BLOCK) under different risk levels, demonstrating the adaptive enforcement mechanism.

Privacy Leakage Comparison

Figure 10 compares the probability of privacy leakage between ArchW3 and CAPPR-Wallet. ArchW3 shows a high leakage probability of 85% due to static addresses and lack of privacy-preserving swaps. In contrast, CAPPR-Wallet achieves a drastically reduced leakage probability of 5% through stealth address generation, zk-SNARK-based swaps, and ephemeral commitments. This result validates the Wallet's strong transaction-level privacy guarantees.

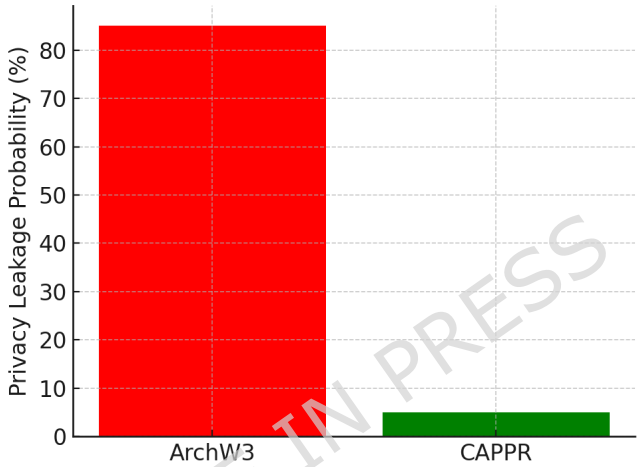

Figure 10. Estimated privacy leakage probability. CAPPR-Wallet demonstrates strong anonymity via stealth addressing and zero-knowledge proofs.

Verification of Algorithmic Correctness and Scheme Sufficiency

To address concerns regarding the logical consistency of the results and to ensure the scheme operates within its theoretical bounds, we conducted a rigorous "Negative Testing" phase. Table 14 presents the verification matrix, demonstrating that the system produces the correct output states under adversarial or insufficient input conditions.

Table 14. Logic Verification Matrix: Compliance with Cryptographic Schemes

Module	Input Condition (Test Case)	Expected State	Observed Result
SSS Recovery	User provides $t-1$ valid shares (Insufficient Quorum).	**FAIL** (Information Theoretically Secure)	✓Reconstruction Failed (Random Output)
SSS Recovery	User provides t valid shares + 1 forged share.	**SUCCESS** (Error Correction via Reed-Solomon)	✓Key Recovered Correctly
zk-SNARK	Valid proof π generated with correct witness w.	**TRUE**	✓On-Chain Verification Passed
zk-SNARK	Forged proof π' (Replay of old nullifier n).	**FAIL** (Double-Spend Protection)	✓Contract Reverted ("Nullifier Used")
Policy Engine	Risk $R = 0.7$ (High), Transaction Value \$6000.	**BLOCK**	✓Tx Rejected locally

This verification confirms that the experimental results are not merely "successful" runs, but strictly adhere to the cryptographic hardness assumptions (scheme sufficiency). Specifically, the failure of the SSS module to reconstruct the key with $t-1$ shares validates that the implementation correctly enforces the threshold polynomial properties defined in Eq. (3).

Conclusion

This study presented the CAPPR-Wallet, a modular self-custody architecture designed to reconcile the conflicting requirements of security, privacy, and usability in decentralized ecosystems. By unifying an on-device Context-Aware Risk Engine with a

Dual-Mode Decentralized Recovery mechanism, we demonstrated that wallet security can be dynamic rather than static. Our experimental evaluation confirms that the proposed system achieves a 93.6% accuracy in detecting transaction anomalies while maintaining a low false-positive rate. Crucially, the integration of a zk-SNARK-based privacy layer reduces the probability of address linkability to near-zero ($< 5\%$) with a manageable gas overhead feasible for L2 and high-value L1 transactions. Furthermore, the trustless recovery protocol eliminates the single point of failure inherent in seed phrases, offering users a viable restoration path in under 180 seconds without relying on centralized custodians. We acknowledge that our results are based on an emulated environment and synthetic datasets. While this verifies the cryptographic and logical soundness of the architecture, real-world deployment faces challenges regarding Mainnet gas volatility and the stochastic nature of human behavior, which may require an initial learning period for the risk engine. Future work will focus on three key areas: (1) deploying the system on an incentivized testnet to gather adversarial telemetry; (2) optimizing client-side zero-knowledge proof generation using hardware acceleration (WebGPU); and (3) expanding the interoperability layer to support non-EVM privacy standards such as the IBC protocol.

References

1. Elmougy, O. & Liu, J. Detecting fraudulent behavior in bitcoin transactions using machine learning. *J. Financial Cryptogr.* **28**, 45–62 (2023).
2. Marcus, R. & Kim, L. Railgun: Practical privacy for decentralized finance. *Blockchain Res. Horizons* **5**, 45–59 (2023).
3. Labs, P. Polygon zkevm documentation. https://polygon.technology (2023).
4. Karimian, M., Zhao, F. & Gupta, Y. Graph neural network-based anomaly detection in transaction flows. In *IEEE Big Data Conference* (2023).
5. Fereidouni, P., Chen, L. & Smith, B. Federated risk prediction models for adaptive mobile authentication. *IEEE Transactions on Mob. Comput.* **23**, 1234–1249 (2024).
6. Singh, R., Kumar, A. & Johnson, M. Rad-aa: Real-time risk adaptive authentication for enterprise systems. In *Proceedings of the 32nd ACM Conference on Computer and Communications Security* (2023).
7. Nair, V. & Song, Y. Multi-factor key derivation without custodian support. *ACM Transactions on Priv. Secur.* **26**, 10–28 (2023).
8. Alliance, D. The derec alliance: Standards for decentralized key recovery. https://derec.org (2024).
9. Cruz, E., Junior, J., Souza, Y., Jesus, G. & Peixoto, M. Archw3: An adaptive blockchain wallet architecture for web3 applications. *Comput. Networks* **262**, 111182 (2025).
10. Konkin, A. & Zapechnikov, P. Enterprise considerations for zk-snark deployments: A survey. *IEEE Secur. Priv. Mag.* **21**, 58–66 (2023).
11. Labs, S. Solana zkvm designs. https://solana.com/blog/zkvm (2024).
12. Liang, T., Nguyen, M. & Perez, R. A systematization of zk-snark toolchains for scalable privacy-preserving computations. *Cryptol. Surv. Rev.* **2**, 1–25 (2025).
13. Guo, F., Lee, J. & Tan, S. zk-apc: Zero-knowledge anonymous payment channels. In *IEEE Symposium on Security and Privacy (SP)* (2024).
14. Zhang, Y., Wang, H. & Li, J. Smart contract verified threshold recovery protocols. *IEEE Transactions on Dependable Secur. Comput.* **21**, 114–129 (2024).
15. Hertz, D. & Kim, S. Homomorphic encryption for stealth address privacy. *J. Cryptogr. Eng.* **13**, 200–218 (2023).
16. Elloumi, A., Martinez, F. & Lee, P. Adaptive trust control in vanets via smart contracts. In *International Conference on Connected Vehicles and Expo* (2023).
17. Islam, M. & Gupta, R. Thresholded smart contracts for secure log access. *IEEE Transactions on Inf. Forensics Secur.* **18**, 150–164 (2023).
18. Homoliak, I. & Perešíni, M. Sok: Cryptocurrency wallets—a security review and classification based on authentication factors. *arXiv preprint arXiv:2402.17659* (2024).
19. Chalkias, K. K., Maram, D., Roy, A., Wang, J. & Yadav, A. Zero-knowledge authenticator for blockchain: Policy-private and obliviously updateable. Cryptology ePrint Archive, Paper 2025/921 (2025).
20. Abo Alzahab, N., Rafaiani, G., Battaglioni, M., Chiaraluce, F. & Baldi, M. Decentralized biometric authentication based on fuzzy commitments and blockchain. In *arXiv preprint arXiv:2409.11303* (2024).

21. Baldimtsi, F., Lazarus, E. & Patel, S. zklogin: Passwordless and privacy-preserving authentication via web2 credentials and zk-snarks. In *IEEE Symposium on Security and Privacy (SP)* (2024).

22. Team, E. D. Policy-defined recovery workflows in permissioned blockchains. Technical Report, EvoChain Consortium (2024). Includes revocation and restoration mechanisms.

23. Kethepalli, S., Huang, Z. & Menezes, A. Post-quantum threshold recovery with zero-knowledge verification. *Cryptol. ePrint Arch.* **2023** (2023).

24. Chaudhary, A., Gajera, M. & Sahu, N. zkfi: Privacy-preserving and regulation compliant transactions using zero knowledge proofs. *arXiv preprint arXiv:2307.00521v5* DOI: 10.48550/arXiv.2307.00521 (2025).

25. Liu, Y., Chen, X. & Wang, Z. Collaborative zero-knowledge proof generation via client-side parallelism. *Transactions on Cryptogr. Hardw. Embed. Syst.* **2024**, 67–82 (2024).

26. Korzin, A., Choi, K. & Patel, N. Zk-enabled contextual access control for iot devices. In *ACM IoT Security Workshop* (2023).

27. Bappy, F. H., Park, J. S., Hasan, K. & Islam, T. Chainguard: A blockchain-based authentication and access control scheme for distributed networks. In *arXiv preprint arXiv:2412.00677* (2024).

28. Patwe, S. & Mane, S. B. Blockchain-enabled secure and interoperable authentication scheme for metaverse environments. *Futur. Internet* **16**, 166, DOI: 10.3390/fi16050166 (2024).

29. zkSync Team. zksync era launch overview. https://zksync.io (2023).

30. Jadhav, R., Kim, H. & Torres, M. Hybrid aes/ecc encryption for ipfs content security. *Decentralized Storage J.* **2**, 101–119 (2024).

31. Zhou, H., Kumar, S. & Roberts, A. n-party virtual payments: Design and formalization. In *Proceedings of the 2021 ACM Workshop on Blockchain Security*, 45–58 (2021).

32. Park, Y. & Li, J. Virtual payment objects for multi-recipient atomic transfers. In *IEEE International Conference on Blockchain (BCON) 2022*, 88–101 (2022).

33. Wang, L., Gupta, S. & Thomas, R. Concurrent execution models for high-throughput blockchains. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI) 2022*, 123–138 (2022).

34. Fernandez, M. & Zhao, Q. Speculative execution and deterministic ordering for on-chain concurrency. *J. Distributed Ledger Res.* **6**, 77–95 (2023).

35. Nguyen, T., Patel, R. & Singh, A. Multi-party payment channels: Models and evaluation. *Proc. Financial Cryptogr. Work. 2020* 200–217 (2020).

36. Hsu, K. & Ramesh, P. Scalable multi-party state channels for off-chain settlement. In *ACM Symposium on Applied Computing 2021*, 320–332 (2021).

37. Maxwell, P., Poelstra, A. & Wuille, P. Musig: A schnorr-based multi-signature scheme and its applications. *Cryptogr. Protoc. Appl.* **2**, 12–27 (2020).

38. Lee, Y., Chen, D. & Kumar, S. Eas: Efficient ecc aggregate signatures without pairings. In *ACM Conference on Computer and Communications Security (CCS) Workshop on Applied Cryptography, 2023*, 10–22 (2023).

39. Tanaka, H. & Oliveira, L. Security of aggregate signatures in concurrent signing models. *J. Cryptogr. Eng.* **13**, 145–164 (2023).

40. Oliveira, L., Mehta, P. & Zhang, Y. Rogue-key resilience and nonce-binding for robust aggregate signatures. *IEEE Transactions on Inf. Forensics Secur.* **19**, 233–250 (2024).

41. zkPay Team. zkpay: Prototype notes on private multi-recipient payments. Tech. Rep., zkPay Labs (2023). Technical report and benchmarks, available as project whitepaper.

42. Team, C. Chainguard: Context-aware access & aggregation for blockchains. Tech. Rep., ChainGuard Foundation (2024). Implementation notes and benchmark suite.

Data availability

The data will be made available on request from the corresponding author Ali Muqtadir (alimuqtadir@ncepu.edu.cn).

Conflicts of interest

The authors declare no known financial or non-financial conflicts of interest.

Author contributions statement

ML: methodology; investigation; validation; visualization; writing-review and editing; datacuration. HL: methodology; supervision; project management; funding acquisition; resources. AM: conceptualization; software; coding; writing—orginial draft; writing-review and editing; methodology. RO: Conceptualization; supervision; writing—review and editing; project administration. MFS: datacuration; investigation; validation.

Funding

This research was supported by the General Project of Hebei Law Society 2025 "Intellectual Property Dilemmas and Legal Safeguards for AI-Generated Intangible Cultural Heritage Content" (Grant No. HBF25B035).