Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

ZKP Banking Interface: A Privacy-Preserving, Context-Aware Authentication System Using Zero-Knowledge Proofs

Mr. Utsav Pandey, Ms. Sania Parkar, Mr. Anant Sarode, Prof. Amit Chakrawarti
Department of Artificial Intelligence & Machine Learning
Dilkap Research Institute of Engineering & Management Studies
Mamdapur, Neral, Karjat, Raigad – 410101, Maharashtra, India

***Abstract*- Traditional banking systems rely on passwords, PINs, and centralized biometric storage, exposing users to phishing, SIM-swap attacks, and credential reuse vulnerabilities. This paper presents the ZKP Banking Interface, a pinless, context-aware financial authentication system built on Zero-Knowledge Proof (ZKP) technology.**

A unified ZK-SNARK circuit (Groth16), implemented in Circom, encodes three simultaneous security factors: identity commitment, trusted network context, and transaction threshold logic using Poseidon hashing. Conditional biometric and puzzle challenges are enforced cryptographically for higher-risk scenarios.

Proof generation and verification are fully automated within the browser using snarkJS and WebAssembly artifacts, eliminating manual proof handling. The system supports two verification modes: local (device-only privacy) and on-chain (public auditability via a Solidity smart contract on a Hardhat network).

Experimental evaluation across five test scenarios confirms correct contextual policy enforcement, successful proof validation in both modes, and a smooth, practical banking user experience. The proposed system demonstrates that ZKP-based authentication can replace traditional credentials while maintaining strong privacy guarantees.

***Keywords*- Zero-Knowledge Proofs; Privacy-Preserving Banking; ZK-SNARKs; Groth16; On-chain Verification; Smart Contracts; Circom; Context-Aware Security; Poseidon Hash; snarkJS**

I. INTRODUCTION

Financial applications increasingly require authentication systems that are both secure and frictionless. Conventional methods such as passwords, PINs, OTPs, and static biometrics expose users to phishing, replay attacks, and credential theft, while simultaneously degrading user experience [1], [2].

Zero-Knowledge Proofs (ZKPs) offer a cryptographic alternative that allows a user to prove possession of a secret without revealing the secret itself. Applied to banking, ZKPs can enable a pinless, context-aware interface that verifies identity and transaction authorization while fully preserving user privacy.

The proposed ZKP Banking Interface combines three independent security signals: identity, network context, and transaction threshold into a single Groth16 proof. The circuit, implemented in Circom [3], uses Poseidon hashing to bind private inputs to public commitments. Private values never leave the device; only cryptographic proofs are transmitted.

Conditional logic within the circuit enforces adaptive security: untrusted networks trigger biometric verification; high-value transactions require a puzzle challenge. This yields a layered security model without adding friction to low-risk transactions.

The system operates in two verification modes: (i) local mode, where the proof is verified in the client browser using snarkJS [4] with no blockchain dependency, and (ii) on-chain mode, where the proof and public inputs are submitted to a Solidity smart contract for verifiable, public auditability.

II. SYSTEM ARCHITECTURE

The ZKP Banking Interface comprises three cooperating layers, as shown in Fig. 1. The Frontend Interface (React + TypeScript) handles proof generation, local secret storage, and conditional authentication prompts.

The ZKP Circuit (Circom, Groth16) encodes identity commitment, trusted network context, and transaction threshold policy as cryptographic constraints. The Smart Contract (Solidity, Hardhat) stores public commitments and verifies proofs on-chain when the user selects on-chain verification mode.

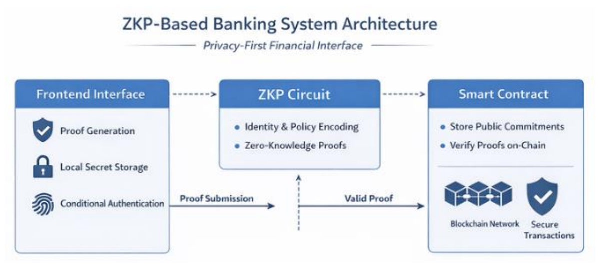

Fig. 1. ZKP-Based Banking System Architecture

The core privacy guarantee is that private inputs : identity secret, biometric phrase, puzzle answer, and network ID never leave the user's device.

Only Poseidon hash commitments and the proof tuple (a, b, c) are ever transmitted, whether to localStorage in local mode or to the blockchain in on-chain mode.

A. Technical Component Interaction

Fig. 2 shows the detailed interaction between components during a proof lifecycle. The user triggers a transaction from the frontend. The Circom circuit processes private and public inputs via snarkJS and WASM artifacts to produce a Groth16 proof.

In local mode, the proof is verified in-browser. In on-chain mode, the proof and public signals are submitted via MetaMask to the ZKPBank smart contract on the Hardhat network, which validates the proof and emits a TransactionAuthorized event.

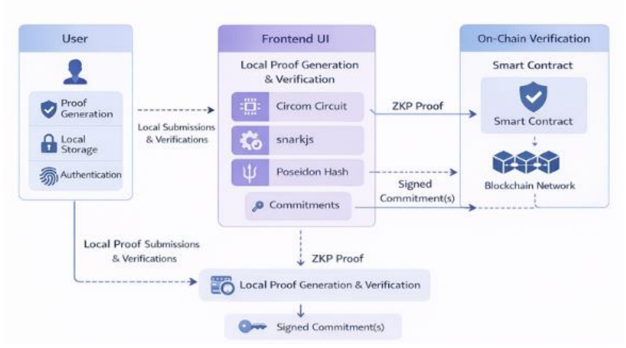

Fig. 2. ZKP Banking Interface — Technical Architecture

III. AUTHENTICATION FLOWCHART

The flowchart in Fig. 3 captures every decision point in the ZKP authentication pipeline.

It shows the two orthogonal risk signals evaluated simultaneously: network trust status and transaction amount versus the user-defined spending threshold.

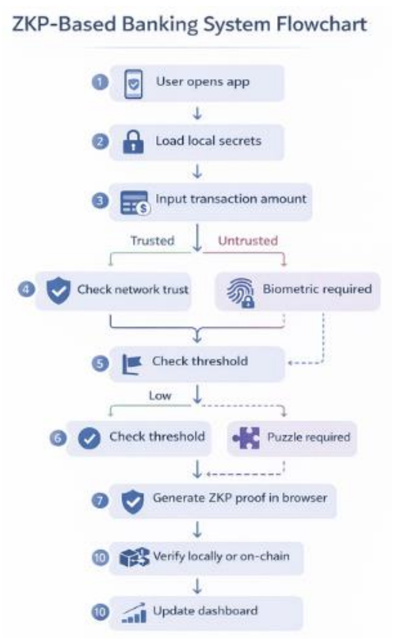

Fig. 3. ZKP-Based Banking System — 10-Step Flowchart

A. Flow Description

Step 1- App Open: The frontend loads and checks localStorage for existing context.

Step 2- Load Secrets: Identity secret, trusted network ID, biometric commitment, and puzzle commitment are loaded. If absent, the user is sent to the Security Center for enrollment.
Step 3- Input Amount: User enters transaction amount and recipient address.

Step 4- Network Check: Current network ID is hashed and compared to the trusted network hash.

Step 5- Biometric Prompt: If network is untrusted, the user must supply their biometric passphrase; the frontend hashes it and passes it as a private circuit input.

Step 6- Threshold Check: Amount is compared to the user-defined spending threshold.

Step 7- Puzzle Prompt: If amount exceeds threshold, the puzzle challenge is required.

Step 8- Generate Proof: snarkjs.groth16.fullProve() is invoked in-browser using WASM and ZKey artifacts.

Step 9- Verify Proof: Local mode verifies in-browser; on-chain mode submits proof to ZKPBank via MetaMask.

Step 10- Update Dashboard: Context Halo, balance, and activity log are updated with result.

IV. UI SCREENS

The interface consists of seven screens across three navigation tabs: Dashboard, Activity, and Security Center. A Pre-Transaction Setup modal handles first-time onboarding. Screens are documented below in the order a user encounters them.

A. Screen 1 — Pre-Transaction Setup

The enrollment modal (Fig. 4) is shown automatically on first launch or when any required context is missing. It captures: (1) Wallet Connection via MetaMask for on-chain mode;

(2) Trusted Network ID, the current network's simulated BSSID hash, scanned via Rescan Network;

(3) Biometric Enrollment a user passphrase that is keccak256-hashed locally and stored as biometricCommitment; the raw passphrase is discarded immediately. Status badges (Wallet / Network / Biometric / Puzzle / Threshold / Crypto) must all be active before the user can continue to the dashboard. The error state Blocking: wallet is shown when MetaMask is not yet connected.

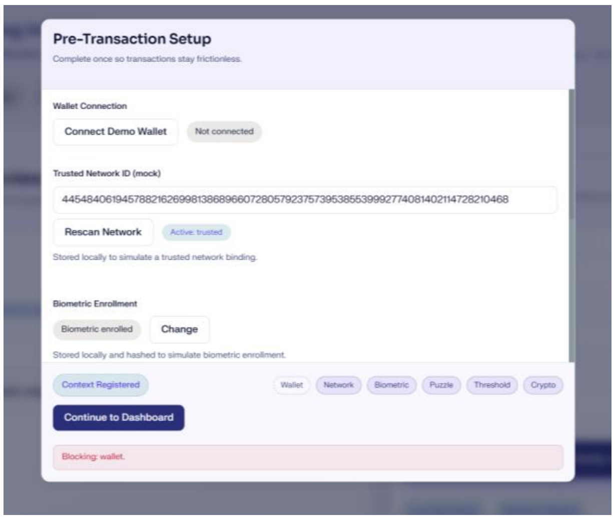
Pre-Transaction Setup
Complete once so transactions stay frictionless.
Wallet Connection: Not connected
Trusted Network ID (mock): 44548406194578821626998138689660728057923757395385539992774081402114728210468
Active: trusted
Stored locally to simulate a trusted network binding.
Biometric Enrollment: Biometric enrolled
Stored locally and hashed to simulate biometric enrollment.
Context Registered
Blocking: wallet.

Fig. 4. Screen 1 — Pre-Transaction Setup (Enrollment Modal)

B. Screen 2 — Dashboard Overview

The main dashboard (Fig. 5) is the primary interface for initiating transactions. It contains four panels.

The Balance Overview (top left) shows on-chain ETH balance with mode, network, and threshold badges.

The Send Funds panel (top right) accepts verification mode selection, recipient address, and transaction amount; the Generate Proof and Execute button triggers the full proof pipeline.

The Context Halo (bottom left) is a large circular visual indicator — blue for TRUSTED, orange for UNTRUSTED — that provides immediate security context feedback.

The Account Status panel (bottom right) shows wallet address, biometric status, active network label, and wallet connection state.

The Secure Context Ready badge in the header confirms all five commitments are registered and ready.

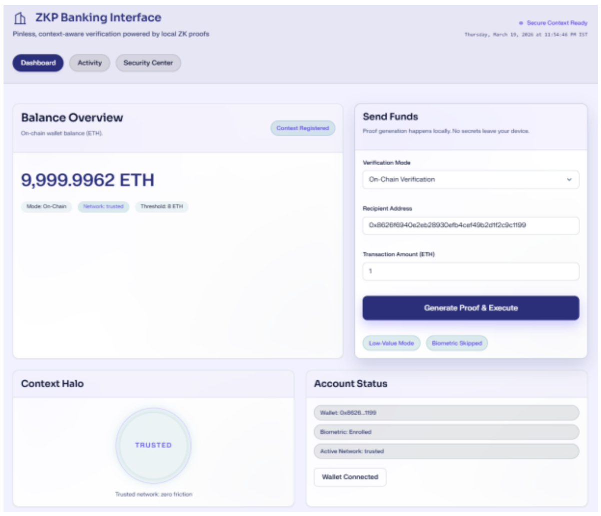
ZKP Banking Interface
Pinless, context-aware verification powered by local ZK proofs
Secure Context Ready
Thursday, March 19, 2026 at 11:54:46 PM IST
Balance Overview
On-chain wallet balance (ETH).
Context Registered
9,999.9962 ETH
Mode: On-Chain · Network: trusted · Threshold: 8 ETH
Send Funds
Proof generation happens locally. No secrets leave your device.
Verification Mode: On-Chain Verification
Recipient Address: 0x8626f6940e2eb28930efb4cef49b2d1f2c9c1199
Transaction Amount (ETH): 1
Low-Value Mode · Biometric Skipped
Context Halo
TRUSTED
Trusted network: zero friction
Account Status
Wallet: 0x8626...1199
Biometric: Enrolled
Active Network: trusted
Wallet Connected

Fig. 5. Screen 2 — Dashboard (Balance, Send Funds, Context Halo)

C. Screen 3 — Verification Mode Selection

The Send Funds panel (Fig. 6) exposes a Verification Mode dropdown with two options.

Local Proof (no blockchain) verifies entirely in-browser, requires no wallet, and incurs no gas cost this mode demonstrates pure device-side privacy.

On-Chain Verification submits the proof to the ZKPBank smart contract, providing public auditability. Both modes use the identical Circom circuit; only the verification destination differs. When a transaction amount exceeding the spending threshold is entered, the status badge below the button immediately updates to High-Value Mode and queues the puzzle prompt.

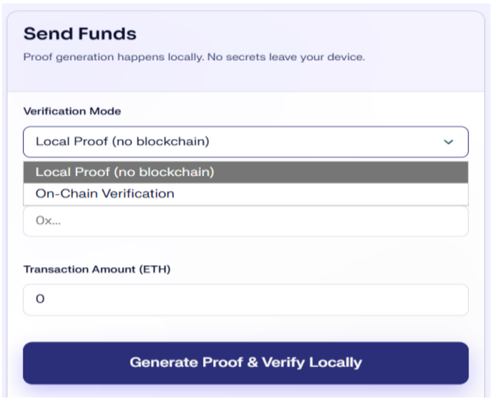
Send Funds
Proof generation happens locally. No secrets leave your device.
Verification Mode: Local Proof (no blockchain)
Local Proof (no blockchain)
On-Chain Verification
0x...
Transaction Amount (ETH): 0

Fig. 6. Screen 3 — Verification Mode Selection Dropdown

D. Screen 4 — MetaMask Wallet Approval

In on-chain mode, after the proof is generated in-browser, MetaMask presents a transaction request (Fig. 7). The request shows: Network = Hardhat Local; Request from localhost:5173; Interacting with the ZKPBank contract address. The network fee (~$0.94 in test ETH) is the gas cost

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

for the on-chain verifyProof call. The proof tuple (a, b, c) and public commitments are what get submitted not the identity secret, biometric phrase, or puzzle answer.

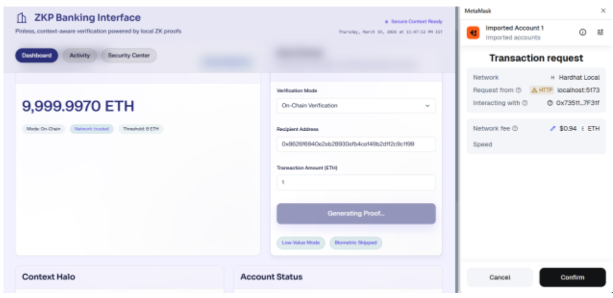

Fig. 7. Screen 4 — MetaMask Wallet Approval for On-Chain Verification

E. Screen 5 — On-Chain Verification Result

After successful on-chain verification (Fig. 8), the UI displays a success toast in the top-right corner: Profile Registered — On-chain context is active, with the full transaction hash for external auditing.

The balance is updated to reflect gas consumed. Dashboard badges confirm Mode: On-Chain, Network: trusted, and the current threshold.

The transaction hash is also written to the Activity log.

If verification fails due to an incorrect biometric or puzzle input, a failure toast is shown and no gas is consumed, since the proof is rejected before submission.

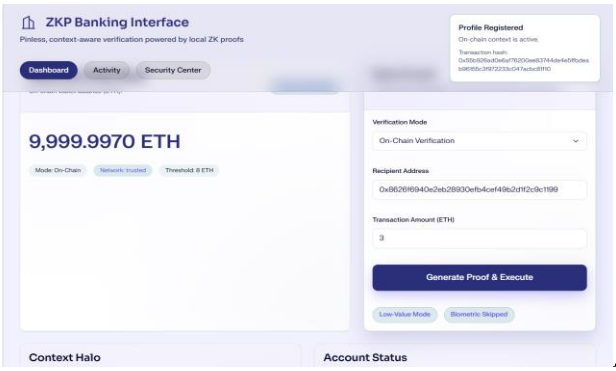

Fig. 8. Screen 5 — On-Chain Verification Result with Transaction Hash

F. Screen 6 — Activity Tab

The Activity tab (Fig. 9) is a read-only audit ledger stored in localStorage. Each entry records the transaction amount (ETH), verification mode (chain or local), network state (trusted or untrusted), and timestamp. On-chain entries include the full transaction hash prefixed On-chain proof verified: 0x... No private data is stored in this log, only public proof receipts and transaction metadata.

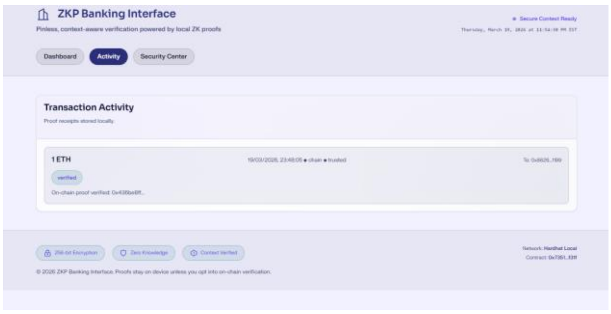

Fig. 9. Screen 6 — Activity Tab (Proof Receipt Ledger)

G. Screen 7 — Security Center

The Security Center (Fig. 10) configures all device-bound secrets and context parameters used by the ZKP circuit.

It allows the user to: switch verification mode; view and rescan the Trusted Network ID; use Network Simulation controls (Force Trusted, Force Untrusted, Auto Flip — which alternates every 90 seconds for live demonstrations); re-enroll the biometric passphrase; and set the spending threshold in ETH.

Register On-Chain Context calls ZKPBank.registerUser() with all five public commitments.

Update Threshold calls updateSpendingThreshold() to sync the new value on-chain.

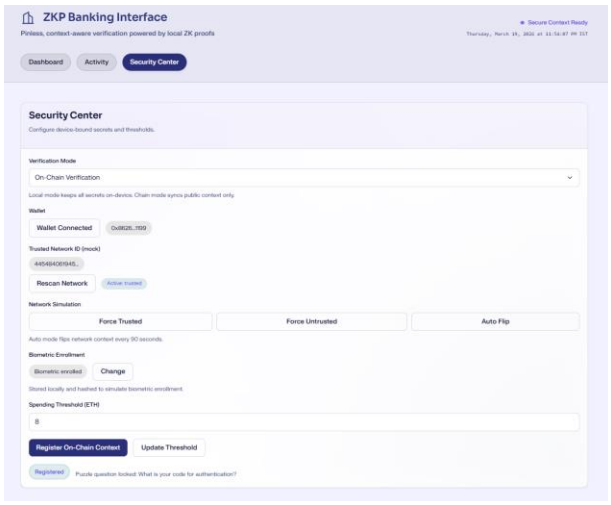

Fig. 10. Screen 7 — Security Center (Context & Threshold Management)

V. SCENARIO VALIDATION

Table I summarises the five test scenarios used to validate contextual policy enforcement. Each scenario tests a distinct combination of network trust, transaction amount, and biometric correctness, confirming that the ZKP circuit enforces the correct adaptive policy independently of any UI-level logic.

TABLE I. SCENARIO VALIDATION MATRIX

ID	Network	Amount	Biometric	Puzzle	Result
A	Trusted	Low	—	—	Success
B	Untrusted	Low	Required	—	Success
C	Trusted	High	—	Required	Success
D	Untrusted	High	Required	Required	Success
E	Untrusted	Low	Wrong	—	Fail

Scenario E confirms that supplying an incorrect biometric passphrase violates the circuit constraint Poseidon(biometricSecret) == biometricCommitment. The snarkJS witness generation fails, no proof is produced, and the activity log records a rejected entry. No blockchain transaction is submitted and no gas is consumed.

VI. DATA STORAGE AND PRIVACY REFERENCE

Table II maps every data item in the system to its storage location and whether it is ever transmitted. This table provides the evidential basis for the privacy claims made in the primary research paper.

TABLE II. DATA STORAGE AND TRANSMISSION MODEL

Data Item	Stored Where	Transmitted?
Identity Secret	localStorage	Never
Network ID	localStorage	Never
Biometric Secret	localStorage	Never
Puzzle Secret	Runtime only	Never
Transaction Amount	Runtime only	Never
Identity Commitment	On-chain / local	On registration
Trusted Network Hash	On-chain / local	On registration
Biometric Commitment	On-chain / local	On registration
Spending Threshold	On-chain / local	On registration
Proof (a, b, c)	Not stored	On-chain mode only
Transaction Hash	Activity log	Display only

VII. UI COMPONENT GLOSSARY

Table III lists all named UI components, their function, and the security behaviour they surface to the user.

TABLE III. UI COMPONENT GLOSSARY

Component	Description
Context Halo	Blue = trusted network (no biometric required). Orange = untrusted (biometric required). Updates in real-time.
Generate Proof & Execute	Triggers the full proof pipeline: input collection, snarkjs.groth16.fullProve(), then verification.
Low-Value Mode badge	Amount ≤ threshold. No puzzle required. Biometric may still apply if network is untrusted.
High-Value Mode badge	Amount > threshold. Puzzle challenge required before proof generation proceeds.
Biometric Skipped	Trusted network — biometric prompt bypassed by circuit conditional constraint.
Biometric Required	Untrusted network — biometric passphrase must be entered; wrong input causes proof failure.
Secure Context Ready	All five commitments (identity, network, biometric, puzzle, threshold) are registered.
Rescan Network	Re-reads the current simulated BSSID and updates trusted network hash in localStorage.
Register On-Chain Context	Calls ZKPBank.registerUser() with all commitments — required once before on-chain mode.
Force Trusted/Untrusted	Security Center simulation controls for demo purposes.
Auto Flip	Alternates network trust state every 90 seconds — used in live demos of adaptive biometric triggering.

VIII. CONCLUSION

This paper presented the ZKP Banking Interface, a pinless, context-aware banking authentication system built on Zero-Knowledge Proofs. The system verifies identity and transaction authorization without exposing private data, combining identity commitments, trusted network context, biometric confirmation, and transaction threshold logic in a single Groth16 ZK-SNARK circuit. Proof generation and verification are fully automated in the browser, and dual verification modes offer both privacy-first local operation and publicly auditable on-chain validation.

Experimental results confirmed correct enforcement of all contextual security policies across five test scenarios, successful proof validation in both modes, and a smooth, intuitive user experience. The system demonstrates that ZKP-based authentication is a practical and viable alternative to traditional PIN/OTP workflows for financial applications.

Future work directions include: (1) integration of AI-driven behavioral biometrics—typing rhythm, mouse dynamics, and interaction patterns—to generate additional trust signals;

(2) hardware secure enclave support for storing secrets with device attestation; (3) Layer-2 deployment for reduced gas costs and faster on-chain confirmation; (4) real biometric API integration with liveness detection; and (5) federated learning models for behavioral baselines that preserve user privacy. These enhancements would advance the system toward a production-ready, multi-modal, privacy-preserving financial authentication platform.

ACKNOWLEDGMENT

We sincerely thank Prof. Akshata Laddha (Project Guide) and Prof. Amit Chakrawarti (Head of Department, AI & ML) for their continuous guidance and support. Gratitude is also extended to the developers of Circom, snarkJS, and the Ethereum ecosystem for providing the foundational open-source technologies that made this work possible.

REFERENCES

[1] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, Handbook of Applied Cryptography. Boca Raton, FL: CRC Press, 1996.

[2] NIST, "Digital Identity Guidelines," NIST Special Publication 800-63B, Jun. 2017.

[3] iden3, "Circom: A Circuit Compiler for ZK-SNARKs," GitHub Repository, 2022. [Online]. Available: https://github.com/iden3/circom

[4] iden3, "snarkJS: JavaScript ZK-SNARK Library," GitHub Repository, 2022. [Online]. Available: https://github.com/iden3/snarkjs

[5] A. K. Jain, A. Ross, and S. Prabhakar, "An introduction to biometric recognition," IEEE Trans. Circuits Syst. Video Technol., vol. 14, no. 1, pp. 4–20, Jan. 2004.

[6] A. Das and N. Borisov, "Tracking the trackers: Measuring intrusiveness of web tracking," Proc. NDSS, 2019.

[7] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[8] J. Groth, "On the size of pairing-based non-interactive arguments," in Proc. Eurocrypt, 2016, pp. 305–326.

[9] E. Ben-Sasson et al., "Zerocash: Decentralized anonymous payments from Bitcoin," in Proc. IEEE Symp. Security and Privacy, 2014, pp. 459–474.

[10] C. Garman, M. Green, and I. Miers, "Accountable privacy for decentralized anonymous payments," in Proc. Financial Cryptography, 2015.

[11] D. Boneh, "Graduate Course Notes on Cryptography," Stanford Univ., 2020.

[12] M. Al-Fuqaha et al., "Context-aware security in IoT and smart systems," IEEE Commun. Surveys Tuts., vol. 20, no. 3, pp. 2114–2132, 2018.