A privacy-compliant Ethereum mixer: Prototype of a financial transaction anonymization tool using zk-SNARKs and ZK-KYC

Farold H. Adoukonou*[1,*,‡], Eugène C. Ezin[1,2,§]

[1]*Institut de Formation et de Recherche en Informatique (IFRI), Université d'Abomey-Calavi, Bénin*
[2]*Institut de Mathématiques et de Sciences Physiques (IMSP), Université d'Abomey-Calavi, Bénin*

Abstract
The emergence of cryptocurrencies and blockchain technology has transformed financial transactions by enhancing efficiency and security through decentralization. However, public blockchains like Ethereum present privacy challenges, as all transactions are transparent and easily accessible. This study explores the development of a prototype tool designed to anonymize transactions on the Ethereum blockchain by leveraging Zero-Knowledge Proofs with Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge (zk-SNARKs) and a Zero-Knowledge Know Your Customer (zk-KYC) system to balance privacy with regulatory compliance. The findings demonstrate that it is possible to enhance transaction privacy on Ethereum while effectively adhering to Know Your Customer and Anti-Money Laundering regulations (KYC/AML), providing a solution that protects user anonymity without compromising legal requirements.

Keywords
Ethereum Blockchain, Anonymization, zk-SNARKs, zk-KYC, KYC/AML Compliance

1. Introduction

The Ethereum blockchain has established itself as a major platform for decentralized financial transactions and smart contract execution, offering transparency, security, and immutability. However, the transparency of public blockchains like Ethereum poses many challenges regarding confidentiality and personal data protection. Every transaction on Ethereum is publicly visible, allowing anyone to view sender and recipient addresses, transferred amounts, and the full transaction history, thus exposing users to risks of privacy infringement, deanonymization, and surveillance.

In parallel, international financial regulations, such as Know Your Customer (KYC) and Anti-Money Laundering (AML), impose strict identity verification and suspicious transaction monitoring obligations on institutions in the financial sector to prevent money laundering and terrorist financing. These requirements often conflict with users' desire to maintain their anonymity on decentralized platforms. The purpose of this study is therefore to design a technical solution that ensures transaction anonymity without compromising blockchain security and transparency while integrating regulatory compliance mechanisms.

This article proposes the development of a prototype for a financial transaction anonymization tool on the Ethereum blockchain, Phantom ETH, which aims to reconcile transaction anonymity with regulatory compliance through the use of Zero-Knowledge Proofs (ZKP), and more specifically zk-SNARKs, with a Zero-Knowledge KYC (ZK-KYC) system.

*Corresponding author.
farold.adoukonousagbadja@uac.bj (F. H. Adoukonou) ; eugene.ezin@uac.bj (E. C. Ezin)
0000-0002-7850-8600 (E. C. Ezin)

2. Context and state of the art

2.1. Pseudonymity vs. real anonymity on the blockchain

The inherent transparency of public blockchains like Ethereum poses significant privacy challenges, as all transactions are publicly visible, revealing sender and recipient addresses, amounts, and full transaction history [1]. This exposes users to risks of deanonymization, surveillance, and privacy infringement.

It is crucial to distinguish between pseudonymity and real anonymity. On Ethereum, users are identified by public keys (e.g., `0x294d776eBbEfe4F989005936716066BC551Fc2ed`) that do not directly contain personal information. This is pseudonymity. However, analysis techniques such as Open-Source Intelligence (OSINT) [2], which involve collecting and analyzing publicly available information, and on-chain and off-chain analysis [3] can link these addresses to real individuals. On-chain analysis refers to data stored directly on the blockchain, while off-chain analysis evaluates transactions and transfers occurring outside the blockchain.

Real anonymity, on the other hand, aims to ensure that no identifiable information can be linked to a transaction or address. This requires sophisticated mechanisms, such as data obfuscation or the use of advanced cryptographic protocols like Zero-Knowledge Proofs (ZKP). Cryptocurrencies like Monero and Zcash are specifically designed to mask identities and transaction amounts [4].

2.2. KYC/AML regulations

Know Your Customer (KYC) [5] and Anti-Money Laundering (AML) [6] regulations are fundamental for the security of financial systems worldwide, aiming to prevent fraud, combat money laundering, and counter terrorist financing. The growing popularity of cryptocurrencies, with their pseudo-anonymity and cross-border nature, has complicated the implementation of these regulations. Recent reports indicate a significant increase in losses related to cryptocurrency scams, and tools like mixers are widely used by criminals to obscure the origin of illicit funds [7]. Regulatory authorities, such as the Financial Crimes Enforcement Network (FinCEN) [8] in the United States and the 5AMLD [9] in the European Union, have strengthened sanctions against the well-known mixer Tornado Cash involved in illicit activities [10].

2.3. Anonymization techniques

2.3.1. Cryptocurrency mixers

Cryptocurrency mixers, also known as tumblers, hide the origin and destination of transactions by mixing funds from multiple users in a common pool (e.g., a pool that only takes 5 ETH or 10 ETH), which breaks the direct link between original and destination addresses [11]. Techniques such as amount splitting and adding random delays enhance anonymity.

A key element to understanding the importance of cryptocurrency mixers lies in the pseudonymous nature of most blockchains. On a public blockchain like Bitcoin and Ethereum, every transaction and every wallet address are publicly visible, allowing anyone to access them. This transparency, while advantageous for traceability, limits the complete anonymity of users, hence the need for tools like mixers to enhance transaction privacy.

However, their misuse by malicious actors for money laundering has raised many regulatory concerns. Examples of illegal use of mixers include:

- North Korea, whose Lazarus Group is considered the largest hacker group, was responsible for stealing $1.7 billion in cryptocurrencies in 2022 [12]. Of this, $455 million was diverted through Tornado Cash, a cryptocurrency mixer.
- ChipMixer, an unlicensed cryptocurrency mixer created in mid-2017, specialized in mixing cryptocurrencies, authorities said. Europol described it as "one of the largest darkweb cryptocurrency laundries" and stated that over €40 million ($42.2 million) in cryptocurrencies had been seized.

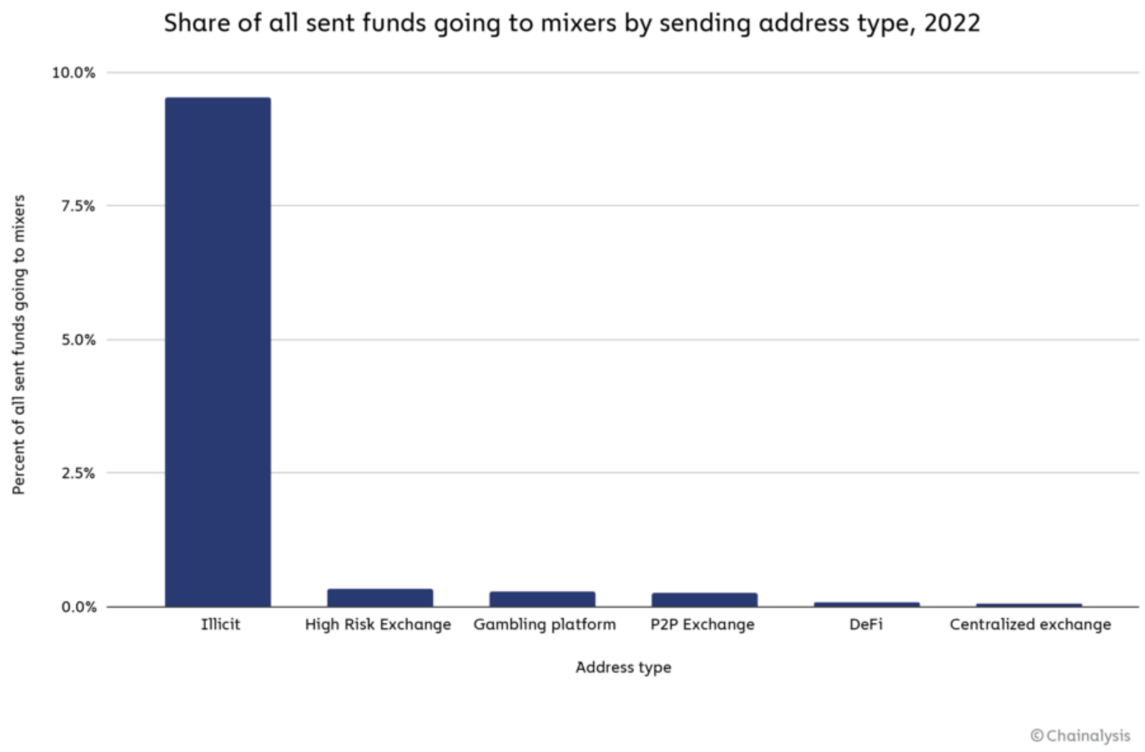

Figure 1: Distribution of funds sent to mixers in 2022 (https://www.chainalysis.com/blog/crypto-mixer-criminal-volume-2022/)

Figure 1 shows that a large portion of funds deposited into mixers comes from illicit sources.

2.3.2. Zero-knowledge proofs (ZKP)

Zero-Knowledge Proofs (ZKP) are protocols that allow one party (the prover) to prove to another party (the verifier) the accuracy of a statement without revealing any information other than the validity of the statement itself [13]. ZKPs are crucial for sensitive information, for example, proving ownership of ethers without revealing the signature or public address. The fundamental principles of ZKP are:

- **Zero-knowledge:** No information about confidential data is revealed.
- **Completeness:** An honest prover can always convince the verifier if the statement is true.
- **Soundness:** A dishonest prover cannot deceive the verifier if the statement is false.

There are interactive and non-interactive ZKP. zk-SNARKs (Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge) are particularly relevant for blockchain because they are succinct (short proofs) and non-interactive (verification without constant dialogue).

2.3.3. Merkle trees

Merkle Trees (hash trees) are essential in blockchain technology for secure and efficient data verification. They were developed by Ralph Merkle in the 1980s [14]. Each leaf of the tree represents a data block, and each internal node contains the cryptographic hash of the concatenation of its children. The Merkle Root summarizes all the data and allows integrity verification without individually analyzing each transaction. They are used in Bitcoin to efficiently store transactions and allow for quick verification [15].

2.3.4. Incremental Merkle trees

While traditional Merkle trees offer an efficient way to store deposit hashes, when there are a large number of deposits, the gas cost would be significant because the entire tree would have to be rebuilt for each deposit to obtain a new Merkle root. The incremental Merkle Tree solves these problems through optimizations that allow new leaves to be added dynamically without rebuilding the entire tree [16]. In this context, it is used to dynamically add new deposits while maintaining a proof of their inclusion in the tree. Below is its operating principle :

- **Initialization:** When the tree is created, its leaves are filled with 'zeros' that serve as placeholders for future deposits.
- **Adding a Deposit:** When a deposit is added to the tree, the zero values are replaced one by one, from the leftmost leaf to the rightmost. The tree is updated efficiently without recalculating the entire tree.

```
uint256 public constant TREE_HEIGHT = 20;
uint256 public constant ZERO_VALUE = 0;

constructor() {
        zeros = new uint256[](TREE_HEIGHT);
        filledSubtrees = new uint256[](TREE_HEIGHT);

        zeros[0] = ZERO_VALUE;

        for (uint32 i = 1; i < TREE_HEIGHT; i++) {
            zeros[i] = hashPair(zeros[i - 1], zeros[i - 1]);
            filledSubtrees[i] = zeros[i];
        }
        roots[0] = hashPair(zeros[TREE_HEIGHT - 1], zeros[TREE_HEIGHT - 1]);
    }

    function hashPair(
        uint256 left,
        uint256 right
    ) public pure returns (uint256) {
        return uint256(keccak256(abi.encodePacked(left, right)));
    }
```

2.4. Case study: Tornado cash

Tornado Cash is a decentralized and open-source cryptocurrency mixing protocol launched in 2019 on Ethereum. It aims to enhance transaction privacy and anonymity on Ethereum by allowing users to deposit cryptocurrencies from one address and withdraw them to another with no traceable link between them.

Tornado Cash operates on the mixing principle: multiple users send their funds to a common smart contract, where deposits are mixed to mask their origin. Although deposits and withdrawals on the smart contract are public, the link between the deposit address and the withdrawal address is broken, provided there are enough other depositors and withdrawers in the smart contract [17].

Its operation relies on ZKP and a Merkle Tree where deposits are recorded as "commitments" (hashes of two secret numbers: a nullifier and a secret). Upon withdrawal, the user proves they know the pre-image of an existing leaf in the tree without revealing that leaf, and a nullifierHash is used to prevent multiple withdrawals.

However, Tornado Cash has faced significant legal issues due to its use by cybercriminals for money laundering. The U.S. Office of Foreign Asset Control (OFAC) sanctioned Tornado Cash, accusing the service of having laundered over $7 billion in cryptocurrencies since its creation, including $445 million stolen by North Korea's Lazarus Group [10]. Although developers were arrested, a U.S. federal appeals court overturned OFAC's sanction decision in November 2024, ruling the sanctions illegal [18].

3. Methodology and system architecture

The Phantom ETH project is designed to reconcile Ethereum transaction anonymity with current KYC/AML financial regulatory requirements, drawing inspiration from Tornado Cash while integrating a native ZK-KYC system.

3.1. Global architecture

The proposed system is based on a three-interconnected layer architecture:

- **Frontend layer:** Decentralized user interfaces developed with Nuxt.js (based on Vue.js) and interacting with the blockchain via Web3.js. It includes an interface for ZK-KYC and another for the Phantom ETH mixer.
- **Blockchain layer:** Two essential smart contracts deployed on Ethereum (or a simulator like Ganache) and written in Solidity:
  - **The ZK-KYC smart contract (KYCRegistry.sol):** Manages identity verification based on ZK proofs and whitelisting addresses.
  - **The mixer smart contract (Mixer.sol):** Manages ETH deposits and withdrawals, integrating a Merkle Tree for privacy and a nullification scheme. It inherits from MerkleTreeWithHistory.sol and uses the OpenZeppelin library for security.
- **Backend layer:** A dishonest prover cannot trick the verifier if the statement is false.
  - A ZK proof generator (using Circom and SnarkJS) for KYC and withdrawals.
  - A relayer service that acts as an intermediary and anonymously executes withdrawal transactions by masking the request address and covering gas fees for the user.

Figure 2 shows a visual diagram of the system components.

3.2. System workflow

The process unfolds in several key phases:

1. **KYC verification:** The user submits their identification information to a central authority, which generates a zero-knowledge proof. This proof is then used to whitelist their addresses in the KYC smart contract on Ethereum. The kyc_verifier.circom circuit verifies KYC compliance without revealing personal data, by validating that the provided identity data matches a pre-registered hash.
2. **Deposit into the mixer:** The user deposits their funds into one of the Mixer smart contract's pools (fixed denominations such as 1, 10, or 100 ETH). The system generates a unique "commitment" (hash of the nullifier and the secret) using the MiMC Sponge hashing function (optimized for zk-SNARKs). This commitment is then added to the Merkle Tree managed by the MerkleTreeWithHistory.sol smart contract. MerkleTreeWithHistory maintains a history of roots for subsequent verifications.
3. **Generation and submission of ZK proof:** During withdrawal, the user provides their "secret note" (containing the nullifier and the secret) and the system generates a ZK proof. This proof, created by the withdraw.circom circuit, demonstrates that the user holds a valid deposit in the mixer and knows the pre-image of the Merkle Tree leaf, without revealing their identity or the commitment. The merkle_tree.circom circuit is used to verify the membership of a leaf in the Merkle tree.

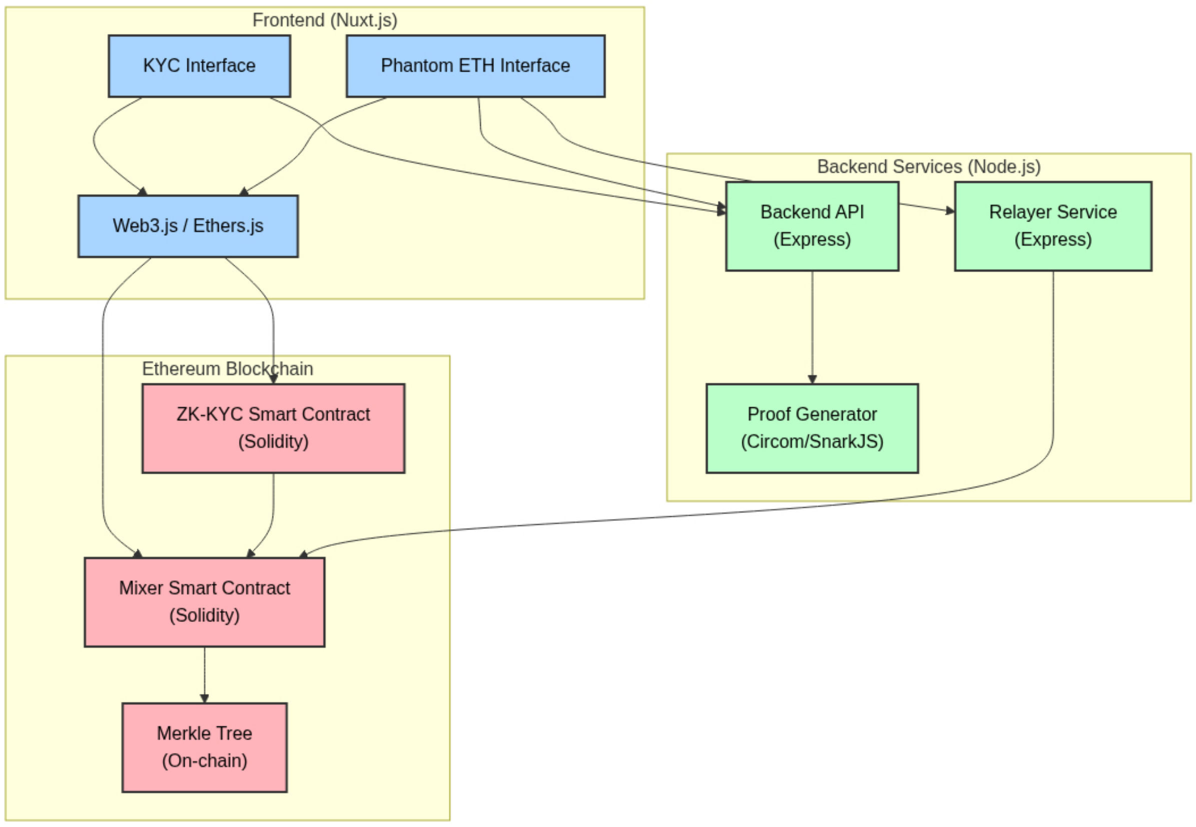
Frontend (Nuxt.js)
KYC Interface
Phantom ETH Interface
Web3.js / Ethers.js
Backend Services (Node.js)
Backend API (Express)
Relayer Service (Express)
Proof Generator (Circom/SnarkJS)
Ethereum Blockchain
ZK-KYC Smart Contract (Solidity)
Mixer Smart Contract (Solidity)
Merkle Tree (On-chain)

Figure 2: System component diagram

4. **Use of the transaction relayer:** During withdrawal, the user provides their "secret note" (containing the nullifier and the secret) and the system generates a ZK proof. The ZK proof and withdrawal information are sent to a relayer who signs the transaction and pays the gas fees, ensuring anonymity by masking the user's withdrawal address.

3.3. The ZK-friendly hashing

For ZK proof operations, the MiMC (Multi-Party Computation friendly) hashing algorithm is used [19], as it is specifically designed to be ‘zk-friendly‘ and more computationally efficient for ZK proof generation than functions like Keccak256.

3.4. The nullifier scheme

To prevent multiple withdrawals while preserving anonymity, the system uses a nullifier mechanism.

Rather than removing leaves from the Merkle tree, which would reveal which deposit is being used, each leaf is constructed from two secret values: a `nullifier` and a `secret`.

The leaf is calculated as the hash of the concatenation of these two values. During a withdrawal, the user submits only the hash of the nullifier (nullifierHash) along with a ZK proof demonstrating that they know the nullifier and the secret corresponding to an existing leaf in the tree. The smart contract stores the used nullifierHashes to prevent their reuse.

```
const _nullifier = "0x" + crypto.randomBytes(31).toString("hex");
const _secret = "0x" + crypto.randomBytes(31).toString("hex");

const { commitmentHash } = await build(_nullifier, _secret, poolAmount);

async function build(_nullifier, _secret, _amount) {
  try {
    const n = BigInt(_nullifier);
    const s = BigInt(_secret);
    const a = BigInt(_amount);
    ...
    const mimcsponge = await circomlibjs.buildMimcSponge();
    const commitment = mimcsponge.multiHash([n, s, a], 0);
    const commitmentHash = mimcsponge.F.toString(commitment);
    ...
}
```

```
mapping(uint256 => bool) public nullifierHashes;
nullifierHashes[nullifierHash] = true;
```

With this approach, only one of the two secret numbers is revealed (via its hash), making it impossible to determine which leaf in the tree is associated with a deposit. This preserves the transaction's anonymity while ensuring that the same deposit can only be withdrawn once.

4. Results

The Phantom ETH prototype was successfully deployed and tested on Ganache, an Ethereum blockchain simulator.

4.1. ZK identity verification

The KYC verification interface allows users to connect their Metamask wallet and submit their personal information to generate a ZK proof. The system validates the proof and allows for whitelisting associated Ethereum addresses. The ability to "cascading whitelist" has been demonstrated, where an already whitelisted address can validate new addresses, thus reducing the need for repeated KYC. Tests confirmed that deposits fail if the sender's address is not on the whitelist.

4.2. Transaction mixer (Phantom ETH)

The Phantom ETH mixer interface is intuitive and allows for ETH deposits and withdrawals. The deposit process involves selecting a fixed amount pool (e.g., 1 ETH, 10 ETH, 100 ETH), generating a secret note, and transferring funds to the smart contract, which records the commitment in the Merkle Tree. During withdrawal, the user enters their secret note and the recipient address. The withdrawal request is submitted to the backend server, which generates a ZK proof and sends it to the relayer. No Metamask request is necessary for withdrawal, as the relayer handles gas fees and transaction signing, thus preserving the anonymity of the withdrawal address. The system verifies that the withdrawal address is also among the whitelisted addresses to ensure regulatory compliance.

Tests confirmed that the withdrawal is successfully executed and funds are transferred to the user's new address, while the relayer is compensated, demonstrating the maintenance of anonymity and compliance.

5. Discussion, challenges, and perspectives

5.1. Comparison with Tornado cash

Phantom ETH distinguishes itself from Tornado Cash by its native integration of a KYC verification mechanism based on ZK-SNARKs. Unlike Tornado Cash, which does not offer identity verification, Phantom ETH allows for cascading whitelisting and ensures KYC traceability while protecting privacy.

However, Tornado Cash supports multi-denominations and has decentralized relayers, points where Phantom ETH can still improve.

Table 1 presents a comparison between the proposed anonymization solution and Tornado Cash, highlighting the advantages in terms of compliance and traceability.

Table 1
Comparison between Phantom ETH and Tornado Cash

Features	**Phantom ETH**	**Tornado Cash**
Functionality		
Transaction Anonymization	✓	✓
Integrated KYC Verification	✓	×
Cascading Authorization	✓	×
Multi-denomination Support	×	✓
Technical		
Use of zk-SNARK	✓	✓
On-chain Merkle Tree	✓	✓
Decentralized Relayers	×	✓
Nullifier hash	✓	✓
Compliance		
Regulatory Compliance	✓	×
KYC Traceability	✓	×
Privacy Protection	✓	✓
User experience		
Simplified Interface	✓	✓
Note Management	✓	✓

5.2. Challenges encountered

Several challenges were encountered during the prototype development:

- Implementation of ZK circuits with Circom, requiring a deep understanding of zero-knowledge proofs.
- Optimization of smart contract gas consumption, especially that of the Merkle tree and on-chain proof verifications.
- Testing and deployment phases on the local development environment, requiring validation of ZK proofs in various scenarios.

5.3. Current limitations of the solution

Despite its effective operation, Phantom ETH has limitations:

- **High gas cost on Ethereum:** On-chain proof verifications and Merkle tree operations are expensive, impacting the economic efficiency of transactions.
- **Absence of a queue for concurrent transactions:** This can lead to delays during periods of high activity.
- **Centralized relayer:** The system currently relies on a single relayer, which goes against the principle of decentralization.

While this work was done on a local test blockchain with Ganache, future deployments on testnets like Sepolia will allow us to evaluate in real-world conditions, especially considering gas volatility and network latency.

5.4. Future development prospects

To improve the robustness and scalability of Phantom ETH, several development avenues are being considered:

- **Cross-chain support:** The system could be extended to support other asset types and blockchains.
- **Support for custom denominations:** For our project, we worked with fixed transaction pools (1 ETH, 10 ETH, 100 ETH). The ideal would be to have a transaction pool that takes user-defined amounts.
- **Relayer decentralization:** Our relayer is hosted on our backend server, which introduces a trust assumption and goes against the principles of decentralization. An improvement would be to implement a relay network with random and decentralized selection logic, regular relay updates, for example via a relay registry contract, and a reputation mechanism. This will improve the decentralization of the system.
- **Gas cost:** In the system, Merkle tree operations and on-chain proof verifications are costly in gas. A future implementation of our system could rely on zkRollups like zkSync [20] or Polygon zkEVM [21] to reduce the cost of on-chain zero knowledge proof verification and merkle tree operations.
- **Audit and security:** Have the contracts and zk-SNARK circuits audited by security experts and the community.

6. Conclusion

This thesis demonstrated the feasibility of reconciling transaction anonymity on the Ethereum blockchain with regulatory compliance requirements. The prototype Phantom ETH offers an innovative solution by integrating a ZK-KYC system based on zero-knowledge proofs with a robust transaction mixer. Tests conducted on Ganache validated the effectiveness of this architecture, proving that it is possible to protect user privacy while adhering to legal obligations.

While limitations remain, particularly in terms of transaction costs and relayer centralization, the identified development prospects pave the way for more complete, optimized, and decentralized future versions of the solution. This project thus contributes to the creation of a more private and responsible blockchain ecosystem.

Declaration on Generative AI

During the preparation of this work, Claude Sonnet was used as a code assistant and Google Gemini 2.5 Flash for error correction and translation of sections of the document from French to English. After using these AI tools, the authors reviewed and modified the content as needed and assume full responsibility for the content of the publication.

A. Project code

The complete source code for Phantom ETH is available on GitHub. This repository contains all system components: smart contracts, ZK circuits, user interfaces, and relayer code.

GitHub Repository: Repository Link (https://github.com/cybfar/anonymous-tx)
The project is organized into several subfolders:

- **zkkyc-ui/**
  - KYC User Interface

- Reusable Vue components
- Solidity Smart Contracts
- Tests and deployment scripts
- ZK-SNARK circuits

- **mixer-ui/**
  - Phantom ETH User Interface
  - Reusable Vue components
  - Solidity Smart Contracts
  - Tests and deployment scripts
  - ZK-SNARK circuits
- **proof-server/**
  - Relayer code
  - Proof generation service
  - REST API

Installation, configuration, and deployment instructions are detailed in the README.md files of each sub-repository. Contributions are welcome.

B. Specific terminology

Gas fees: A term from the Ethereum blockchain, gas refers to the unit used to measure the cost of executing an operation or smart contract on the network. "Gas fees" therefore represent the costs paid in Ether for miners or validators to process and validate the transactions. This term is kept in English in the common language of blockchain developers to avoid any confusion with energy "gas".

Acknowledgments

I want to express my deep gratitude to Professor Eugène C. EZIN for his invaluable advice, insightful discussions, and constant support throughout the preparation of this work. His expertise and encouragement were a great help in successfully completing this work.

References

[1] pseudonymatblockchain, Identity: What is the difference between pseudonymity and anonymity ?, https://start-in-blockchain.fr/pseudonymat-anonymat-definition-difference/, 2024. ,Accessed on 18 October 2024.
[2] osintsrc, What is open-source intelligence?, https://www.sans.org/blog/what-is-open-source-intelligence/, 2024. ,Accessed on 18 October 2024.
[3] onchainsrc, Combining on-chain and off-chain analysis: A primer, https://www.soliduslabs.com/post/off-chain-and-on-chain-analysis, 2024. ,Accessed on 19 October 2024.
[4] anonpseudo, https://www.investopedia.com/tech/introduction-monero-xmr/, 2024. ,Accessed on 17 September 2024.
[5] kycdef, Kyc: definition, procedure and regulations, https://www.signicat.com/fr/blog/kyc-know-your-customer-importance, 2024. ,Accessed on 23 October 2024.
[6] amldef, Aml: How to comply with online anti-money laundering regulations, https://fastercapital.com/fr/contenu/AML---Comment-se-conformer-aux-reglementations-anti-blanchiment-d-argent-en-ligne.html, 2024. ,Accessed on 01 November 2024.
[7] fbireport, https://www.ic3.gov/AnnualReport/Reports/2023_IC3Report.pdf, 2023. ,Accessed on 05 November 2024.

[8] fincen, Financial crimes enforcement network, https://fr.wikipedia.org/wiki/Financial_Crimes_Enforcement_Network, 2024. ,Accessed on 17 November 2024.

[9] 5amld, Imposing sanctions on virtual currency mixer tornado cash, https://www.linklaters.com/en/insights/blogs/fintechlinks/2018/june/eu-opens-door-for-cryptocurrency-exchanges-to-apply-aml-rules, 2018. ,Accessed on 17 November 2024.

[10] tornadosanctions, Us treasury sanctions tornado cash, accused of laundering stolen crypto, https://techcrunch.com/2022/08/08/treasury-tornado-cash-laundering-stolen-crypto/, 2022. ,Accessed on 30 November 2024.

[11] cryptomixerstumblers, Crypto mixers and tumblers, what are they ?, https://preciousruby.medium.com/crypto-mixers-tumblers-what-are-they-579eb624d885, 2024. ,Accessed on 15 November 2024.

[12] lazarus, North korea's lazarus hackers behind recent crypto heists: Fbi, https://therecord.media/north-korea-lazarus-behind-crypto-heists, 2024. ,Accessed on 17 November 2024.

[13] zeroKP, Zero-knowledge proofs, https://ethereum.org/en/zero-knowledge-proofs/, 2025. ,Accessed on 07 February 2025.

[14] merkletreedef, Merkle tree in blockchain: What is it, how does it work and benefits, https://www.simplilearn.com/tutorials/blockchain-tutorial/merkle-tree-in-blockchain, 2024. ,Accessed on 07 December 2024.

[15] merkletree, How are merkle trees used in blockchains?, https://docs.alchemy.com/docs/merkle-trees-in-blockchains, 2024. ,Accessed on 07 December 2024.

[16] imerkletree, Incremental merkle tree, https://zokyo-auditing-tutorials.gitbook.io/zokyo-tutorials/tutorial-16-zero-knowledge-zk/definitions-and-essentials/incremental-merkle-tree, 2025. ,Accessed on 15 June 2024.

[17] tornadocash, How does tornado cash works ?, https://www.rareskills.io/post/how-does-tornado-cash-work, 2025. ,Accessed on 29 March 2025.

[18] tornadocancelsanctions, Victory for tornado cash as court rules sanctions were unlawful, https://www.bakerlaw.com/insights/victory-for-tornado-cash-as-court-rules-sanctions-were-unlawful/, 2025. ,Accessed on 30 November 2024.

[19] mimcsponge, Mimc, https://byt3bit.github.io/primesym/mimc/, 2016. ,Accessed on 15 January 2025.

[20] zksync, What is zksync (zk)? zoom in on this zkrollup fully compatible with the ethereum virtual machine (evm), https://cryptoast.fr/zksync-premier-zkrollup-compatible-ethereum-virtual-machine/, 2025. ,Accessed on 21 June 2025.

[21] polygonzkevm, What is polygon zkevm?, https://fr.beincrypto.com/apprendre/quest-ce-que-polygon-zkevm-on-vous-explique-tout/, 2025. ,Accessed on 21 June 2025.