zkAML: Zero-knowledge Anti Money Laundering in Smart Contracts with whitelist approach

Donghwan Oh
Hanyang University

Semin Han
Hanyang University

Jihye Kim
Kookmin University, Zkrypto

Hyunok Oh
Hanyang University, Zkrypto

Jiyeal Chung
Hanyang University

Jieun Lee
Bank of Korea

Hee-Jun Yoo
Bank of Korea

Tae wan Kim
Bank of Korea

Abstract

In the interconnected global financial system, anti-money laundering (AML) and combating the financing of terrorism (CFT) regulations are indispensable for safeguarding financial integrity. However, while illicit transactions constitute only a small fraction of overall financial activities, traditional AML/CFT frameworks impose uniform compliance burdens on all users, resulting in inefficiencies, transaction delays, and privacy concerns. These issues stem from the institution-centric model, where financial entities independently conduct compliance checks, resulting in repeated exposure of personally identifiable information (PII) and operational bottlenecks. To address these challenges, we introduce zkAML, a cryptographic framework that offers a novel approach to AML/CFT compliance. By leveraging zero-knowledge Succinct Non-Interactive Argument of Knowledge (zk-SNARK) proofs, zkAML enables users to cryptographically demonstrate their regulatory compliance without revealing sensitive personal information. This approach eliminates redundant identity checks, streamlines compliance procedures, and enhances transaction efficiency while preserving user privacy. We implement and evaluate zkAML on a blockchain network to demonstrate its practicality. Our experimental results show that zkAML achieves 55 transactions per second (TPS) on a public network and 324 TPS on a private network. The zk-SNARK proof generation times are 226.59ms for senders and 215.76ms for receivers, with a constant verification time of 1.47ms per transaction. These findings highlight zkAML's potential as a privacy-preserving and regulation-compliant solution for modern financial systems.

1 Introduction

In the global financial system, anti-money laundering (AML) and combating the financing of terrorism (CFT) measures have been pivotal in preventing illicit financial activities [2]. In particular, AML/CFT compliance has gained renewed prominence in the context of Central Bank Digital Currencies (CBDCs), given the decline in physical cash usage and the rise of digital economies [1].

Traditional AML/CFT compliance in cross-border scenarios often requires users to submit personal information multiple times for verification. For example, consider a cross-border transfer in which Alice, located in the United States, initiates a funds transfer to Bob in the EU. Upon receiving the request, the originating bank performs comprehensive know your customer (KYC) checks, verifies her identity, and conducts customer due diligence (CDD) to assess any AML/CFT risks. This initial verification process requires the bank to validate various details about Alice, such as her personal identification and transaction history. Once Alice's identity and transaction are approved, the originating bank prepares a payment message containing essential information such as Alice's reference, Bob's beneficiary details, and the transaction amount. Despite this initial verification, intermediary financial institutions—such as correspondent banks or financial market infrastructures (FMIs)— conduct their own redundant compliance checks before passing the transaction along. Finally, when the payment message reaches Bob's bank (the beneficiary bank in the EU), yet another layer of AML/CFT compliance is applied. Bob's bank again conducts identity checks and KYC procedures, verifying Bob's account details and ensuring that the transaction complies with both internal policies and the relevant EU regulatory frameworks. The funds are credited to Bob's account only after all verifications are completed. This repetitive compliance process creates two major challenges:

- *Transaction delays*: Multiple layers of verification significantly slow down international money transfers, reducing operational efficiency.
- *Privacy risks*: Users must repeatedly provide personal identifiable information (PII) across multiple institutions, increasing the risk of data breaches and the complexity of meeting regulations such as the EU General Data Protection Regulation (GDPR)[1].

[1]While the GDPR permits the processing of personal data when necessary

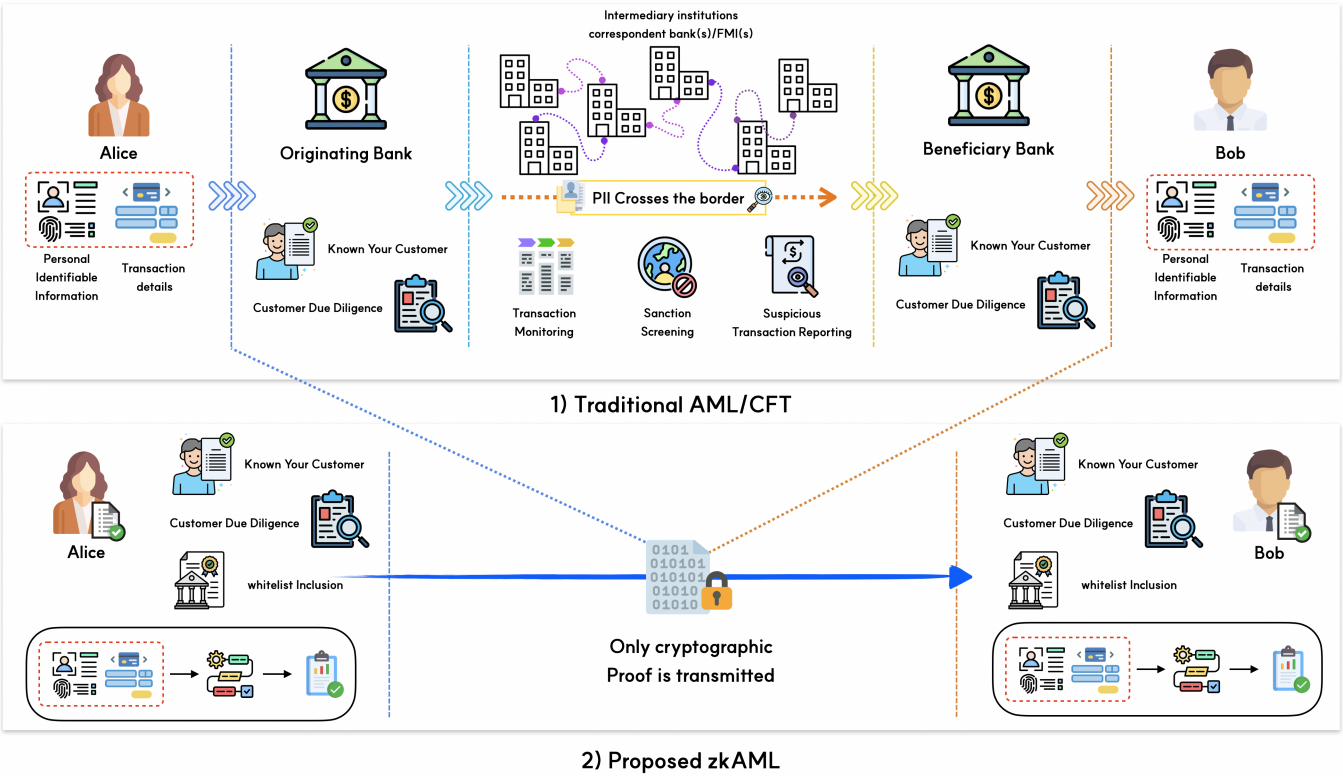

Figure 1: Comparison between zkAML and traditional AML/CFT

Despite these challenges, statistics indicate that regulatory compliance remains indispensable. According to the United States Sentencing Commission (USSC), the median loss in money laundering cases was $ 554,353 [14]. Nevertheless, USSC data also suggest that money laundering accounts for only a minuscule fraction of all transactions. Consequently, to regulate the illicit activities of a small minority, the vast majority of ordinary transactions—including those by creditworthy individuals—must also endure the aforementioned delays and privacy concerns. Thus, in addition to traditional AML/CFT measures, there is a need to establish a tailored transaction process for this wide range of ordinary transfers—one that improves structural efficiency and minimizes privacy exposure.

In light of these challenges, a new compliance framework must ensure AML/CFT enforcement while minimizing redundant verification. An ideal solution would enable financial institutions to verify compliance status without continuously collecting and processing sensitive personal data. This requires a shift from an institution-centric compliance model to a proof-based model that can guarantee regulatory adherence without excessive identity exposure.

for legal compliance, it still requires that only the minimum amount of data be used for narrowly defined purposes. Institutions enforcing AML/CFT regulations must also adopt additional data protection measures, for example through standard contractual clauses or binding corporate rules.

1.1 Our work

We propose zkAML, a cryptographic framework designed to provide frictionless AML/CFT compliance while preserving user privacy. zkAML fundamentally transforms the AML/CFT compliance model by eliminating redundant identity checks through zk-SNARK proof. This mechanism allows users to cryptographically prove that they belong to a whitelist—comprising individuals deemed very low risk—without disclosing their underlying personal information, thereby introducing a novel approach to balancing privacy and compliance. Instead of requiring users to repeatedly submit personal data to multiple institutions, zkAML enables them to prove their compliance status just once through a cryptographic proof.

zkAML introduces a trust-enabled compliance model that complements traditional AML/CFT frameworks by offering efficiency benefits to low risk users. Under the current institution-centric paradigm, each financial entity operates as an isolated compliance unit (as depicted in Figure 1), often leading to redundant identity checks. In contrast, zkAML allows one-time demonstration of compliance through verifiable cryptographic proofs, eliminating the need to repeatedly disclose personal information. By offering a streamlined and privacy-preserving pathway for low-risk (i.e., whitelisted) users, this approach coexists with the established regulatory

system while challenging its siloed structure and opening the way toward verifiable yet private compliance.

Specifically, zkAML enables users to generate a zk-SNARK proof demonstrating inclusion in a whitelist, which is maintained by a trusted authority and consists of creditworthy users[2]. This whitelist status, proven via zk-SNARK, serves as a cryptographic attestation of the user's pre-verified compliance. The remittance is transmitted via an anonymous transfer protocol (e.g., Blockmaze [18] or Azeroth [19]) to ensure privacy, i.e., transaction is encrypted and zk-SNARK proof is submitted to demonstrate the validity of transaction[3].

As illustrated in Figure 1, compared to traditional AML/CFT approaches, zkAML introduces two key distinctions: 1) Regulatory compliance is verified proactively, whereas traditional approaches require multiple verifications, and 2) zkAML requires users to submit a zk-SNARK proof instead of disclosing personal information. As a result, creditworthy users utilizing zkAML can reduce the unnecessary burdens imposed by traditional AML/CFT approaches and benefit two key aspects: 1) a streamlined transaction process, reducing delays, and 2) enhanced privacy protection by minimizing data exposure.

We emphasize that zkAML is designed to complement rather than replace existing AML/CFT compliance frameworks. Furthermore, it serves as an incentive mechanism for the majority of honest users (i.e., those with high creditworthiness). By offering such positive reinforcement, zkAML motivates users to improve their credit standing and compliance behavior.

Our contributions are summarized as follows:

- **A novel privacy-preserving AML/CFT compliance framework:** We address the conflict between regulatory compliance and user privacy by integrating compliance verification into zk-SNARKs. Creditworthy users no longer need to prove their identity repeatedly, improving overall transaction efficiency. Additionally, since they submit a zk-SNARK proof and the personal data is handled as a witness instead of submitting personal data itself, their privacy is significantly enhanced.

- **Security proof:** A privacy-preserving regulatory compliance framework must ensure both user privacy and regulatory compliant transactions. We define two key security notions as *transaction permission restrictness* and *compliance enforcement*. Transaction permission restrictness ensures that users retain control over their transactions, preventing unauthorized actions by bank or other entities, even when transaction authority is delegated. Compliance enforcement guarantees that all transactions adhere to AML/CFT regulations by verifying that both the sender and receiver fulfill the required conditions, thereby preventing illicit activities. We formally prove that zkAML satisfies these security properties under the assumptions of its underlying cryptographic primitives, demonstrating that it is secure, privacy-preserving regulatory compliance framework.

- **Implementation and evaluation:** We implement zkAML and empirically validate its practicality through experiments. Experimental results show that zkAML achieves 55 TPS on a public test network and 324 TPS on a private test network. Under a whitelist of 2^{20} users, proving times for the sender and receiver are 226.59ms and 215.76ms respectively (including regulatory checks), while verification time remains constant at about 1.47ms per transaction.

1.2 Related work

Privacy-preserving transfer. After Chaum [12] introduced anonymous payments, research on anonymous transfer grew significantly. In particular, with the emergence of blockchain, studies on anonymous transfer have been catalyzed. One of the earliest prominent protocols, Zerocash [7] , introduced anonymous transfer under the Unspent Transaction Output (UTXO) model by leveraging Zero-Knowledge Proofs (ZKPs) to ensure transaction validity. Subsequently, account-based protocols for anonymous transfers, such as Zether [9] and Blockmaze [18], were proposed. However, because these approaches rely on anonymous transfers, potential concerns remain regarding illicit financial activities.

Regulatory compliance through auditing. In response to the growing demand for compliance with regulations on illicit financial activities, various protocols have been proposed to support regulatory compliance. Camenisch et al. [10] sought to balance privacy and auditability by limiting how much a user can spend per transaction. Driven by advances in blockchain technology, additional auditable blockchain solutions have been introduced [4, 11, 13, 19, 20]. Although these approaches provide auditing and anonymous transfer to provide regulatory-compliant and user privacy simultaneously, there is still room for improvement: some do not support public blockchains [4, 20], others are feasible only for restricted environment (e.g., the number of users should be small) [13], or auditable only if whole keys and transaction details are revealed to auditor. Jeong et al. [19] propose and auditable anonymous transfer protocl that addresses these limitations by leveraging zk-SNARKs. However, it still does not prevent illicit financial activities in advance.

CBDCs. In a slightly different domain, research on regulatory-compliant CBDCs has recently gained momentum, spurred by the heightened emphasis on meeting AML/CFT require-

[2]Checking inclusion in a whitelist in zkAML is related to the traditional blacklist approach. In zkAML, whitelist inclusion implies the user would not be on a blacklist in a traditional system, representing a similar level of compliance assurance.

[3]For brevity, we provide only a high-level overview of the zkAML transfer. The details of the anonymous transfer mechanisms are discussed in Section 2.6 and their integration within zkAML is described in Section 3.

ments [1, 15]. This line of research focuses on embedding compliance mechanisms directly into CBDC architectures. Auer and Böhme [6] propose a "CBDC pyramid structure" that organizes design elements into hierarchical layers, allowing CBDCs to be tailored to consumer needs. They advocate choosing between a token-based model, emphasizing privacy and security, and an account-based model, which is more conducive to AML/CFT compliance. Although either option can be effective, each often requires regulatory adjustments. Pocher and Veneris [22] propose a regulation-by-design framework in Distributed Ledger Technology (DLT) to address privacy and transparency within CBDCs. They introduce a layered CBDC architecture that employs various Privacy-Enhancing Technologies (PETs)–such as ZKPs, Homomorphic encryption, and transaction-mixing–to validate transaction legitimacy without exposing sensitive details, meeting AML/CFT standards while preserving privacy.

Platypus [23] and PEReDi [21] each propose privacy-preserving, regulatory compliant CBDC frameworks that illustrates centralized and decentralized approaches, respectively. Both leverage ZKPs to enhance privacy and comply with AML/CFT regulations. Platypus, a centralized model, is governed by a single authority that enforces compliance, but this structure creates a single point of failure and increases dependency on central authorities. Conversely, PEReDi employs a decentralized structure that eliminates this vulnerability and ensures continuity even if the central authority becomes unavailable. However, both models rely on interactive transaction protocols—meaning that both sender and receiver must be online to complete a transaction. In PEReDi, for example, only the receiver's presence for zero-knowledge proof submission to claim funds. This online requirement limits usability for scenarios that demand non-interactive, asynchronous transactions.

1.3 Structure of the paper

After presenting the necessary notations and cryptographic primitives in Section 2, we introduce zkAML, our solution, in Section 3. We then provide the security notion underlying our construction in Section 4. In Section 5, we present the experimental evaluation of our construction. Finally, we summarize and conclude our work in Section 6.

2 Preliminaries

In this section, we provide some notations and (informal) definitions of cryptographic primitives used throughout this paper.

2.1 Notations

Let λ be the security parameter, with the standard notation $\leftarrow\!\!\$$ indicating random selection. We denote by $\mathbb{F}$ a finite field and by $\mathbb{G}$ a group.

Given the security parameter 1^λ, a relation generator $\mathcal{RG}$ outputs a polynomial-time decidable relation $\mathcal{R} \leftarrow \mathcal{RG}(1^\lambda)$. For a pair $(x,w) \in \mathcal{R}$, we refer to w as a witness for the statement x, indicating that x (an input/output statement) is in the relation.

Additionally, we employ a collision-resistant hash function (CRH) and a commitment scheme (COM). For any input x, we denote the output of the hash function by $y \leftarrow \mathsf{CRH}(x)$. For the commitment scheme, a commitment cm to a message u with an opening value o is defined as $\mathsf{cm} \leftarrow \mathsf{COM}(u;o)$.

Throughout the construction, the subscripts r, s, b, and t will denote the receiver, sender, bank, and transfer, respectively.

2.2 zk-SNARKs

As described in [16, 17] given a relation $\mathcal{R}$, a zero-knowledge Succinct Non-interactive Arguments of Knowledge (zk-SNARKs) consists of a set of algorithms $\Pi_{\mathsf{snark}} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{Sim})$ defined as follows:

- $\mathsf{Setup}(\lambda, \mathcal{R}) \rightarrow \mathsf{crs} := (\mathsf{ek}, \mathsf{vk}), \mathsf{td}$: This algorithm takes a security parameter λ and a relation $\mathcal{R}$ as inputs, returning a common reference string crs comprising an evaluation key ek and a verification key vk, along with a simulation trapdoor td.
- $\mathsf{Prove}(\mathsf{ek}, x, w) \rightarrow \pi$: Given an evaluation key ek, a statement x, and a witness w such that $(x,w) \in \mathcal{R}$, this algorithm outputs a proof π.
- $\mathsf{Verify}(\mathsf{vk}, x, \pi) \rightarrow \mathsf{true}/\mathsf{false}$: Using a verification key vk, a statement x, and a proof π as inputs, this algorithm outputs true if the proof is valid and false otherwise.
- $\mathsf{Sim}(\mathsf{ek}, \mathsf{td}, x) \rightarrow \pi_{\mathsf{sim}}$: This simulation algorithm, given an evaluation key ek, a simulation trapdoor td, and a statement x, generates a simulated proof π_{sim} that is indistinguishable from a real proof such that $\mathsf{Verify}(\mathsf{vk}, x, \pi_{\mathsf{sim}}) \rightarrow \mathsf{true}$.

This zk-SNARKs scheme satisfies the following properties: completeness, knowledge soundness, zero-knowledge, and succinctness.

Completeness. For any valid statement-witness pair (x,w) that satisfies the relation $\mathcal{R}$, the honest verifier always accepts the proof. Formally, for any security parameter $\lambda \in \mathbb{N}$, relation $\mathcal{R}_\lambda$, and any $(x,w) \in \mathcal{R}_\lambda$, it holds that:

$$\Pr\left[\mathsf{true} \leftarrow \mathsf{Verify}(\mathsf{vk}, x, \pi) \,\middle|\, \begin{array}{c} (\mathsf{ek}, \mathsf{vk}, \mathsf{td}) \leftarrow \mathsf{Setup}(\mathcal{R}); \\ \pi \leftarrow \mathsf{Prove}(\mathsf{ek}, x, w) \end{array} \right] = 1$$

Knowledge Soundness. Knowledge soundness ensures that if a prover outputs a valid proof π, then they must "know" a witness w for the statement x such that $(x,w) \in \mathcal{R}$. This knowledge is guaranteed by a knowledge extractor $\mathcal{E}$, which

can retrieve w from the prover's interaction in polynomial time. Formally, if there exists an extractor $\mathcal{E}$ for any probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ such that: $\Pr\left[\mathsf{Game}^{\mathsf{KS}}_{\mathcal{RG},\mathcal{A},\mathcal{E}} = \mathsf{true}\right] = \mathsf{negl}(\lambda)$ then Πsnark has knowledge soundness. The following describes this game:

$$\begin{array}{l}\mathsf{Game}^{\mathsf{KS}}_{\mathcal{RG},\mathcal{A},\mathcal{E}} \to \mathsf{res}\\ \hline (\mathcal{R},\mathsf{aux}_R) \leftarrow \mathcal{RG}(1^\lambda); (\mathsf{crs} := (\mathsf{ek},\mathsf{vk}),\mathsf{td}) \leftarrow \mathsf{Setup}(\mathcal{R});\\ (x,\pi) \leftarrow \mathcal{A}(\mathcal{R},\mathsf{aux}_R,\mathsf{crs}); w \leftarrow \mathcal{E}(\mathsf{transcript}_{\mathcal{A}});\\ \textbf{Return } \mathsf{res} \leftarrow (\mathsf{Verify}(\mathsf{vk},x,\pi) \wedge (x,\pi) \notin \mathcal{R})\end{array}$$

Zero-Knowledge. Zero-knowledge ensures that a proof π reveals no additional information about the witness w beyond the validity of the statement x. For Π_{snark} to be zero-knowledge, there must exist a simulator such that for any adversary $\mathcal{A}$, the following holds:

$$\Pr\left[\begin{array}{c}(\mathcal{R},\mathsf{aux}_R) \leftarrow \mathcal{RG}(1^\lambda); (\mathsf{crs} := (\mathsf{ek},\mathsf{vk}),\mathsf{td}) \leftarrow \Pi.\mathsf{Setup}(\mathcal{R})\\ : \pi \leftarrow \mathsf{Prove}(\mathsf{ek},x,w); \mathsf{true} \leftarrow \mathcal{A}(\mathsf{crs},\mathsf{aux}_R,\pi)\end{array}\right]$$

$$\equiv$$

$$\Pr\left[\begin{array}{c}(\mathcal{R},\mathsf{aux}_R) \leftarrow \mathcal{RG}(1^\lambda); (\mathsf{crs} := (\mathsf{ek},\mathsf{vk}),\mathsf{td}) \leftarrow \mathsf{Setup}(\mathcal{R})\\ : \pi_{\mathsf{sim}} \leftarrow \mathsf{Sim}(\mathsf{ek},\mathsf{td},x); \mathsf{true} \leftarrow \mathcal{A}(\mathsf{crs},\mathsf{aux}_R,\pi_{\mathsf{sim}})\end{array}\right]$$

Succinctness. A zk-SNARKs argument system Π_{snark} is succinct if the proof size is small and verification is efficient. Specifically:

$$|\pi| \leq \mathsf{Poly}(\lambda)(\lambda + log|w|)$$
$$\mathsf{Time}_{\mathsf{Verify}} \leq \mathsf{Poly}(\lambda)(\lambda + log|w| + |x|)$$

2.3 Commitment Schemes

A commitment scheme for some message $\mathbf{m}$ in message space $\mathcal{M}$, the triple of probabilistic polynomial time (PPT) algorithms (KeyGen,Commit,Open) are defined as follows:

- $\mathsf{KeyGen}(\lambda) \to (\mathbf{ck})$: The algorithm takes a security parameter λ as input and returns a commitment key ck.
- $\mathsf{Commit}(\mathbf{ck},\mathbf{m}) \to (C,o)$: The algorithm takes the commitment key $\mathbf{ck}$ and the message as input and outputs commitment C and opening o.
- $\mathsf{Open}(\mathbf{ck};C,\mathbf{m},o) \to \mathsf{true}/\mathsf{false}$: The algorithm takes the commitment, opening and claimed message to decide whether to accept claimed message as a valid opening of the commitment. The algorithm returns true with acceptance, and false otherwise.

Computational binding. A triple of three algorithms (KeyGen,Commit,Open) provides computational binding property if for any PPT adversary $\mathcal{A}$ having knowledge of $\mathbf{ck}$:

$$\Pr\left[\begin{array}{l|l}\mathsf{Open}(\mathbf{ck},C,\mathbf{m},o) & (C,\mathbf{m},o,\mathbf{m}',o')\\ \wedge\ \mathsf{Open}(\mathbf{ck},C,\mathbf{m}',o') & \leftarrow \mathcal{A}(\mathbf{ck})\\ \wedge\ \mathbf{m} \neq \mathbf{m}' & \end{array}\right] \leq \mathsf{negl}(1^\lambda).$$

Perfect hiding. A triple of three algorithms (KeyGen,Commit, Open) provides perfect hiding property if for all unbounded adversary $\mathcal{A} = (\mathcal{A}_0,\mathcal{A}_1)$:

$$\Pr\left[b = b' \left|\begin{array}{l}\mathbf{ck} \leftarrow \mathsf{KeyGen}(1^\lambda)\\ \wedge\ (\mathbf{a}_0,\mathbf{a}_1,st) = \mathcal{A}_0(\mathbf{ck})\\ \wedge\ b \leftarrow\$ \{0,1\}\\ \wedge\ (C,o) \leftarrow \mathsf{Commit}(\mathbf{ck},\mathbf{a}_b)\\ \wedge\ b' \leftarrow \mathcal{A}_1(C,st)\end{array}\right.\right] = \frac{1}{2}.$$

Perfect hiding ensures that no information about the committed value is revealed until the opening. In our construction, these properties allow us to commit to user balances or whitelist membership credentials without revealing the underlying data, ensuring privacy while still enabling compliance checks.

2.4 Signature schemes

A signature scheme is a triple of PPT algorithms $\mathsf{KeyGen},\mathsf{Sign},\mathsf{Verify}$ invoked as follows

- $\mathsf{KeyGen}(\lambda) \to (\mathsf{pk},\mathsf{sk})$: The algorithm takes a security parameter λ as input and returns a key pair containing a public verification key pk and a signing key sk.
- $\mathsf{Sign}(\mathsf{sk},m) \to \sigma$: The algorithm takes a signing key sk and a message m as inputs and returns a signature σ.
- $\mathsf{Verify}(\mathsf{pk},m,\sigma) \to \mathsf{true}/\mathsf{false}$: The algorithm takes a public verification key pk, a message m, and a signature σ as inputs, and returns true if the message and signature form a valid pair, or false otherwise.

Correctness. A triple of three algorithms $(\mathsf{Keygen},\mathsf{Sign},\mathsf{Verify})$ is correct if honestly generated signatures verify correctly with following probability:

$$\Pr\left[\begin{array}{c}\mathsf{true} \leftarrow \mathsf{Verify}(\mathsf{pk},m,\sigma)\\ :\ (\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{KeyGen}(\lambda), (m,\sigma) \leftarrow \mathsf{Sign}(\mathsf{pk},\mathsf{m})\end{array}\right] = 1$$

Strongly Unforgeable. A triple of three algorithms $(\mathsf{Keygen},\mathsf{Sign},\mathsf{Verify})$ is strongly unforgeable if the following adversary $\mathcal{A}$ has negligible advantage:

$$\Pr\left[\begin{array}{l:l}\mathsf{true} \leftarrow \mathsf{Verify}(\mathsf{pk},m,\sigma) & (\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{KeyGen}(\lambda)\\ (m,\sigma) \notin (m_i,\sigma_i)_{i=1}^q & (m,\sigma) \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot)}(\mathsf{pk})\end{array}\right]$$

where $\mathsf{Sign}(m_i)$ returns $\sigma_i \leftarrow \mathsf{Sign}(\mathsf{sk},m_i)$ for $i = 1,\ldots,q$.

The strong unforgeability property ensures that no adversary can produce a valid new signature on a message not previously signed. In our framework, this guarantees the authenticity and integrity of compliance proofs and transactions, preventing malicious entities from injecting unauthorized transactions.

2.5 Membership Proofs

A membership proof scheme [8] is a cryptographic proof that allows one to demonstrate that a particular element is a member of a predefined set without revealing the entire set. A membership proof scheme can be seen as a triple of algorithms $\mathsf{Acc}, \mathsf{Prove}, \mathsf{Verify}$ with the following functionality:

- $\mathsf{Acc}(S) \to A$: The algorithm compresses a set S into a short accumulator A.
- $\mathsf{Prove}(S, x) \to \pi_x$: The algorithm takes a set S and a member x such that $x \in S$ as inputs and returns a membership proof π_x.
- $\mathsf{Verify}(A, x, \pi_x) \to \mathsf{true}/\mathsf{false}$: The algorithm returns true if the membership proof π_x is valid or false otherwise.

We will utilize membership proofs to confirm that a user's credentials (e.g., whitelist inclusion) are valid without revealing the entire set of valid users or transferring PII, thereby preserving user privacy while maintaining compliance.

2.6 Anonymous transfer protocol

On blockchains like Ethereum, transactions are fully transparent, meaning that transaction details, such as payment information, are visible to anyone. This implies that, to maintain anonymity in transactions, such details must be concealed from unrelated third parties. To ensure user privacy on a public blockchain, we employ anonymous transfer protocols such as Azeroth [19], zeroCash [7], zether [9] and blockMaze [18]. In this paper, we apply Azeroth due to its auditability while safeguarding user privacy, which aligns with our framework's requirement, i.e., ensuring both privacy and regulatory compliance. Moreover, Azeroth is optimized for gas consumption, making it more efficient compared to other anonymous transfer protocols. We stress that the efficiency of Azeroth enables privacy-preserving transactions to be executed with minimal computational overhead in zkAML.

Revisit Azeroth. Azeroth utilizes two account types: Externally Owned Accounts (EOAs), similar to standard Ethereum accounts, and Encrypted Accounts (ENAs), which conceal the account balance using symmetric-key encryption. The core transaction, zkTransfer, leverages zk-SNARKs to enable anonymous transfers between ENAs, while allowing for deposits and withdrawals to and from EOAs. Importantly, zkTransfer ensures that transaction details related to ENAs, such as the amount, sender, and receiver, remain hidden during the verification process. We chose Azeroth for its auditable property, aligning with our framework's need for both privacy and regulatory compliance.

3 zkAML

This section describes the data structures used in our proposed scheme zkAML, referring to the notions defined in Section 2. Subsequently, we present an overview of the zkAML system, detailing its core techniques and providing a concrete description of its construction. The overview comprehensively explains the core functions' overall structure and functionality.

3.1 Overview

Before delving into zkAML, we provide an outline of its overall design. zkAMLis designed to facilitate privacy-preserving, regulatory-compliant cross-border transactions by integrating advanced cryptographic techniques. More specifically, zkAMLaddresses the tension between privacy and regulatory compliance by combining a whitelist-based compliance verification with zk-SNARKs, thereby enabling proactive verification.

3.1.1 Whitelist-based compliance verification

zkAML leverages a whitelist maintained by authorized institutions to pre-validate users' eligibility to perform transactions. By embedding compliance verification directly into the transaction protocol, zkAML ensures that only users who meet AML/CFT requirements can initiate transactions, thereby removing the need for subsequent audits.

zkAMLentrusts whitelist management to authorized national credit assessment institutions–such as public credit bureaus or private credit rating agencies–recognized by regulatory authorities. While each institution may use different methodologies to determine creditworthiness, zkAMLremains agnostic to these specifics and focuses on cryptographically verifying the legitimacy of an institution's claims. When a user initiates a cross-border transactions, the sending country's assessment institution generates a proof that certifies the user's acceptable credit standing and transaction eligibility. This proof is then submitted to the sending country's remittance institution (e.g., the user's bank), which verifies its authenticity to ensure compliance with both sending and receiving jurisdictions' AML/CFT standards. All verification steps employ cryptographic techniques to confirm the proof's validity without revealing sensitive personal information.

3.1.2 Privacy preservation

zkAML prioritizes user privacy while ensuring compliance with regulations. Instead of transmitting PII across borders, zkAML employs zk-SNARKs to generate cryptographic proofs. These proofs demonstrates that AML/CFT requirements–such as whitelist membership, sufficient creditworthiness, and adherence to transaction limits–are met, all without disclosing underlying sensitive data. This approach

helps satisfy regulations like GDPR and CCPA, which enforce strict controls over data processing. By confining PII within the user's home jurisdiction and using cryptographic proofs for cross-border validation, zkAML effectively mitigates privacy risks associated with international data transfers.

Furthermore, the use of smart contracts enhances the system by facilitating automated transaction execution once regulatory requirements are met. These contracts minimize the need for intermediaries, thereby reducing both delays and operational costs while improving the efficiency of cross-border financial transactions. This automated process, combined with the scalability and flexibility of zkAML, broadens the framework's applicability beyond CBDCs to include international remittances, inter-bank settlements, and decentralized finance (DeFi).

3.2 Data Structures

Ledger. All users can access the ledger denoted as L, which contains the information of all blocks. Each block in L includes:

- A list of transactions.
- A timestamp indicating when the block was created.
- A reference to the previous block, forming a chain.
- A Merkle root for verifying the integrity of transactions within the block.

Additionally, L is sequentially expanded by appending new transactions to the previous blocks (i.e., for any $\mathsf{T}' < \mathsf{T}$, L_{T} always incorporates $\mathsf{L}_{\mathsf{T}'}$).

Commitment. We employ a commitment scheme to construct privacy-preserving AML/CFT transactions. A commitment scheme allows a user to commit to a chosen value (or a chosen statement) while keeping it hidden from others, with the ability to reveal the committed value later. Commitments are utilized in zkAML to hide sensitive information such as credit limits ($\mathsf{v}_{\mathsf{lim}}$) and addresses ($\mathsf{addr}$).

Specifically, a commitment cm is created using the user's credit limit $\mathsf{v}_{\mathsf{lim}}$, address addr, and a randomly chosen opening value o as follows:

$$\mathsf{cm} = \mathsf{COM}(\mathsf{v}_{\mathsf{lim}}, \mathsf{addr}; \mathsf{o})$$

Note that the opening value o is known only to the user who owns the account, leading to ensure that sensitive information remains private.

Signature. Our zkAML protocol employs two distinct signatures:

- σ_{bank}: Issued by the bank, this signature guarantees that the user's ID and account address form a valid pair through the KYC process.
- σ_{send}: Issued by the sender, this signature authorizes the bank to execute the transfer and generate the zkAML proof. It contains the sender's address, the recipient's address, and the transfer amount, ensuring that only authorized transactions are processed by the bank.

Our signatures are defined as follows:

$$\sigma_{\mathsf{bank}} = \mathsf{Sign}(\mathsf{sk}_{\mathsf{b}}, (\mathsf{addr}, \mathsf{ID}))$$

$$\sigma_{\mathsf{send}} = \mathsf{Sign}(\mathsf{sk}_{\mathsf{s}}, (\mathsf{addr}_s, \mathsf{addr}_r, v))$$

3.3 Construction

Here we present the construction of zkAML. The details of the client algorithm and smart contract algorithm are available in Figure 2 and Figure 3, respectively. The relations for zk-SNARKs are depicted in Figure 4.

3.3.1 System Setup

Before transactions can occur, the system must be initialized by generating cryptographic keys, deploying the smart contract, and establishing the whitelist.

- **Setup:** A trusted authority runs $\mathsf{Setup}_{\mathsf{Client}}$ to generate the evaluation key ek and verification key vk, forming the public parameters $\mathsf{pp} := (\mathsf{ek}, \mathsf{vk}, G, 1^{\lambda})$.
- **Smart Contract Deployment:** The zkAML smart contract is deployed using $\mathsf{Setup}_{\mathsf{SC}}$, which stores vk and initializes a Merkle Tree for managing the whitelist $\mathsf{whitelist}$.
- **User Registration:** Each user must register with their respective bank before initiating transactions. During registration, the user provides their identity ID along with their account address addr. The bank verifies the user's provided identity against the existing whitelist $\mathsf{whitelist}$. Upon successful verification, the bank issues a signature σ_{bank} over $(\mathsf{ID}, \mathsf{addr})$. This signature ensures that the provided identity and account address correspond to a verified entity within the system and will be required in later transaction processes.

3.3.2 Transaction Initiation and Receiver Proof Generation

A transaction begins when a sender requests a cross-border transfer. The sender must be pre-verified against the whitelist managed by the authorized national credit assessment institution before proceeding.

1. **Transaction Intent Notification:** The sender communicates the transaction details (recipient address and amount v) to the receiver.

$\mathsf{Setup}_{\mathsf{Client}}(1^\lambda, \mathcal{R}_{\mathsf{zkAML}})$

$(\mathsf{ek}, \mathsf{vk}) \leftarrow \Pi_{\mathsf{snark}}.\mathsf{Setup}(\mathcal{R}_{\mathsf{zkAML}})$

$G \leftarrow\$ \mathbb{G}$ // Choose a generator

return $\mathsf{pp} := (\mathsf{ek}, \mathsf{vk}, G, 1^\lambda)$

$\mathsf{Sign}_{\mathsf{Client}}(\mathsf{sk})$

$\sigma_b \leftarrow \mathsf{Sign}_{\mathsf{b}}(\mathsf{sk}_{\mathsf{b}}, (\mathsf{addr}, \mathsf{ID}))$

return σ_b

$\mathsf{Delegate}_{\mathsf{Client}}(\mathsf{sk})$

$\sigma_s \leftarrow \mathsf{Sign}_{\mathsf{s}}(\mathsf{sk}_{\mathsf{s}}, (\mathsf{addr}_{\mathsf{s}}, \mathsf{addr}_{\mathsf{r}}, \mathsf{v}))$

return σ_s

$\mathsf{zkAMLTransfer}_{\mathsf{Client}}(\mathsf{pp}, \mathsf{note}, \mathsf{apk}, \mathsf{usk}^{\mathsf{send}}, \mathsf{upk}^{\mathsf{send,recv}}, \mathsf{v}^{\mathsf{priv}}_{\mathsf{out}}, \mathsf{v}^{\mathsf{pub}}_{\mathsf{in}}, \mathsf{v}^{\mathsf{pub}}_{\mathsf{out}}, \mathsf{EOA}^{\mathsf{recv}}, \mathsf{AML})$

parse $\mathsf{usk}^{\mathsf{send}}$ as $(\mathsf{k}^{\mathsf{send}}_{\mathsf{ENA}}, \mathsf{sk}^{\mathsf{send}}_{\mathsf{own}}, \mathsf{sk}^{\mathsf{send}}_{\mathsf{enc}})$

parse $\mathsf{upk}^{\{\mathsf{send,recv}\}}$ as $(\mathsf{addr}^{\{\mathsf{send,recv}\}}, \mathsf{pk}^{\{\mathsf{send,recv}\}}_{\mathsf{own}}, \mathsf{pk}^{\{\mathsf{send,recv}\}}_{\mathsf{enc}})$

parse AML as $(\mathsf{rt}_s, \mathsf{cm}_r, \mathsf{pk}_b, \mathsf{pk}_s, \sigma_b, \sigma_s, \mathsf{ID}_s, \mathsf{v}_{\mathsf{lim}\,r}, \mathsf{v}_{\mathsf{lim}\,s}, \gamma_r, \mathsf{Path}_s)$

if $\mathsf{note} \neq \perp$ **then**

  parse note as $(\mathsf{cm}_{\mathsf{old}}, \mathsf{o}_{\mathsf{old}}, \mathsf{v}^{\mathsf{priv}}_{\mathsf{in}})$

else

  $\mathsf{v}^{\mathsf{priv}}_{\mathsf{in}} \leftarrow 0; \mathsf{o}_{\mathsf{old}} \leftarrow\$ \mathbb{F}$

  $\mathsf{cm}_{\mathsf{old}} \leftarrow \mathsf{COM}(\mathsf{v}^{\mathsf{priv}}_{\mathsf{in}}, \mathsf{addr}^{\mathsf{send}}; \mathsf{o}_{\mathsf{old}})$

endif

$\mathsf{sct}_{\mathsf{old}} \leftarrow \mathsf{ENA}[\mathsf{addr}^{\mathsf{send}}]; \mathsf{v}^{\mathsf{ENA}}_{\mathsf{old}} \leftarrow \mathsf{SE.Dec}_{\mathsf{k}^{\mathsf{send}}_{\mathsf{ENA}}}(\mathsf{sct}_{\mathsf{old}})$

$\mathsf{nf} \leftarrow \mathsf{PRF}_{\mathsf{sk}^{\mathsf{send}}_{\mathsf{own}}}(\mathsf{cm}_{\mathsf{old}})$

$\mathsf{rt}_t \leftarrow \mathsf{List}_{\mathsf{rt}}.\mathsf{Top}$

  $\mathsf{Path} \leftarrow \mathsf{ComputePath}_{\mathsf{MT}}(\mathsf{cm}_{\mathsf{old}})$

$\mathsf{cm}_{\mathsf{new}} \leftarrow \mathsf{COM}(\mathsf{v}^{\mathsf{priv}}_{\mathsf{out}}, \mathsf{addr}^{\mathsf{recv}}; \mathsf{o}_{\mathsf{new}})$

$\mathsf{pct}_{\mathsf{new}}, \mathsf{aux}_{\mathsf{new}} \leftarrow \mathsf{PE.Enc}_{\mathsf{pk}^{\mathsf{recv}}_{\mathsf{enc}}, \mathsf{apk}}(\mathsf{o}_{\mathsf{new}} || \mathsf{v}^{\mathsf{priv}}_{\mathsf{out}} || \mathsf{addr}^{\mathsf{recv}})$

$\mathsf{v}^{\mathsf{ENA}}_{\mathsf{new}} \leftarrow \mathsf{v}^{\mathsf{ENA}}_{\mathsf{old}} + \mathsf{v}^{\mathsf{priv}}_{\mathsf{in}} - \mathsf{v}^{\mathsf{priv}}_{\mathsf{out}} + \mathsf{v}^{\mathsf{pub}}_{\mathsf{in}} - \mathsf{v}^{\mathsf{pub}}_{\mathsf{out}}$

$\mathsf{sct}_{\mathsf{new}} \leftarrow \mathsf{SE.Enc}_{\mathsf{k}^{\mathsf{send}}_{\mathsf{ENA}}}(\mathsf{v}^{\mathsf{ENA}}_{\mathsf{new}})$

$\vec{x}_s = \left\{\mathsf{apk}, \mathsf{rt}_t, \mathsf{rt}_s, \mathsf{nf}, \mathsf{upk}^{\mathsf{send}}, \mathsf{cm}_{\mathsf{new}}, \mathsf{cm}_r, \mathsf{pk}_b, \mathsf{pk}_{\mathsf{send}}, \mathsf{sct}_{\mathsf{old}}, \mathsf{sct}_{\mathsf{new}}, \mathsf{v}^{\mathsf{pub}}_{\mathsf{in}}, \mathsf{v}^{\mathsf{pub}}_{\mathsf{out}}, \mathsf{pct}_{\mathsf{new}}\right\}$

$\vec{w}_s = \left\{\mathsf{usk}^{\mathsf{send}}, \mathsf{cm}_{\mathsf{old}}, \mathsf{o}_{\mathsf{old}}, \mathsf{v}^{\mathsf{priv}}_{\mathsf{in}}, \mathsf{upk}^{\mathsf{recv}}, \mathsf{o}_{\mathsf{new}}, \mathsf{v}^{\mathsf{priv}}_{\mathsf{out}}, \mathsf{aux}_{\mathsf{new}}, \mathsf{Path}_t, \mathsf{Path}_s, \sigma_b, \sigma_s, \mathsf{ID}_s, \mathsf{v}_{\mathsf{lim}\,r,s}, \gamma_r\right\}$

$\pi_s \leftarrow \Pi_{\mathsf{snark}}.\mathsf{Prove}(\mathsf{ek}, \vec{x}_s, \vec{w}_s)$

$\mathsf{Tx}_{\mathsf{ZKT}} := (\pi_{s,r}, \vec{\mathsf{rt}}, \mathsf{nf}, \mathsf{addr}^{\mathsf{send}}, \mathsf{pk}_{r,s}, \mathsf{cm}_{\mathsf{new}}, \mathsf{cm}_r, \mathsf{sct}_{\mathsf{new}}, \mathsf{v}^{\mathsf{pub}}_{\mathsf{in}}, \mathsf{v}^{\mathsf{pub}}_{\mathsf{out}}, \mathsf{pct}_{\mathsf{new}}, \mathsf{EOA}^{\mathsf{recv}})$

return $\mathsf{Tx}_{\mathsf{ZKT}}$

Figure 2: zkAML client (Client) Algorithms

$\mathsf{Setup}_{\mathsf{SC}}(\mathsf{vk})$

/ Deploy an zkAML's smart contract

Store a zk-SNARK verification key vk

Initialize a Merkle Tree for whitelist whitelist

$\mathsf{zkAMLTransfer}_{\mathsf{SC}}(\mathsf{Tx}_{\mathsf{ZKT}})$

$$\textbf{parse } \mathsf{Tx}_{\mathsf{ZKT}} := \left(\begin{array}{c}\pi_{s,r},\ \vec{\mathsf{rt}},\mathsf{nf},\mathsf{addr}^{\mathsf{send}},\mathsf{pk}_{s,r},\mathsf{cm}_{\mathsf{new}},\mathsf{cm}_{\mathsf{recv}},\\ \mathsf{sct}_{\mathsf{new}},\mathsf{v}_{\mathsf{in}}^{\mathsf{pub}},\mathsf{v}_{\mathsf{out}}^{\mathsf{pub}},\mathsf{pct}_{\mathsf{new}},\mathsf{EOA}^{\mathsf{recv}}\end{array}\right)$$

$\textbf{parse } \vec{\mathsf{rt}} := (\mathsf{rt}_{\mathsf{transfer}},\mathsf{rt}_{\mathsf{send}},\mathsf{rt}_{\mathsf{recv}})$

assert $\mathsf{rt}_{\mathsf{transfer}} \in \mathsf{List}_{\mathsf{rt}}$

assert $\mathsf{nf} \notin \mathsf{List}_{\mathsf{nf}}$

assert $\mathsf{addr}^{\mathsf{send}} \in \mathsf{List}_{\mathsf{addr}}$

assert $\mathsf{cm}_{\mathsf{new}} \notin \mathsf{List}_{\mathsf{cm}}$

Get APK and $\mathsf{sct}_{\mathsf{old}}$

$$\vec{x}_{\mathsf{send}} = \left\{\begin{array}{c}\mathsf{APK},\ \mathsf{rt}_{\mathsf{send,transfer}},\mathsf{nf},\mathsf{upk}^{\mathsf{send}},\mathsf{pk}_s,\mathsf{cm}_r,\mathsf{cm}_{\mathsf{new}},\\ \mathsf{sct}_{\mathsf{old}},\mathsf{sct}_{\mathsf{new}},\mathsf{v}_{\mathsf{in}}^{\mathsf{pub}},\mathsf{v}_{\mathsf{out}}^{\mathsf{pub}},\mathsf{pct}_{\mathsf{new}}\end{array}\right\}$$

$\vec{x}_{\mathsf{recv}} = \{\mathsf{rt}_{\mathsf{recv}},\mathsf{cm}_r,\mathsf{pk}_r\}$

assert $\Pi_{\mathsf{snark}}\mathsf{VerProof}(\mathsf{vk},\pi_{\mathsf{send}},\vec{x}_{\mathsf{send}}) = \mathsf{true}$

assert $\Pi_{\mathsf{snark}}\mathsf{VerProof}(\mathsf{vk},\pi_{\mathsf{recv}},\vec{x}_{\mathsf{recv}}) = \mathsf{true}$

$\mathsf{ENA}[\mathsf{addr}^{\mathsf{send}}] \leftarrow \mathsf{sct}_{\mathsf{new}}; \mathsf{rt}_{\mathsf{new}} \leftarrow \mathsf{TreeUpdate}_{\mathsf{MT}}(\mathsf{cm}_{\mathsf{new}})$

$\mathsf{List}_{\mathsf{rt}}.\mathsf{append}(\mathsf{rt}_{\mathsf{new}}); \mathsf{List}_{\mathsf{nf}}.\mathsf{append}(\mathsf{nf})$

if $\mathsf{v}_{\mathsf{in}}^{\mathsf{pub}} > 0$ **then** $\mathsf{TransferFrom}(\mathsf{EOA}^{\mathsf{send}},\mathit{this},\mathsf{v}_{\mathsf{in}}^{\mathsf{pub}})$

if $\mathsf{v}_{\mathsf{out}}^{\mathsf{pub}} > 0$ **then** $\mathsf{TransferFrom}(\mathit{this},\mathsf{EOA}^{\mathsf{recv}},\mathsf{v}_{\mathsf{out}}^{\mathsf{pub}})$

Figure 3: zkAML's Smart Contract Algorithms

2. **Receiver Eligibility Verification:** The receiver uses the relation $\mathcal{R}_{\mathsf{recv}}$ to generate a zk-SNARK proof (π_{recv}) demonstrating that:
   - **Whitelist Membership:** The receiver's commitment (cm_r) exists in the receiver's whitelist Merkle Tree ($\mathsf{whitelist}_r$, with root rt_r), proving their membership.
   - **Identity Verification:** The receiver's address (addr_r) and ID (ID_r) match, as verified by the bank's signature (σ_b).
   - **Valid Signature:** σ_b is a valid signature on $(\mathsf{addr}_r,\mathsf{ID}_r)$ with the bank's public key pk_b.

$\mathsf{zkAML}_{\mathsf{recv}}$ relation $\mathcal{R}(\vec{x};\vec{w})$

$\vec{x} = \{\mathsf{rt}_r,\mathsf{cm}_r,\mathsf{pk}_b\}$

$\vec{w} = \{\sigma_b,\mathsf{addr}_r,\mathsf{ID}_r,\mathsf{v}_{\mathsf{lim}r},\gamma_r,\mathsf{Path}_r\}$

assert $\mathsf{true} = \mathsf{Membership}_{\mathsf{whitelist}}(\mathsf{rt},\mathsf{COM},\mathsf{Path}_r)$

assert $\mathsf{true} = \mathsf{SIG.verify}(\mathsf{pk_b},(\mathsf{addr}_r,\mathsf{ID_r}),\sigma_\mathsf{b})$

assert $\mathsf{cm}_r = \mathsf{COM}(\mathsf{addr}_r,\mathsf{v}_{\mathsf{lim}r};\gamma_r)$

$\mathsf{zkAML}_{\mathsf{send}}$ relation $\mathcal{R}(\vec{x};\vec{w})$

$\vec{x} = \{\mathsf{rt}_{\mathsf{send}},\mathsf{cm}_r,\mathsf{pk}_b,\mathsf{pk}_s\}$

$\vec{w} = \{\sigma_b,\sigma_s,\mathsf{addr}_{r,s},\mathsf{ID}_s,\mathsf{v}_{\mathsf{lim}r,s},v,\gamma_r,\mathsf{Path}_s\}$

assert $\mathsf{true} = \mathsf{Membership}_{\mathsf{whitelist}}(\mathsf{rt},\mathsf{COM},\mathsf{Path}_s)$

assert $\mathsf{v}_{\mathsf{lim}r} \geq v \wedge \mathsf{v}_{\mathsf{lim}s} \geq v$

assert $\mathsf{true} = \mathsf{SIG.verify}(\mathsf{pk_b},(\mathsf{addr_s},\mathsf{ID_s}),\sigma_\mathsf{b})$

assert $\mathsf{true} = \mathsf{SIG.verify}(\mathsf{pk_s},(\mathsf{addr_s},\mathsf{addr_r},\mathsf{v}),\sigma_\mathsf{s})$

assert $\mathsf{cm}_r = \mathsf{COM}(\mathsf{addr}_r,\mathsf{v}_{\mathsf{lim}r};\gamma_r)$

Figure 4: zkAML Relation

   - **Valid Commitment**: cm_r is a valid commitment to the receiver's address (addr_r) and credit limit ($\mathsf{v}_{\mathsf{lim}r}$) with randomness γ_r.
   - **Sufficient Credit Limit:** The receiver's committed credit limit ($\mathsf{v}_{\mathsf{lim}r}$) is sufficient to accept the specified transfer amount v.
3. **Proof Transmission:** The receiver sends the proof π_{recv} to the sender as authorization to proceed with the transaction.

3.3.3 Sender Proof Generation and Transaction Generation

Once the receiver's proof is validated, the sender must generate their own compliance proof before executing the transfer.

1. **Sender Proof Generation:** The sender constructs the zk-SNARKs proof (π_{send}) using $\mathsf{zkAMLTransfer}_{\mathsf{Client}}$ with the extended relation $\mathcal{R}_{\mathsf{zkAMLTransfer}}$. The statement $\vec{x}$ and witness $\vec{w}$ for this relation are as follows:

$$\vec{x} = \left(\begin{array}{c}\mathsf{apk},\ \mathsf{rt}_{\mathsf{send,transfer}},\mathsf{nf},\mathsf{upk}^{\mathsf{send}},\mathsf{pk}_s,\mathsf{cm}_r,\\ \mathsf{cm}_{\mathsf{new}},\mathsf{sct}_{\mathsf{old}},\mathsf{sct}_{\mathsf{new}},\mathsf{v}_{\mathsf{in}}^{\mathsf{pub}},\mathsf{v}_{\mathsf{out}}^{\mathsf{pub}},\mathsf{pct}_{\mathsf{new}}\end{array}\right)$$

zkAMLTransfer relation $\mathcal{R}(\vec{x};\vec{w})$

$$\vec{x} = \left\{ \begin{array}{c} \mathsf{apk},\ \mathsf{rt}_{\mathsf{send,transfer}}, \mathsf{nf}, \mathsf{upk}^{\mathsf{send}}, \mathsf{pk}_s, \mathsf{cm}_r, \mathsf{cm}_{\mathsf{new}}, \\ \mathsf{sct}_{\mathsf{old}}, \mathsf{sct}_{\mathsf{new}}, \mathsf{v}_{\mathsf{in}}^{\mathsf{pub}}, \mathsf{v}_{\mathsf{out}}^{\mathsf{pub}}, \mathsf{pct}_{\mathsf{new}} \end{array} \right\}$$

$$\vec{w} = \left\{ \begin{array}{c} \mathsf{usk}^{\mathsf{send}}, \mathsf{cm}_{\mathsf{old}}, \mathsf{o}_{\mathsf{old}}, \mathsf{v}_{\mathsf{in}}^{\mathsf{priv}}, \mathsf{upk}^{\mathsf{recv}}, \mathsf{o}_{\mathsf{new}}, \mathsf{v}_{\mathsf{out}}^{\mathsf{priv}}, \mathsf{aux}_{\mathsf{new}}, \\ \mathsf{Path}_t, \mathsf{Path}_s, \sigma_b, \sigma_s, \mathsf{ID}_s, \mathsf{v}_{\mathsf{lim}r,s}, \gamma_r \end{array} \right\}$$

parse $\mathsf{usk}^{\mathsf{send}}$ as $\left(\mathsf{k}_{\mathsf{ENA}}^{\mathsf{send}}, \mathsf{sk}_{\mathsf{own}}^{\mathsf{send}}, \mathsf{sk}_{\mathsf{enc}}^{\mathsf{send}}\right)$

parse $\mathsf{upk}^{\{\mathsf{send,recv}\}}$

as $\left(\mathsf{addr}^{\{\mathsf{send,recv}\}}, \mathsf{pk}_{\mathsf{own}}^{\{\mathsf{send,recv}\}}, \mathsf{pk}_{\mathsf{enc}}^{\{\mathsf{send,recv}\}}\right)$

if $\mathsf{v}_{\mathsf{in}}^{\mathsf{priv}} > 0$ **then**

  assert $\mathsf{true} = \mathsf{Membership}_{\mathsf{whitelist}}(\mathsf{rt}, \mathsf{cm}_{\mathsf{old}}, \mathsf{Path})$

endif

assert $\mathsf{pk}_{\mathsf{own}}^{\mathsf{send}} = \mathsf{CRH}(\mathsf{sk}_{\mathsf{own}}^{\mathsf{send}})$

assert $\mathsf{addr}^{\mathsf{send}} = \mathsf{CRH}(\mathsf{pk}_{\mathsf{own}}^{\mathsf{send}} || \mathsf{pk}_{\mathsf{enc}}^{\mathsf{send}})$

assert $\mathsf{cm}_{\mathsf{old}} = \mathsf{COM}(\mathsf{v}_{\mathsf{in}}^{\mathsf{priv}}, \mathsf{addr}^{\mathsf{send}}; \mathsf{o}_{\mathsf{old}})$

assert $\mathsf{nf} = \mathsf{PRF}_{\mathsf{sk}_{\mathsf{own}}^{\mathsf{send}}}(\mathsf{cm}_{\mathsf{old}})$

assert $\mathsf{pct}_{\mathsf{new}}, \mathsf{aux}_{\mathsf{new}} = \mathsf{PE.Enc}_{\mathsf{pk}_{\mathsf{enc}}^{\mathsf{recv}}, \mathsf{apk}}(\mathsf{o}_{\mathsf{new}} || \mathsf{v}_{\mathsf{out}}^{\mathsf{priv}} || \mathsf{addr}^{\mathsf{recv}})$

assert $\mathsf{addr}^{\mathsf{recv}} = \mathsf{CRH}(\mathsf{pk}_{\mathsf{own}}^{\mathsf{recv}} || \pi_{\mathsf{enc}}^{\mathsf{recv}})$

assert $\mathsf{cm}_{\mathsf{new}} = \mathsf{COM}(\mathsf{v}_{\mathsf{out}}^{\mathsf{priv}}, \mathsf{addr}^{\mathsf{recv}}; \mathsf{o}_{\mathsf{new}})$

if $\mathsf{sct}_{\mathsf{old}} = 0$ **then** $\mathsf{v}_{\mathsf{old}}^{\mathsf{ENA}} \leftarrow 0$

else $\mathsf{v}_{\mathsf{old}}^{\mathsf{ENA}} \leftarrow \mathsf{SE.Dec}_{\mathsf{k}_{\mathsf{ENA}}^{\mathsf{send}}}(\mathsf{sct}_{\mathsf{old}})$ **endif**

assert $\mathsf{v}_{\mathsf{new}}^{\mathsf{ENA}} \leftarrow \mathsf{SE.Dec}_{\mathsf{k}_{\mathsf{ENA}}^{\mathsf{send}}}(\mathsf{sct}_{\mathsf{new}})$

assert $\mathsf{v}_{\mathsf{new}}^{\mathsf{ENA}} = \mathsf{v}_{\mathsf{old}}^{\mathsf{ENA}} + \mathsf{v}_{\mathsf{in}}^{\mathsf{priv}} - \mathsf{v}_{\mathsf{out}}^{\mathsf{priv}} + \mathsf{v}_{\mathsf{in}}^{\mathsf{pub}} - \mathsf{v}_{\mathsf{out}}^{\mathsf{pub}}$

assert $\mathsf{v}_{\mathsf{out}}^{\mathsf{priv}} \geq 0; \mathsf{v}_{\mathsf{in}}^{\mathsf{priv}} \geq 0; \mathsf{v}_{\mathsf{in}}^{\mathsf{pub}} \geq 0; \mathsf{v}_{\mathsf{out}}^{\mathsf{pub}} \geq 0$

assert $\mathsf{v}_{\mathsf{new}}^{\mathsf{ENA}} \geq 0; \mathsf{v}_{\mathsf{old}}^{\mathsf{ENA}} \geq 0$

assert $\mathsf{v}_{\mathsf{lim}r} \geq \mathsf{v}_{\mathsf{out}}^{\mathsf{priv}} \wedge \mathsf{v}_{\mathsf{lim}s} \geq \mathsf{v}_{\mathsf{out}}^{\mathsf{priv}}$

assert $\mathsf{true} = \mathsf{SIG.verify}(\mathsf{pk}_{\mathsf{s}}, (\mathsf{addr}^{\mathsf{send}}, \mathsf{addr}^{\mathsf{recv}}, \mathsf{v}_{\mathsf{out}}^{\mathsf{priv}}), \sigma_{\mathsf{send}})$

Figure 5: zkAMLTransfer Relation

$$\vec{w} = \left(\begin{array}{c} \mathsf{usk}^{\mathsf{send}}, \mathsf{cm}_{\mathsf{old}}, \mathsf{o}_{\mathsf{old}}, \mathsf{v}_{\mathsf{in}}^{\mathsf{priv}}, \mathsf{upk}^{\mathsf{recv}}, \mathsf{o}_{\mathsf{new}}, \mathsf{v}_{\mathsf{out}}^{\mathsf{priv}}, \\ \mathsf{aux}_{\mathsf{new}}, \mathsf{Path}_t, \mathsf{Path}_s, \sigma_b, \sigma_s, \mathsf{ID}_s, \mathsf{v}_{\mathsf{lim}r,s}, \gamma_r \end{array} \right)$$

The relation $\mathcal{R}_{\mathsf{zkAMLTransfer}}$ extends the receiver's relation $\mathcal{R}_{\mathsf{recv}}$ and is considered valid if and only if the following conditions hold:

The relation $\mathcal{R}_{\mathsf{zkAMLTransfer}}$ extends the sender's relation $\mathcal{R}_{\mathsf{send}}$ and is considered valid if and only if the following conditions hold:

- **Receiver Verification**: The receiver's commitment (cm_r) is valid, and the receiver is eligible to receive the transaction (as verified in the receiver's proof).
- **Sender Delegation Signature**: σ_s is a valid signature on $(\mathsf{addr}_s, \mathsf{addr}_r, v)$ with the sender's public key pk_s, authorizing the transaction.
- **Transaction Amount Check**: The transaction amount (v, implicitly included in $\mathsf{v}_{\mathsf{out}}^{\mathsf{priv}}$) is non-negative and does not exceed the sender's committed credit limit ($\mathsf{v}_{\mathsf{lim}s}$).
- **Privacy preserving Transfer**: The proof must also satisfy all the constraints of underlying privacy-preserving transfer relation (zkTransfer [19]), including:
  - Correct computation of the new commitment ($\mathsf{cm}_{\mathsf{new}}$).
  - Correct encryption of the new ENA balance ($\mathsf{sct}_{\mathsf{new}}$).
  - Correct derivation of the nullifier (nf) from the old commitment ($\mathsf{cm}_{\mathsf{old}}$) and the sender's secret key.
  - Non-negativity of all amounts ($\mathsf{v}_{\mathsf{out}}^{\mathsf{priv}}$, $\mathsf{v}_{\mathsf{in}}^{\mathsf{priv}}$, $\mathsf{v}_{\mathsf{in}}^{\mathsf{pub}}$, $\mathsf{v}_{\mathsf{out}}^{\mathsf{pub}}$, $\mathsf{v}_{\mathsf{old}}^{\mathsf{ENA}}$, $\mathsf{v}_{\mathsf{new}}^{\mathsf{ENA}}$).
  - Balance equation: $\mathsf{v}_{\mathsf{new}}^{\mathsf{ENA}} = \mathsf{v}_{\mathsf{old}}^{\mathsf{ENA}} + \mathsf{v}_{\mathsf{in}}^{\mathsf{priv}} - \mathsf{v}_{\mathsf{out}}^{\mathsf{priv}} + \mathsf{v}_{\mathsf{in}}^{\mathsf{pub}} - \mathsf{v}_{\mathsf{out}}^{\mathsf{pub}}$.

3.3.4 Transaction Execution and Completion

Once the sender's and receiver's proofs are generated, the transaction proceeds through the following steps, ensuring compliance and privacy:

1. **Smart Contract Execution**: The sender (or the sender's bank on behalf of the sender) interacts with the zkAML smart contract. The zkAML smart contract $\mathsf{zkAMLTransfer}_{\mathsf{SC}}$ performs the following checks:
   - Verifies the sender's zk-SNARK proof (π_{send}), which encapsulates both the compliance checks and the underlying anonymous transfer protocol's requirements.

2. **State Update**: Upon successful verification of the proof, the smart contract updates the on-chain state, including:
    - Adding the nullifier (nf) to the list of spent nullifiers.
    - Updating the Merkle Tree root to reflect the new commitment and updating the recipient's encrypted balance in the ENA.
3. **Notification**: Both the sender and the receiver are notified of the successful transaction completion.

This proactive and privacy-preserving approach not only ensures compliance with AML/CFT requirements but also streamlines cross-border financial interactions by leveraging pre-verified whitelist mechanisms and reducing reliance on redundant post-transaction audits.

4 Security

Following the similar model defined in [7, 19], we define the security properties of zkAML including *ledger indistinguishability*, *transaction non-malleability*, *balance*, and *auditability*, and define *transaction permission restrictness* and *complience enforcement* as a new property.

Since zkAML transactions are encapsulated with zkTransfer transaction of [19]. Thus *ledger indistinguishability*, *transaction non-malleability*, *balance*, and *auditability* properties are inherited.

4.0.1 Transaction permission restrictness

We define the transaction permission restrictness property of zkAML. Transaction permission restrictness refers to the property that ensures a bank or any other entity cannot initiate or alter a user's transaction without the user's explicit permission, even if the authority to create transactions has been delegated to the bank.

Intuitively, a transaction follows the transaction permission restrictness property, no transaction can be created or executed by the bank without the user's authorization. This means the bank cannot independently modify or transfer the user's funds unless the transaction is explicitly approved by the user.

Definition 4.1 *Let* $\Pi_{\mathsf{zkAML}} = (\mathsf{Setup}, \mathsf{SIGN}, \mathsf{Delegate}, \mathsf{zkAMLTransfer})$ *be a* zkAML *scheme in Figure 2. We say for every* $\mathcal{A}$ *and security parameter* λ, Π_{zkAML} *is* TPR *secure if the following equations holds:*

$$\Pr\left[\mathsf{zkAML}.\mathcal{G}_{\mathcal{A}}^{\mathsf{TPR}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda)$$

4.0.2 Compliance enforcement

Compliance enforcement refers to the property that guarantees any transaction generated through zkAML adheres to AML/CFT regulations, ensuring that both the sender and the receiver are eligible and compliant with these standards.

More precisely, a transaction satisfies the compliance enforcement property, it can only be generated when both the sender and the receiver meet AML/CFT eligibility requirements, ensuring the transaction is lawful and compliant with regulatory standards. This property ensures that transactions are adhere to legal frameworks, preventing illicit activities.

Definition 4.2 *Let* $\Pi_{\mathsf{zkAML}} = (\mathsf{Setup}, \mathsf{SIGN}, \mathsf{Delegate}, \mathsf{zkAMLTransfer})$ *be a* zkAML *scheme in Figure 2. We say for every* $\mathcal{A}$ *and security parameter* λ, Π_{zkAML} *is* CE *secure if the following equations holds:*

$$\Pr\left[\mathsf{zkAML}.\mathcal{G}_{\mathcal{A}}^{\mathsf{CE}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda)$$

Theorem 4.1 *Let* $\Pi_{\mathsf{zkAML}} = (\mathsf{Setup}, \mathsf{SIGN}, \mathsf{Delegate}, \mathsf{zkAMLTransfer})$ *be a* zkAML *scheme in Figure 2.* Π_{zkAML} *satisfies ledger indistinguishability, transaction non-malleability, balance, auditability, transaction permission restrictness and Compliance enforcement.*

4.1 Security Proofs

We now formally prove Theorem 4.1 by showing that zkAML construction satisfies ledger indistinguishability, transaction non-malleability, balance, auditability, transaction permission restrictness and compliance enforcement.

4.1.1 Transaction permission restrictness

Let $\varepsilon := \mathbf{Adv}_{\mathsf{TPR}}^{\mathsf{zkAML},\mathcal{A}}(\lambda)$, where $\mathbf{Adv}_{\mathsf{TPR}}^{\mathsf{zkAML},\mathcal{A}}(\lambda)$ is the advantage that adversary $\mathcal{A}$ has in breaking TPR security.

We now construct an algorithm $\mathcal{B}$ that uses $\mathcal{A}$'s success in violating TPR to either break transaction non-malleability ($\mathsf{TR}-\mathsf{NM}$) or the unforgeability of the signature scheme.

Algorithm $\mathcal{B}$:

1. **Run** $\mathcal{A}$**:** Simulate the interaction between $\mathcal{A}$ and the challenger $\mathcal{C}$. Let Tx' be the transaction output by $\mathcal{A}$:
$$\mathsf{Tx}' = (\pi, \sigma_{\mathsf{bank}}, \sigma_{\mathsf{send}}, \ldots)$$
2. **Check** σ_{send}**:** Verify if σ_{send} is a valid signature from the user for the transaction parameters, including addresses and values. If the signature is invalid, $\mathcal{A}$ has failed, and $\mathcal{B}$ outputs 0.

   If σ_{send} is valid, proceed to step 3.
3. **Run zk-SNARKs Extractor** $\mathcal{E}$**:** Extract a valid witness $\vec{w}$ from $\mathcal{A}$'s zk-SNARKs proof π for Tx'.
4. **Verify zk-SNARKs Witness:** If $\vec{w}$ does not match the correct parameters (i.e., if the proof π is invalid), $\mathcal{A}$ has failed, and $\mathcal{B}$ outputs 0.

$\underline{\mathsf{zkAML}.\mathcal{G}_{\mathcal{A}}^{\mathsf{TPR}}(\lambda):}$

$\mathsf{pp} \leftarrow \mathsf{Setup}(\lambda)$

$\mathsf{L} \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{Azeroth}}}(\mathsf{pp}, \mathsf{ask})$

$\mathsf{Tx}' \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{Azeroth}}}(\mathsf{L})$

$\sigma_{\mathsf{send}}, m \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{Sign}}}(\mathsf{Tx}')$

$b \leftarrow \mathsf{VerifyTx}(\mathsf{Tx}', \mathsf{L}') \wedge \mathsf{Tx} \notin \mathsf{L}' \wedge \mathsf{VerifySign}(\mathsf{pk}_{\mathsf{s}}, \sigma_{\mathsf{send}}, m)$

return $b \wedge (\exists \mathsf{Tx} \in \mathsf{L} : \mathsf{Tx} \neq \mathsf{Tx}' \wedge \mathsf{Tx.nf} = \mathsf{Tx}'.\mathsf{nf})$

$\underline{\mathsf{zkAML}.\mathcal{G}_{\mathcal{A}}^{\mathsf{CE}}(\lambda):}$

$\mathsf{pp} \leftarrow \mathsf{Setup}(\lambda)$

$\mathsf{L} \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{Azeroth}}}(\mathsf{pp}, \mathsf{ask})$

$\mathsf{Tx}' \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{Azeroth}}}(\mathsf{L})$

$b \leftarrow \mathsf{VerifyTx}(\mathsf{Tx}', \mathsf{L}') \wedge \mathsf{Tx} \notin \mathsf{L}'$

return $b \wedge (\exists \mathsf{Tx} \in \mathsf{L} : \mathsf{Tx} \neq \mathsf{Tx}' \wedge \mathsf{Tx.nf} = \mathsf{Tx}'.\mathsf{nf})$

Figure 6: The experiments to transaction permission restrictness (TPR) and compliance enforcement (CE).

5. **Check for Signature Forgery:** If σ_{send} is valid and $\mathcal{A}$ has succeeded in producing a valid transaction Tx', then $\mathcal{B}$ has found a forgery for the user's signature. This would violate the unforgeability of the signature scheme.

6. **Check for Transaction Non-Malleability Violation:** If $\mathcal{A}$ has modified the transaction without authorization, $\mathcal{B}$ has found a violation of TR-NM, as a transaction was created with altered parameters (e.g., changed amounts or addresses) while still passing verification.

Since the transaction permission restrictness relies on valid user signatures and zk-SNARKs proofs, any successful attack on TPR implies a break in either the unforgeability of the signature scheme or the zk-SNARKs properties that uphold transaction non-malleability.

Thus, the probability that $\mathcal{A}$ wins the TPR experiment is bounded by the advantage in breaking TR-NM or the signature scheme, both of which are negligible:

$$\Pr[\mathsf{zkAML}.\mathcal{G}_{\mathcal{A}}^{\mathsf{TPR}}\mathcal{A}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$$

4.1.2 Compliance enforcement

Let $\varepsilon := \mathbf{Adv}_{\mathsf{CE}}^{\mathsf{zkAML},\mathcal{A}}(\lambda)$, where $\mathbf{Adv}_{\mathsf{CE}}^{\mathsf{zkAML},\mathcal{A}}(\lambda)$ is the advantage that adversary $\mathcal{A}$ has in breaking the Compliance Enforcement (CE) security of zkAML.

Since Compliance Enforcement relies on the same zk-SNARKs proof that guarantees transaction non-malleability ($\mathsf{TR}-\mathsf{NM}$), we can directly reduce breaking CE to breaking TR-NM.

We define an adversary $\mathcal{A}$ that attempts to break the CE property by generating a transaction that violates AML/CFT compliance, i.e., either the sender or receiver does not meet the eligibility criteria.

Algorithm $\mathcal{B}$:

1. **Simulate zkAML Interaction:** $\mathcal{B}$ simulates the interaction with the zkAML system, including the setup and execution of zk-SNARKs proofs for both sender and receiver.

2. **Extract zk-SNARKs Witness:** Let Tx' be the adversarial transaction output by $\mathcal{A}$ that claims to violate CE:

   $$\mathsf{Tx}' = (\pi_{s,r} \cdots)$$

   Run the zk-SNARKs extractor $\mathcal{E}$ to extract the witness $\vec{w}$ corresponding to π.

3. **Verify zk-SNARKs Witness:** Verify that $\vec{w}$ is valid for both sender and receiver compliance checks (i.e., both parties meet AML/CFT requirements). If the proof π is valid but $\mathcal{A}$ succeeded in breaking CE, then $\mathcal{B}$ has found a contradiction.

4. **Reduction to TR-NM:** Since CE and TR-NM are both enforced by the same zk-SNARKs proof, breaking CE implies that $\mathcal{A}$ has successfully generated a transaction that bypasses the zk-SNARKs compliance check. This directly implies a violation of TR-NM as well, as a non-compliant transaction (i.e., with an ineligible sender or receiver) has been accepted.

   Therefore, if $\mathcal{A}$ successfully breaks CE, $\mathcal{B}$ can break $\mathsf{TR}-\mathsf{NM}$.

The probability that $\mathcal{A}$ breaks CE is negligible, as it would require breaking the same zk-SNARKs proof that guarantees $\mathsf{TR}-\mathsf{NM}$. Thus, we have:

$$\Pr[\mathsf{zkAML}.\mathcal{G}_{\mathcal{A}}^{\mathsf{CE}}(\lambda) = 1] \leq \Pr[\mathsf{Azeroth}.\mathcal{G}_{\mathcal{A}}^{\mathsf{TR\text{-}NM}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$$

5 Implementation and Experiment

5.1 Implementation

Our zkAML implementation coded in rust, typescript and Solidity languages consists of two parts; the client and the smart contract. The client interacts with the blockchain network using Ethers library[4] from TypeScript. We implement zk-SNARKs in zkAMLusing the Arkworks library [5] based on Gro16 [16]. We use the curve BN254 for our instantiation.

[4] https://github.com/ethers-io/ethers.js

Table 1: Benchmark of zkAML

(a) Evaluation of zk-SNARKs in zkAML

	whitelist depth	constraint number	$\Pi_{\mathsf{snark}}.\mathsf{Prove}$(ms)	$\Pi_{\mathsf{snark}}.\mathsf{Verify}$(ms)	$\Pi_{\mathsf{snark}}.\mathsf{Setup}$(ms)
Receiver	10	16,389	179.50	1.47	158.02
	15	18,224	200.90	1.46	168.82
	20	20,059	215.76	1.47	190.22
Sender	10	20,359	197.84	1.47	178.20
	15	22,194	212.06	1.46	190.64
	20	24,029	226.59	1.47	210.01

(b) Throughput and gas consumption of $\mathsf{zkAML}_{\mathsf{SC}}$ with $\mathsf{MiMC7}_{32}$

		local test network (public)	in-process hardhat network	local test network (private)
zkAML	TPS	55.44	431.22	324.53
	Gas(k)	638		

The smart contract is deployed on the Ethereum test network using Hardhat[5].

In terms of COM and CRH, we instantiate the commitment using a hash function CRH as follows.

$$\mathsf{COM}(\mathsf{v}, \mathsf{addr}; \mathsf{o}) := \mathsf{CRH}(\mathsf{v}||\mathsf{addr}||\mathsf{o})$$

We use MiMC7 [3], SNARK-friendly hash function for CRH.

All the benchmarks presented in Section 5.2.1 were evaluated on a laptop with Apple M1 Pro processor and 32GB of RAM. The benchmarks in Section 5.2.2 are evaluated on various testing environments–local public test network, in-process hardhat network, and local private test network.

5.2 Experiment

5.2.1 Performance evaluation of zk-SNARK of zkAML

As demonstrated in Table 1a, the constraint number increases as the depth of the whitelist (whitelist) grows, leading to a corresponding increase in proving time. Specifically, the proving time grows from 179.50 ms to 226.59 ms as the depth increases from 10 to 20, reflecting the greater complexity associated with deeper whitelists. Unlike the proving time, the verification time remains constant approximately at 1.47ms, implying that it is independent of the whitelist depth and thereby making it efficient.

5.2.2 Performance evaluation of zkAML smart contract

We analyze the performance of smart contract $\mathsf{zkAML}_{\mathsf{SC}}$ in terms of throughput (transactions per second, TPS) and gas consumption. As illustrated in Table 1b, zkAML achieves 55.44 TPS on the public network, and it is increased to 324.53 TPS when it is run on the private network. In an in-process hardhat network, which provides an isolated and efficient testing environment, the system demonstrates a significantly higher throughput of 431.22 TPS. These results indicate that zkAML is capable of high performance across different blockchain environments, with the in-process hardhat network showing the best-case performance.

The smart contract consistently consumes around $638,000$ gas units across all three networks. The result shows that the gas consumption is manageable even though it accompanies the most time-consuming operation, $\Pi_{\mathsf{snark}}.\mathsf{Verify}$ in the smart contract. It implies that zkAMLis feasible on Ethereum or similar blockchains.

6 Conclusion

In this paper, we have presented a novel transaction protocol designed to address the significant challenges faced by traditional AML/CFT frameworks in combating financial crime. By leveraging advanced cryptographic techniques such as zero-knowledge proofs, signature schemes, and membership proofs, our protocol effectively balances the need for strict compliance with AML/CFT regulations while preserving user privacy.

The introduction of a whitelist mechanism, combined with zk-SNARKs, ensures that only legitimate users are able to perform transactions, dramatically reducing false negative rates and enhancing the overall efficiency of the system. Our approach significantly cuts down the costly and time-consuming investigations typically associated with traditional AML methods, offering a more scalable and flexible solution for financial institutions.

Moreover, the ability to verify user legitimacy without disclosing sensitive information positions our protocol as a critical innovation in the ongoing battle against financial economic crime. By minimizing the need for centralized authorities and enabling secure collaboration across financial institutions, our protocol supports both privacy and regulatory compliance,

[5] https://hardhat.org/

making it a robust and practical solution for the modern financial landscape.

The security and efficiency of our protocol have been validated through formal analysis and implementation on the Ethereum testnet. The experimental results affirm that our system is not only theoretically sound but also practical enough for real-world deployment. Moving forward, this work lays the groundwork for future advancements in the prevention of financial crimes, offering a forward-thinking approach to AML/CFT compliance in an increasingly digital world.

While the current work primarily focuses on preserving privacy through cryptographic techniques, future research will explore how incorporating additional transaction data in the proof can further enhance compliance. By leveraging the whitelist-based approach, we anticipate that these future developments will address the challenges of scalability and cross-border transactions, ensuring more robust compliance without compromising user privacy. This expanded approach will allow for real-time transaction validation and contribute to solving many of the limitations that traditional AML/CFT systems face today.

Additionally, although our protocol is designed with CBDCs in mind, the underlying framework and privacy-preserving techniques can be extended to other financial transactions beyond CBDCs. This flexibility highlights the broader applicability of our system to various financial use cases, such as international remittances or inter-bank transactions, further strengthening its utility across the financial ecosystem.

References

[1] Technical evaluation for a u.s. central bank digital currency system. Technical report, The White House, 2022.

[2] Noura Ahmed Al-Suwaidi and Haitham Nobanee. Anti-money laundering and anti-terrorism financing: a survey of the existing literature and a future research agenda. *Journal of Money Laundering Control*, 24(2):396–426, 2021.

[3] Martin R. Albrecht, Lorenzo Grassi, Christian Rechberger, Arnab Roy, and Tyge Tiessen. Mimc: Efficient encryption and cryptographic hashing with minimal multiplicative complexity. In *ASIACRYPT*, pages 191–219, 2016.

[4] Elli Androulaki, Jan Camenisch, Angelo De Caro, Maria Dubovitskaya, Kaoutar Elkhiyaoui, and Björn Tackmann. Privacy-preserving auditable token payments in a permissioned blockchain system. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, pages 255–267, 2020.

[5] arkworks contributors. `arkworks` zksnark ecosystem, 2022.

[6] Raphael Auer and Rainer Böhme. The technology of retail central bank digital currency. *BIS Quarterly Review, March*, 2020.

[7] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474, 2014.

[8] Josh Benaloh and Michael de Mare. One-way accumulators: A decentralized alternative to digital signatures. In Tor Helleseth, editor, *Advances in Cryptology — EUROCRYPT '93*, pages 274–285, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[9] Benedikt Bünz, Shashank Agrawal, Mahdi Zamani, and Dan Boneh. Zether: Towards privacy in a smart contract world. In Joseph Bonneau and Nadia Heninger, editors, *Financial Cryptography and Data Security - 24th International Conference*, pages 423–443, 2020.

[10] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Balancing accountability and privacy using e-cash. In *International conference on security and cryptography for networks*, pages 141–155. Springer, 2006.

[11] Ethan Cecchetti, Fan Zhang, Yan Ji, Ahmed Kosba, Ari Juels, and Elaine Shi. Solidus: Confidential distributed ledger transactions via pvorm. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 701–717, 2017.

[12] David Chaum. Blind signatures for untraceable payments. In *Advances in Cryptology: Proceedings of Crypto 82*, pages 199–203. Springer, 1983.

[13] Yu Chen, Xuecheng Ma, Cong Tang, and Man Ho Au. Pgc: Decentralized confidential payment system with auditability. In *Computer Security–ESORICS 2020: 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14–18, 2020, Proceedings, Part I 25*, pages 591–610. Springer, 2020.

[14] United States Sentencing Commission. https://www.ussc.gov/sites/default/files/pdf/research-and-publications/quick-facts/Money_Laundering_FY23.pdf, 2024.

[15] Yaya J Fanusie. Central bank digital currencies: the threat from money launderers and how to stop them. *The Digital Social Contract: A Lawfare Paper Series*, pages 1–23, 2020.

[16] Jens Groth. On the size of Pairing-Based non-interactive arguments. In *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the*

Theory and Applications of Cryptographic Techniques, pages 305–326, 2016.

[17] Jens Groth and Mary Maller. Snarky signatures: Minimal signatures of knowledge from Simulation-Extractable SNARKs. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference*, pages 581–612, 2017.

[18] Zhangshuang Guan, Zhiguo Wan, Yang Yang, Yan Zhou, and Butian Huang. Blockmaze: An efficient privacy-preserving account-model blockchain based on zk-snarks. *IEEE Transactions on Dependable and Secure Computing*, 2020.

[19] Gweonho Jeong, Nuri Lee, Jihye Kim, and Hyunok Oh. Azeroth: Auditable zero-knowledge transactions in smart contracts. *IEEE Access*, 11:56463–56480, 2023.

[20] Hui Kang, Ting Dai, Nerla Jean-Louis, Shu Tao, and Xiaohui Gu. Fabzk: Supporting privacy-preserving, auditable smart contracts in hyperledger fabric. In *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 543–555. IEEE, 2019.

[21] Aggelos Kiayias, Markulf Kohlweiss, and Amirreza Sarencheh. Peredi: Privacy-enhanced, regulated and distributed central bank digital currencies. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 1739–1752, 2022.

[22] Nadia Pocher and Andreas Veneris. Privacy and transparency in cbdcs: A regulation-by-design aml/cft scheme. *IEEE Transactions on Network and Service Management*, 19(2):1776–1788, 2021.

[23] Karl Wüst, Kari Kostiainen, Noah Delius, and Srdjan Capkun. Platypus: A central bank digital currency with unlinkable transactions and privacy-preserving regulation. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 2947–2960, 2022.