REGKYC: Supporting Privacy and Compliance Enforcement for KYC in Blockchains

Xihan Xiong[†*] , Michael Huth[‡], William Knottenbelt[†]
[†]Imperial College London, United Kingdom
[‡]Technische Universität Nürnberg, Germany

***Abstract*—Know Your Customer (KYC) is a core component of the Anti-Money Laundering (AML) framework, designed to prevent illicit activities within financial systems. However, enforcing KYC and AML on blockchains remains challenging due to difficulties in establishing accountability and preserving user privacy. This study proposes REGKYC, a privacy-preserving Attribute-Based Access Control (ABAC) framework that balances user privacy with externally mandated KYC and AML requirements. REGKYC leverages a structured ABAC model to support the flexible verification of KYC attributes and the enforcement of compliance policies, providing benefits to multiple stakeholders. First, it enables legitimate users to meet compliance requirements while preserving the privacy of their on-chain activities. Second, it empowers Crypto-asset Service Providers (CASPs) to tailor compliance policies to operational needs, ensuring adaptability to evolving regulations. Finally, it enhances regulatory accountability by enabling authorized deanonymization of malicious actors. We hope this work inspires future research to harmonize user privacy and regulatory compliance in blockchain systems.**

***Index Terms*—Blockchain, Compliance, Privacy, Know Your Customer (KYC), Attribute-Based Access Control (ABAC)**

I. INTRODUCTION

In traditional finance, Know Your Customer (KYC) procedures are implemented as part of regulatory obligations under Anti-Money Laundering (AML) frameworks to prevent illicit activities and ensure financial stability. In the context of the EU and UK, financial institutions such as banks, investment firms, insurance companies, and money service businesses are required to perform KYC under the EU's Anti-Money Laundering Directive (AMLD) and the UK's Money Laundering Regulations (MLR). KYC procedures often involve collecting and verifying customer information, such as identity documents, proof of address, and financial history.

In the cryptocurrency space, Crypto-asset Service Providers (CASPs) such as Centralized Exchanges (CEXs) and custodian wallet providers are often required to implement KYC procedures. The EU's 5th AMLD (AMLD5) extended AML regulations to include CASPs, while the Markets in Crypto-Asset (MiCA) regulation built a broader regulatory framework and reinforced AML obligations. Similarly, the UK's amended MLR (2019) brought CASPs under AML compliance.

However, the pseudonymous and decentralized nature of blockchain presents unique challenges for KYC, making it difficult to determine the identity behind on-chain fund flows once assets leave the regulated platforms. Typically, KYC in the cryptocurrency space is only feasible at points where fiat currencies are exchanged for cryptocurrencies or vice versa. CEXs, which serve as key entry and exit points for users within the blockchain system, are therefore required to verify and ensure the legitimacy of users' identities.

Although KYC procedures are widely adopted by CEXs for AML compliance, they remain insufficient in addressing several fundamental challenges. First, from the regulatory aspect, although the KYC process at CEXs links user identities to their initial on-chain addresses, there is no simple mechanism to determine whether a given on-chain address corresponds to an identifiable user. This gap in traceability presents significant issues when blockchain addresses are used for illicit activities. Without a clear and verifiable link between a blockchain address and a real-world identity, regulatory bodies cannot enforce accountability against criminal activities.

Second, from the perspective of CASPs, they will face increasingly stringent compliance requirements as the regulatory framework evolves. For example, verifying user nationality is crucial to comply with AML and sanctions regulations, as it helps identify individuals from high-risk or sanctioned jurisdictions. This ensures that CASPs can apply due diligence to prevent illicit activity. In addition, under the reverse solicitation rule, as outlined in Article 61 of the MiCA regulation, while CASPs are allowed to serve non-EU customers who actively seek out their services without being solicited, CASPs must still demonstrate that such interactions were initiated by the customer and not through proactive promotion. Without a reliable mechanism to determine the nationality behind blockchain addresses, CASPs may face compliance breaches.

Third, from the user's perspective, the privacy of a legitimate user should be protected after withdrawing assets from a CEX to on-chain addresses. However, through blockchain analytics, the CEX can monitor all subsequent transactions linked to a given identity. The recent US court ruling on Tornado Cash (TC), an on-chain mixing service, underscores the tension between privacy and regulation. TC was sanctioned in August 2022 for facilitating money laundering, but in November 2024, the court overturned the sanction, stating that targeting immutable smart contracts exceeded regulatory authority[1]. This affirms the value of user privacy while highlighting the challenge of aligning it with AML compliance.

[*] Corresponding author. Email: xihan.xiong20@imperial.ac.uk

[1] Court overturns US sanctions against cryptocurrency mixer Tornado Cash.

To mitigate these challenges, we propose REGKYC, a structured Attribute-Based Access Control (ABAC) [1] framework that balances user privacy with externally imposed AML and KYC compliance requirements in blockchain systems. From a regulatory perspective, REGKYC enables blockchain addresses to be linked to verifiable identities without disclosing specific details under normal circumstances. In cases of illicit activities, REGKYC provides a mechanism for authorized deanonymization, ensuring accountability while preserving privacy in regular operations. From the CASPs' perspective, REGKYC enables the design of tailored compliance policies, allowing flexible verification of KYC attributes and their combinations. This empowers CASPs to serve customers confidently while meeting evolving regulatory requirements. From the users' perspective, REGKYC ensures their on-chain activities remain unlinkable to their real-world identity by CEXs while maintaining compliance when interacting with CASPs. We summarize our main contributions as follows:

- **REGKYC Design.** We propose REGKYC, a privacy-preserving ABAC system that balances user privacy with externally imposed AML and KYC compliance requirements on smart contract-enabled blockchains. REGKYC enables CASPs to define tailored compliance policies. It also allows legitimate users to prove KYC verification and attribute compliance while preventing CEXs from linking users' KYC identities to their on-chain activities. In addition, it provides regulators with a mechanism to deanonymize malicious addresses when accountability is required.
- **Evaluation.** We evaluate the gas cost, scalability, and off-chain computation of the PRIVKYC pool, a core component in the REGKYC framework. The results demonstrate efficient gas usage and scalability, supporting up to 2^{30} depositors, aligning with Ethereum's projected address growth. The evaluation also shows the need to balance depositor capacity with the off-chain circuit generation time.
- **Application.** We discuss REGKYC's flexibility in verifying different combinations of KYC attributes. This allows CASPs to design tailored attribute verification mechanisms aligned with their service needs, addressing current compliance requirements while adapting to future regulations.

II. RELATED WORK

ABAC system. ABAC is a dynamic access control model that grants or denies access to resources based on the evaluation of policies against a set of attributes [1]. In recent years, some researchers have explored integrating the ABAC framework into blockchain systems to enhance privacy protection [2, 3].

Blockchain-based KYC Solutions. Martens *et al.* [4] explored the potential of blockchain for KYC. Mansoor *et al.* [5] provided a review of blockchain-based decentralized KYC solutions. Hussain and Usmani [6] proposed a decentralized KYC framework to enhance efficiency and reduce costs.

Blockchain-based Privacy Solutions. Bernabe *et al.* [7] highlighted the primary goals of privacy protection in blockchain systems: *anonymity*, which ensures that participants' identities remain undisclosed; *unlinkability*, which prevents transactions or blockchain addresses from being associated with one another; and *confidentiality*, which safeguards the content of transactions or smart contract data. Various cryptographic techniques are utilized to achieve these objectives. Peng *et al.* [8] provided a taxonomy of these techniques, including Zero-Knowledge Proofs (ZKPs), secure multi-party computation, homomorphic encryption, ring signatures, etc. Among these solutions, mixing services such as TC [9] have proven highly effective in breaking the traceability of transactions and blockchain addresses. TC utilizes Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (zk-SNARK) to anonymize transactions by allowing users to deposit funds into a smart contract and later withdraw them to a new address, thus achieving anonymity and unlinkability.

Privacy-Preserving KYC Solutions. Pauwels *et al.* [10] introduced zkKYC, which uses ZKPs and self-sovereign identity to enable privacy-preserving KYC in blockchain systems. Sun *et al.* [11] introduced a privacy-preserving KYC scheme for cross-chain identity management and compliance. Biryukov *et al.* [12] proposed KYCE, which uses cryptographic accumulators and ZKPs for compliance and privacy.

Compliance-Oriented Privacy Solutions. Academic research is actively exploring regulatory-compliant privacy solutions. For example, Buterin *et al.* [13] introduced two proof methods for privacy pools: *membership proofs* to show funds belong to a legitimate set and *exclusion proofs* to prove funds are not linked to illicit sources. Dotan *et al.* [14] introduces Haze and Daze, two compliant privacy mixers with mechanisms to block or retroactively deanonymize non-compliant users.

Our work draws inspiration from on-chain mixers such as TC but differs in three aspects. First, REGKYC enables KYC verification while preserving privacy. Second, REGKYC introduces a deanonymization mechanism, ensuring traceability when necessary. Third, REGKYC supports verifying specific combinations of KYC attributes to meet diverse compliance needs, beyond the generic privacy focus of mixers.

III. PRELIMINARIES

We adopt standard approaches [14–16] in defining foundational concepts. We denote the security parameter by 1^λ and the negligible function $\mathsf{negl}(\lambda)$. Public key pk is derived deterministically from private key sk: $\mathsf{pk} = \text{EXTRACTPK}(\mathsf{sk})$. The concatenation of binary strings k and r is denoted as $k||r$.

A. Hash Function

A hash function converts inputs of varying lengths into a fixed-size output. Denoted as a family H, each function within this family maps binary strings $\{0,1\}^*$ to a fixed output size $\{0,1\}^\lambda$. A family H is considered *collision-resistant* if no probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ can, with non-negligible probability, find two distinct inputs x and x' such that $h(x) = h(x')$ for a randomly selected $h \in H$.

B. Digital Signature

A digital signature is a cryptographic method to verify the authenticity and integrity of digital messages. In blockchain, it authorizes transactions and ensures data authenticity.

- Setup: Given a security parameter 1^λ, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$ initializes system parameters and generates sk and pk.
- Sign: Given the private key sk, $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$ processes the message m to produce a unique signature σ, which is later verifiable by the corresponding public key pk.
- Verify: This function validates whether a signature σ for a message m was generated using the sk corresponding to pk. It outputs a Boolean result, $0/1 \leftarrow \mathsf{Verify}(\mathsf{pk}, m, \sigma)$.

C. ZKP and zk-SNARK

A ZKP is a cryptographic protocol that typically involves a prover convincing a verifier that they know a witness w related to a public statement x, without disclosing w [17, 18].

A ZKP protocol must satisfy three properties: completeness, soundness, and zero-knowledge [17–19]. *Completeness* ensures that if w is a valid witness for the statement x, then an honest prover will always convince the verifier to accept the proof. *Soundness* implies that if the statement is false, an honest verifier will reject it with high probability. *Zero-knowledge* ensures that the protocol does not reveal any information about the w other than the fact that the x is true.

A zk-SNARK [20] is a specialized type of ZKP that enables a prover to convince a verifier of the truth of a statement, with the additional properties of succinctness and simulation extractability [21]. *Succinctness* indicates that the zk-SNARK system produces proofs that are very small in size and can be verified in constant time. *Simulation extractability* ensures that even if an attacker can access simulated proofs, they cannot produce a valid proof without knowing w.

- Setup: Given a circuit C, $(S_p, S_v) \leftarrow \mathsf{Setup}(1^\lambda, C)$ generates the public parameters for the prover and verifier.
- Prove: Given S_p, x and w, the prover generates a proof, represented as $\pi \leftarrow \mathsf{Prove}(S_p, x, w)$.
- Verify: Given S_v, x and π, the verifier outputs either true or false, represented as $0/1 \leftarrow \mathsf{Verify}(S_v, x, \pi)$.

D. Commitment Scheme

A commitment scheme [22] is a cryptographic protocol that allows one party to commit to a chosen value while keeping it hidden from another party with the option of later revealing the committed value. A commitment scheme consists of two main phases: commit and reveal. In the commit phase, the committer generates a commitment to a value m using a random value r, resulting in a commitment $\mathsf{com} = \mathsf{commit}(m, r)$. In the reveal phase, the committer can open the commitment by providing m and r. The receiver can verify the commitment and decide whether to accept it or not: $0/1 \leftarrow \mathsf{verify}(m, r, \mathsf{com})$.

E. Merkle Tree

A Merkle tree is a cryptographic data structure designed to efficiently verify the integrity and membership of data elements. Following previous studies [15, 16], we define:

- T.Setup: Given a security parameter λ and a list $L = (l_1, l_2, \ldots, l_k)$, this setup function builds a Merkle tree whose leaves are L and returns the root hash root as output. Formally, we define $\mathsf{root} \leftarrow T.\mathsf{Setup}(1^\lambda, L)$.
- T.Prove: This function generates a membership proof for an element $l_j \in \{0,1\}^*$, given its index j $(1 \leq j \leq k)$ and the list $L = (l_1, \ldots, l_k)$. Formally, $\mathsf{path}_j \leftarrow T.\mathsf{Prove}(j, l_j, L)$ outputs a proof path_j to show that l_j is indeed in L.
- T.Verify: This function takes an element $l_j \in \{0,1\}^*$, an index j $(1 \leq j \leq k)$, the root hash $\mathsf{root} \in \{0,1\}^\lambda$, and path_j as a proof. Formally, $0/1 \leftarrow T.\mathsf{Verify}(j, l_j, \mathsf{root}, \mathsf{path}_j)$ outputs 1 if π correctly verifies that l_j is in the tree.
- T.Update: This function updates the tree by replacing the j-th element $l_j \in L$ with a new element $l \in \{0,1\}^*$. Formally, $\mathsf{root}' \leftarrow T.\mathsf{Update}(\mathsf{root}, j, l)$ recomputes $\mathsf{root}' = T.\mathsf{Setup}(1^\lambda, L')$ where L' is list L with l_j replaced by l, and root is the root computed by the list L.

F. Public-Key Encryption

A public-key encryption scheme allows a sender to encrypt messages using a public key so that only the intended recipient, who possesses the corresponding private key, can decrypt them. The system consists of three core functions:

- Setup: $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$ generates a pair of keys: a public key pk and a private key sk.
- Encrypt: $c \leftarrow \mathsf{Encrypt}(m, \mathsf{pk})$ takes as input the pk and a message m to produce a ciphertext c.
- Decrypt: $m \leftarrow \mathsf{Decrypt}(c, \mathsf{sk})$ uses the private key sk and the ciphertext c to recover the original message m.

G. Threshold Encryption

Threshold Encryption [23] is a cryptographic scheme that encrypts a message such that decryption requires collaboration from at least t out of n authorized parties, where n is the total number of parties and $t \leq n$ is the threshold.

- Setup: $(\mathsf{pk}, \{\mathsf{sk}_i\}_{i=1}^n) \leftarrow \mathsf{Setup}(1^\lambda, t, n)$ generates a public encryption key pk and a set of private decryption key shares $\{\mathsf{sk}_1, \mathsf{sk}_2, ..., \mathsf{sk}_n\}$ for n participating parties.
- Encrypt: $c = \mathsf{Encrypt}(m, \mathsf{pk})$ takes as input the public key pk and a message m to produce a ciphertext c.
- Decrypt: $m = \mathsf{Combine}(\{\mathsf{ParticialDecrypt}(\mathsf{sk}_i, c)\})$ requires at least t decryption shares to recover the original message m. First, each party i generates a partial decryption $\mathsf{DecShare}_i = \mathsf{ParticialDecrypt}(\mathsf{sk_i}, \mathsf{c})$ using its private key share sk_i. Then at least t partial decryptions are combined to fully decrypt the ciphertext $m = \mathsf{Combine}(\mathsf{DecShare}_{i \in T})$ where $T \subseteq \{1, .., n\}$ with $|T| \geq t$.

IV. System Model

A. The ABAC System

ABAC [1] is an access control model that makes authorization decisions based on policies defined through a set of attributes. It typically contains the following components:

Policy Administration Point (PAP) defines access control policies. In RegKYC (Fig. 1), external regulatory authorities establish high-level compliance requirements. CASPs, acting as the PAP, translate these requirements into actionable rules and attribute policies according to their operational needs.

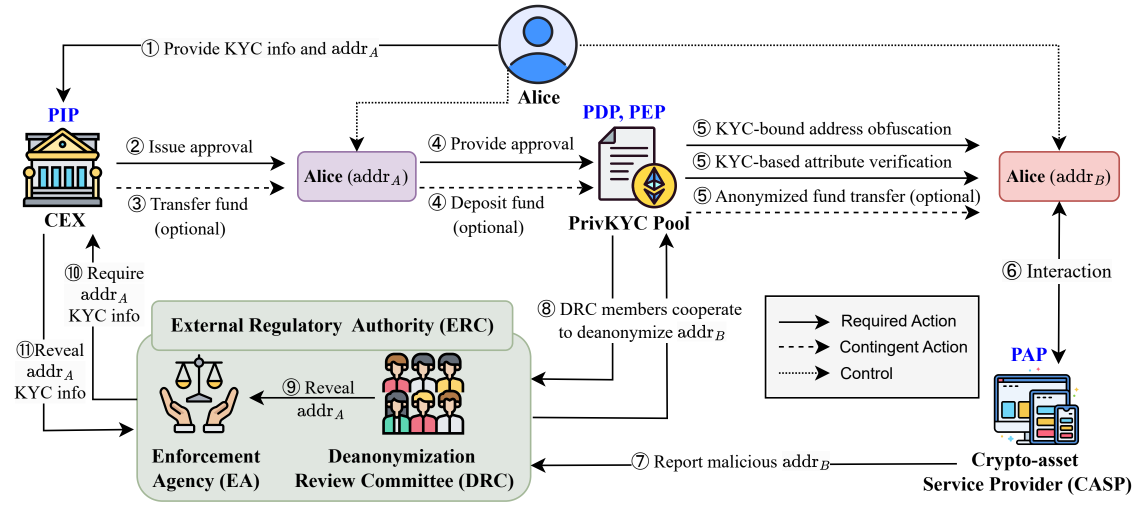
① Provide KYC info and addr$_A$
Alice
PIP
CEX
② Issue approval
③ Transfer fund (optional)
Alice (addr$_A$)
④ Provide approval
④ Deposit fund (optional)
PDP, PEP
PrivKYC Pool
⑤ KYC-bound address obfuscation
⑤ KYC-based attribute verification
⑤ Anonymized fund transfer (optional)
Alice (addr$_B$)
⑥ Interaction
PAP
Crypto-asset Service Provider (CASP)
⑦ Report malicious addr$_B$
⑧ DRC members cooperate to deanonymize addr$_B$
Required Action
Contingent Action
Control
External Regulatory Authority (ERC)
⑨ Reveal addr$_A$
Enforcement Agency (EA)
Deanonymization Review Committee (DRC)
⑩ Require addr$_A$ KYC info
⑪Reveal addr$_A$ KYC info

Fig. 1: Illustration of the RegKYC ABAC system, using Alice's interaction as an example to demonstrate key workflows.

Policy Information Point (PIP) collects attribute data for policy evaluation. In RegKYC, this role is fulfilled by the CEX, who collects KYC attributes and issues approvals.

Policy Decision Point (PDP) evaluates attributes against predefined compliance policies. The PrivKYC pool, acting as the PDP, verifies whether the attributes meet the criteria.

Policy Enforcement Point (PEP) enforces the PDP's decisions based on evaluation results. The PrivKYC pool also acts as the PEP to enforce the PDP's decisions on whether to accept users' deposit and withdrawal access requests.

B. System Participants and Components

User. Alice is a legally conscious user who seeks to participate in compliant on-chain activities. She provides her KYC information and attributes (e.g., nationality $\notin$ high-risk countries), along with a blockchain address (addr$_A$), to the CEX for approval. Alice wishes to engage in on-chain transactions and wants her interacting counterparties, such as CASPs, to know that the blockchain address she uses (e.g., addr$_B$) is associated with verified KYC information and attributes. This helps CASPs ensure that they are dealing with a compliant and verified user with qualified KYC attributes.

Additionally, Alice equally values her privacy. She wants to ensure that, although the CEX knows her KYC information and its association with her initial addr$_A$, the CEX cannot link her KYC to addr$_B$ or her subsequent on-chain activities. This allows Alice to maintain privacy over her transactions and address usage beyond the initial KYC process.

CASP. Acting as the PAP, CASPs translate high-level external regulatory requirements into actionable compliance policies tailored to their service and operation needs. For example, a CASP may define a compliance policy requiring the interacting address (e.g., addr$_B$) to prove that it belongs to a user from a non–high-risk jurisdiction. In addtion, CASPs can also report malicious addresses to the system.

CEX. The CEX, acting as the PIP, collects and verifies KYC information and attributes, issuing approvals based on verified user data. The CEX verifies Alice's KYC information, attributes, and addr$_A$, then issues an approval with a digital signature over these elements. This approval serves as proof of identity for the PrivKYC pool. If required, the CEX may disclose KYC data linked to addr$_A$ to regulatory authorities.

PrivKYC Pool. The PrivKYC pool acts as a PDP to evaluate whether users' KYC attributes meet predefined compliant attributes. Meanwhile, it also functions as a PEP, executing the decisions on access requests (e.g., deposits and withdrawals). This pool has the following functions:

- ***KYC-Bound Address Obfuscation***. The primary function of the PrivKYC pool is to break the linkability between Alice's original address (addr$_A$), which is associated with her KYC information, and her new address (addr$_B$). This ensures that while addr$_B$ remains compliant, it cannot be directly traced back to Alice's original identity or previous activities. As such, the CEX cannot link Alice's KYC to addr$_B$ or any subsequent transactions beyond addr$_A$.
- ***KYC Attribute Verification***. The pool verifies Alice's KYC attributes based on predefined criteria. For example, the pool verifies that her nationality is not classified as "high-risk" before permitting her to use addr$_B$ to interact with CASPs that restrict access from such regions. This guarantees that addr$_B$, while obfuscated, remains compliant with the specific requirements of the interacting CASPs.
- ***Anonymized Fund Withdrawal***. Alice can also use the PrivKYC pool to withdraw funds in a privacy-preserving manner. Analogous to traditional mixing services, the pool facilitates the transfer of funds from addr$_A$ to addr$_B$ while ensuring that the CEX and other observers cannot trace the flow of funds back to addr$_A$ or Alice's identity.
- ***KYC/AML Compliance***. The pool ensures that Alice's addr$_B$ remains compliant with KYC/AML standards when interacting with CASPs. In terms of illicit activities, it allows external regulatory authorities, such as the Deanonymization Review Committee (DRC), to demononymize malicious

addresses securely under authorized circumstances.

External Regulatory Authority (ERC). The ERC is an external regulatory body overseeing compliance within the blockchain system. It contains the DRC and the Enforcement Agency (EA). The DRC consists of legal and blockchain experts. When a suspicious address (e.g., Alice's addr_B) is flagged by a CASP, the DRC uses a threshold decryption scheme to collectively decide whether deanonymization should proceed. When a predefined threshold of DRC members approves the request, the corresponding deposit address (e.g., addr_A) will be revealed. The DRC functions similarly to audit logs or enforcement triggers in traditional access control systems, ensuring deanonymization occurs securely.

After deanonymization, the DRC reveals addr_A to the EA. Then, the EA is able to retrieve Alice's original KYC data from the CEX and take enforcement action against Alice.

C. System Properties

The REGKYC system satisfies the following properties.

- **Security**: An unapproved address cannot be used to deposit into the PRIVKYC pool. For any single deposit, the user is limited to making only one withdrawal. Similarly, the same approval cannot be used for multiple deposits.
- **Privacy**: The PRIVKYC pool can break the linkability between the deposit and withdrawal transactions [24]. As a result, withdrawal addresses remain pseudonymous under normal circumstances, ensuring that compliant users' identities and transaction histories are not exposed unnecessarily. Furthermore, users' on-chain activities cannot be traced back to their KYC information by CASPs or other observers.
- **Compliance**: REGKYC ensures compliance through a dual approach. For legitimate users, it enables proofs of KYC verification and attribute compliance when interacting with CASPs. For illicit actors, REGKYC ensures regulatory accountability by allowing DRC members to collaboratively deanonymize withdrawal addresses and the EA to take appropriate enforcement actions.

V. SYSTEM DESIGN

A. System Workflow

This section describes the system workflow. Optional fund transfers are excluded from the discussion, as this functionality is implemented in existing mixers and can be adopted directly.

1) Setup: The system runs the following setups:

- $\mathsf{par} \leftarrow \mathsf{SystemSetup}(1^\lambda)$: The setup takes as input a security parameter λ, and outputs the system parameters.
- The CEX specifies a hash function H used to compute the hash value of the KYC information. The CEX also sets up a private key $\mathsf{sk}_{\mathsf{CEX}}$, a public key $\mathsf{pk}_{\mathsf{CEX}}$, and a signature scheme Π_{sig} to sign and verify the KYC approval.
- The DRC sets up a public key $\mathsf{pk}_{\mathsf{DRC}}$, a set of private keys $\{\mathsf{sk}^i_{\mathsf{DRC}}\}^n_{i=1}$, and a threshold encryption scheme Π_{te} with threshold $t \leq n$ to encrypt and decrypt the deposit data.
- A PRIVKYC pool with an attribute Attr is created. It contains $\mathsf{pk}_{\mathsf{CEX}}$, $\mathsf{pk}_{\mathsf{DRC}}$, the Merkle tree depth d, two SNARK-friendly hash functions H_p and H_{2p}, and a zk-SNARK instance Π_{zk}. Π_{zk}'s evaluation key is ek_{wd} and verification key is vk_{wd}. A Merkle tree T with depth d is initialized to store deposit commitments, with initial root Root^0_{dep}.

2) Approval from CEX: Upon receiving the KYC approval request from a user, the CEX performs the Approve algorithm:

- $\sigma_{\mathsf{addr}_{dep}}/\bot \leftarrow \mathsf{Approve}(\mathsf{addr}_{dep}, \mathsf{KYC}, \mathsf{Attr})$ takes as input the address addr_{dep}, the KYC information KYC, and the KYC attribute Attr (e.g., nationality $\notin$ high-risk country) provided by the user. The CEX verifies the validity of the provided KYC and Attr data. If the information is invalid, $\mathsf{Approve}$ returns $\bot$. Otherwise, the CEX computes the hash values of $H(\mathsf{KYC})$ and $H(\mathsf{Attr})$, then runs the signing algorithm $\sigma_{\mathsf{addr_{dep}}} \leftarrow \Pi_{sig}.\mathsf{Sign}(\mathsf{sk}_{\mathsf{CEX}}, H(\mathsf{KYC})||H(\mathsf{Attr})||\mathsf{addr}_{dep})$, and finally returns the signature $\sigma_{\mathsf{addr}_{dep}}$ as the approval for addr_{dep}.

3) Deposit KYC approvals into the PRIVKYC *pool:* A user issues a deposit transaction to interact with the pool.

- $((k_{dep}, r_{dep}), tx_{dep}) \leftarrow \mathsf{CreateDeposit}(H(\mathsf{KYC}), H(\mathsf{Attr}), \mathsf{addr}_{dep}, \sigma_{\mathsf{addr}_{dep}})$: The user selects two random values $k_{dep} \overset{\$}{\leftarrow} \{0,1\}^\lambda$ and $r_{dep} \overset{\$}{\leftarrow} \{0,1\}^\lambda$, and computes the deposit commitment $\mathsf{cm} = H_{2p}(k_{dep}||r_{dep})$. The algorithm returns a note $= (k_{dep}, r_{dep})$, and a deposit transaction tx_{dep} whose input data includes the hash values of $H(\mathsf{KYC})$ and $H(\mathsf{Attr})$, the approved address addr, the approval $\sigma_{\mathsf{addr}_{dep}}$ from the CEX, and the deposit commitment cm.

After receiving tx_{dep} from the user, the pool executes:

- $1/0 \leftarrow \mathsf{ExecuteDeposit}(tx_{dep})$ first parses tx_{dep} input data elements as: $(H(\mathsf{KYC}), H(\mathsf{Attr}), \mathsf{addr}_{dep}, \sigma_{\mathsf{addr}_{dep}}, \mathsf{cm})$. Then it checks whether $H(\sigma_{\mathsf{addr}_{dep}})$ is stored in $\mathsf{ListOfApprovals}$ to prevent double deposits. If $H(\sigma_{\mathsf{addr}_{dep}})$ is not found in $\mathsf{ListOfApprovals}$, the algorithm performs $\Pi_{sig}.\mathsf{Verify}(\mathsf{pk}_{\mathsf{CEX}}, H(\mathsf{KYC})||H(\mathsf{Attr})||\mathsf{addr}_{dep}, \sigma_{\mathsf{addr}_{dep}})$ to verify the approval from the CEX, and returns 0 for invalid approval. If valid, the smart contract executes: *(i)* Adds cm to $\mathsf{ListOfCommitments}$; *(ii)* Extracts the latest Merkle tree root $\mathsf{Root}^{old}_{dep} = \mathsf{ListOfRoots}_{dep,k}[\mathrm{idx}]$, where k is the root list size and idx is the current root index; *(iii)* Inserts cm into the Merkle tree to compute the new root: $\mathsf{Root}^{new}_{dep} = T.\mathsf{Update}(\mathsf{Root}^{old}_{dep}, j+1, \mathsf{cm})$, where j is the latest inserted element index of the Merkle tree; *(iv)* Updates $\mathsf{ListOfRoots}_{dep,k}$ with $\mathsf{Root}^{new}_{dep}$ and updates the current root index idx; *(v)* Appends $H(\sigma_{\mathsf{addr}_{dep}})$ to $\mathsf{ListOfApprovals}$ and finally returns 1.

4) Withdrawal from the PRIVKYC *Pool:* Similar to the existing on-chain mixers [9, 15, 16], the withdrawal algorithm allows a user to utilize the private note and private key sk_{wd} to produce a proof and initiate a withdrawal transaction.

- $tx_{wd} \leftarrow \mathsf{CreateWithdraw}(\mathsf{sk}_{wd}, \mathsf{note})$: The user first uses $\mathsf{note} = (k_{dep}, r_{dep})$ to calculate the withdrawal nullifier $\mathsf{nf}_{wd} = H_p(k_{dep})$ and the deposit commitment $\mathsf{cm} = H_{2p}(k_{dep}||r_{dep})$. Next, the user locates the index i such that $\mathsf{ListOfCommitments}^{\mathsf{b}}[i] = \mathsf{cm}$ and retrieves the public parameter $\mathsf{par}^{\mathsf{b}}$ from the smart contract in block b. The user extracts the most recent updated Merkle tree root $\mathsf{Root}_{dep} = \mathsf{ListOfRoots}_{dep,k}[\mathrm{idx}]$, and computes the Merkle path proof path_i, satisfying $T.\mathsf{Verify}(i, \mathsf{cm}, \mathsf{Root}_{dep}, \mathsf{path}_i) = 1$. The user encrypts the commitment cm using the DRC's public key $\mathsf{pk}_{\mathsf{DRC}}$: $\mathsf{enc} = \Pi_{te}.\mathsf{Encrypt}(\mathsf{cm}||H_p(r_{dep}), \mathsf{pk}_{\mathsf{DRC}})$. Then the user generates a fresh address addr_{wd} and uses the withdrawal witness $\mathsf{wit}_{dep} = (\mathsf{sk}_{wd}, k_{dep}, r_{dep}, \mathsf{path}_i)$ and the public statement $\mathsf{stmt} = \mathsf{stmt}[\mathsf{addr}_{wd}, \mathsf{nf}_{wd}, \mathsf{Root}_{dep}, \mathsf{enc}]$ to generate a proof $\pi_{wd} = \Pi_{zk}.\mathsf{Prove}(\mathsf{ek}_{wd}, \mathsf{wit}_{dep}, \mathsf{stmt})$, which satisfies $\mathsf{cm} = H_{2p}(k_{dep}||r_{dep}) \wedge \mathsf{nf}_{wd} = H_p(k_{dep}) \wedge T.\mathsf{Verify}(i, \mathsf{cm}, \mathsf{root}, \mathsf{path}_i) = 1 \wedge \mathsf{enc} = \Pi_{te}.\mathsf{Encrypt}(\mathsf{cm}||H_p(r_{dep}), \mathsf{pk}_{\mathsf{DRC}})$. The proof π_{wd} shows that the user knows the private note of a deposit without leaking it. π_{wd} also ensures that enc is indeed derived from the cm and $H_p(r_{dep})$. The user finally composes $(\mathsf{addr}_{wd}, \mathsf{enc}, \mathsf{nf}_{wd}, \mathsf{Root}_{dep}, \pi_{wd})$ as the input data of the withdrawal transaction tx_{wd}. Upon receiving tx_{wd}, the pool verifies whether the user has deposited the KYC approval into the pool and correctly generated the encryption enc.

- $1/0 \leftarrow \mathsf{ExecuteWithdraw}(tx_{wd})$ first parses the tx_{wd} input data as $(\mathsf{addr}_{wd}, \mathsf{enc}, \mathsf{nf}_{wd}, \mathsf{Root}_{dep}, \pi_{wd})$ and then checks whether nf_{wd} is not included in the withdraw ListOfNullifier and whether Root_{dep} is one element in the most recent Merkle tree $\mathsf{ListOfRoots}_{dep,k}$. It returns 0 if the conditions are not satisfied. Otherwise, the smart contract checks the validity of proof π_{wd} by verifying whether $\Pi_{zk}.\mathsf{Verify}(\mathsf{vk}_{wd}, \pi_{wd}, \mathsf{stmt}[pk_{sender}, \mathsf{nf}_{wd}, \mathsf{Root}_{dep}, \mathsf{enc}]) = 1$, where pk_{sender} is the public key of the sender of tx_{wd}. The smart contract then appends nf_{wd} into ListOfNullifier to avoid double withdrawal and the pair $(\mathsf{addr}_{wd}, \mathsf{enc})$ into the ListOfWithdrawAddr. The algorithm finally returns 1.

5) Interaction with CASPs: After withdrawing from a PRIVKYC pool that verifies the attribute Attr, the user can use the withdrawal address addr_{wd} along with the withdrawal transaction tx_{wd} to interact with CASPs. The CASP verifies:

- $1/0 \leftarrow \mathsf{VerifyAuthorization}(tx_{wd}, \mathsf{addr}_{wd}, \mathsf{Attr})$: Upon receiving an interaction request from the address addr_{wd}, the CASP checks its withdrawal transaction tx_{wd} to verify that addr_{wd} is indeed a withdrawal address from a PRIVKYC pool that handles the attribute Attr. If the verification is successful, the algorithm returns 1, allowing addr_{wd} to interact further with the CASP; otherwise, it returns 0.

6) Deanonymization: The DRC conducts the following deanonymization process when accountability is required:

- $\mathsf{addr}_{dep}/\perp \leftarrow \mathsf{Deanonymize}(\mathsf{addr}_{wd}, \{\mathsf{sk}^i_{\mathsf{DRC}}\}_{i\in T}, t)$: This algorithm first interacts with the pool to extract the pair $(\mathsf{addr}_{wd}, \mathsf{enc})$ from ListOfWithdrawAddr. If $(\mathsf{addr}_{wd}, \mathsf{enc})$ is not in the list, then it returns 0. Otherwise, the DRC members collborate to decrypt the ciphertext enc to obtain the plaintex $\mathsf{cm}||H_p(r_{dep})$ with a pre-defined threshold t: $\mathsf{cm}||H_p(r_{dep}) = \Pi_{te}.\mathsf{Decrypt}(\{\mathsf{sk}^i_{\mathsf{DRC}}\}_{i\in T}, t, \mathsf{enc})$ where $|T| \geq t$. Then the DRC parses $\mathsf{cm}||H_p(r_{dep})$ to obtain cm and retrieve the corresponding deposit address addr_{dep}. Finally, the algorithm returns addr_{dep} to EA, and the EA can request the CEX to provide the KYC data of the identity behind addr_{dep} and take enforcement actions accordingly.

VI. SYSTEM PROPERTIES ANALYSIS

A. Security

- **Double Withdrawal**: We follow the definition of existing work [15, 16] to define the probability of double withdrawal. Consider an adversary $\mathcal{A}$ who deposits into the PRIVKYC pool using a transaction tx^{b}_{dep} in block b, where the transaction contains the commitment cm. We define the adversarial advantage for double withdrawal as the probability that the adversary can successfully create two distinct withdrawal transactions linked to tx^{b}_{dep}:

$$\begin{aligned}\Pr[\mathcal{A}(tx^{\mathsf{b}}_{dep}(\mathsf{cm})) \rightarrow \left(tx^0_{wd}\left(\mathsf{nf}^{\mathsf{b}_0}_{wd}\right), tx^1_{wd}\left(\mathsf{nf}^{\mathsf{b}_1}_{wd}\right)\right)\\ \text{s.t. } \mathsf{nf}^{\mathsf{b}_0}_{wd} \overset{origin}{\leftarrow} \mathsf{cm} \wedge \mathsf{nf}^{\mathsf{b}_1}_{wd} \overset{origin}{\leftarrow} \mathsf{cm}\\ \wedge\, \mathsf{b}_0 > \mathsf{b} \wedge \mathsf{b}_1 > \mathsf{b} \wedge \mathsf{nf}^{\mathsf{b}_0}_{wd} \neq \mathsf{nf}^{\mathsf{b}_1}_{wd}] \leq \mathsf{negl}(\lambda)\end{aligned}$$

  Proof Sketch. Consider a deposit transaction tx_{dep} that includes the commitment cm. If an adversary can identify two distinct nullifiers, $\mathsf{nf}^{\mathsf{b}_0}_{wd}$ and $\mathsf{nf}^{\mathsf{b}_1}_{wd}$, both derived from cm, then there must exist two different pairs, (k_0, r_0) and (k_1, r_1), such that $H_{2p}(k_0||r_0) = H_{2p}(k_1||r_1) = \mathsf{cm}$, and $H_p(k_0) = \mathsf{nf}^{\mathsf{b}_0}_{wd} \neq H_p(k_1) = \mathsf{nf}^{\mathsf{b}_1}_{wd}$. Given secure cryptographic primitives, the probability of breaking the hash function's collision resistance is negligible.

- **Double Deposit**: A user cannot use the same approval to deposit twice. Consider an adversary who generates two deposit transactions tx^0_{dep} and tx^1_{dep}, which contain the same approval σ but different depsoit addresses addr_0 and addr_1. Then the probability that the pool will accept both of the two deposit transactions is negligible, i.e.,

$$\begin{aligned}\Pr[\mathcal{A}(H(\mathsf{KYC}), H(\mathsf{Attr}), \mathsf{addr}_0, \mathsf{addr}_1, \sigma,) \rightarrow (tx^0_{dep}, tx^1_{dep}) \text{ s.t.}\\ \mathsf{ExecuteDeposit}(tx^0_{dep}) = \mathsf{ExecuteDeposit}(tx^1_{dep}) = 1] \leq \mathsf{negl}(\lambda)\end{aligned}$$

  Proof Sketch. Because the PRIVKYC pool verifies the uniqueness of elements in ListOfApprovals during each deposit transaction, it is impossible for the contract to store the same value of $H(\sigma)$ more than once.

- **Unapproved Access**: A user cannot use an address with unapproved KYC information to deposit into the PRIVKYC pool. Consider an adversary $\mathcal{A}$ attempting to issue a deposit transaction to interact with the PRIVKYC pool using an address addr that has not been approved through the CEX 's KYC process. The probability that the pool will accept its deposit is negligible, i.e.,

$$\begin{aligned}\Pr[\mathcal{A}(H(\mathsf{KYC}), H(\mathsf{Attr}), \mathsf{addr}, \sigma) \rightarrow tx_{dep}\\ \text{s.t. } \mathsf{ExecuteDeposit}(tx_{dep}) = 1] \leq \mathsf{negl}(\lambda)\end{aligned}$$

  Proof Sketch. We assume the security of the underlying cryptographic primitives. If σ is not generated by the algorithm $\mathsf{Approve}(\mathsf{addr}, \mathsf{KYC}, \mathsf{Attr})$, the probability that $\Pi_{sig}.\mathsf{Verify}(\mathsf{pk}_{\mathsf{CEX}}, H(\mathsf{KYC})||H(\mathsf{Attr})||\mathsf{addr}, \sigma) = 1$ for an unapproved address is negligible. Consequently, the probability of the pool accepting tx_{dep} is also negligible.

B. Privacy

In the following, we analyze the privacy of PRIVKYC pool users against CEXs, CASPs, and other users.

1) Privacy against CASPs and other users: Following the definition of mixer privacy in [15, 16], we consider the privacy of a REGKYC user against CASPs and other users as follows:

Let $\mathsf{DepAddrs}^{[0,\mathsf{b}]}$ and $\mathsf{WdAddrs}^{[0,\mathsf{b}]}$ denote the set of addresses that are used to deposit into and withdraw from the PRIVKYC pool before block number b. Given a specific withdrawal address $\mathsf{addr}_{wd} \in \mathsf{WdAddrs}^{[0,\mathsf{b}]}$, the advantage for a PPT adversary $\mathcal{A}$ in associating its correct deposit address $\mathsf{addr}_{dep} \in \mathsf{DepAddrs}^{[0,\mathsf{b}]}$ is bounded as follows:

$$\Pr[\mathcal{A}(\mathsf{addr}_{wd}) \to \mathsf{addr}_{dep},\ \text{s.t. } \mathsf{addr}_{dep} \in \mathsf{DepAddrs}^{[0,\mathsf{b}]} \wedge \mathsf{addr}_{dep} \overset{assoc}{\leftrightarrow} \mathsf{addr}_{wd}] \leq \frac{1}{\left|\mathsf{DepAddrs}^{[0,\mathsf{b}]}\right|} + \mathsf{negl}(\lambda)$$

Proof Sketch. The adversary's advantage in breaking the cryptographic primitive is negligible, denoted as $\mathsf{negl}(\lambda)$. Each new deposit address increases the anonymity set. For a withdrawal address addr_{wd} at block b, the probability of identifying the corresponding deposit address addr_{dep} is at most $\frac{1}{|\mathsf{DepAddrs}^{[0,\mathsf{b}]}|}$. Therefore, the overall adversarial advantage is bounded by $\frac{1}{|\mathsf{DepAddrs}^{[0,\mathsf{b}]}|} + \mathsf{negl}(\lambda)$, which approaches $\mathsf{negl}(\lambda)$ as $|\mathsf{DepAddrs}^{[0,\mathsf{b}]}|$ grows large.

2) Privacy against CEX: Let $\mathsf{DepAddrs}^{[0,\mathsf{b}_0]}$ denote the set of deposit addresses before block number b_0 and $\mathsf{WdAddrs}^{[\mathsf{b}_0,\infty)}$ denote set of withdrawal addresses after b_0. Given a specific deposit address $\mathsf{addr}_{dep} \in \mathsf{DepAddrs}^{[0,\mathsf{b}_0]}$, the advantage for a PPT adversary $\mathcal{A}$ in associating its correct withdrawal address $\mathsf{addr}_{wd} \in \mathsf{WdAddrs}^{[\mathsf{b}_0,\infty)}$ is bounded as:

$$\Pr[\mathcal{A}(\mathsf{addr}_{dep}) \to \mathsf{addr}_{wd},\ \text{s.t. } \mathsf{addr}_{wd} \in \mathsf{WdAddrs}^{[\mathsf{b}_0,\infty)} \wedge \mathsf{addr}_{wd} \overset{assoc}{\leftrightarrow} \mathsf{addr}_{dep}] \leq \frac{1}{\left|\mathsf{WdAddrs}^{[\mathsf{b}_0,\infty)}\right|} + \mathsf{negl}(\lambda)$$

Proof Sketch. The adversary's advantage in compromising the cryptographic primitive remains negligible. If the adversary learns that the deposit address received approval before block b_0, they can infer that the withdrawal occurs after b_0. The total probability of identifying the withdrawal address is $\frac{1}{|\mathsf{WdAddrs}^{[\mathsf{b}_0,\infty)}|} + \mathsf{negl}(\lambda)$, which diminishes as $\left|\mathsf{WdAddrs}^{[\mathsf{b}_0,\infty)}\right|$ grows, preventing CEXs from tracing on-chain activities or linking them to users' KYC data.

C. Compliance

Given a withdraw address $\mathsf{addr}_{wd} \in \mathsf{WdAddrs}^{[0,\mathsf{b}]}$ that has been flagged for accountability before block number b, the probability for the DRC to link it to the correct deposit address addr_{dep} through threshold decyrption satisfies:

$$\Pr[\mathcal{DRC}(\mathsf{addr}_{wd}) \to \mathsf{addr}_{dep},\ \text{s.t. } \mathsf{addr}_{dep} \in \mathsf{DepAddrs}^{[0,\mathsf{b}]} \wedge \mathsf{addr}_{dep} \overset{assoc}{\leftrightarrow} \mathsf{addr}_{wd}] \geq 1 - \mathsf{negl}(\lambda)$$

Proof Sketch. If $\mathsf{addr}_{wd} \in \mathsf{WdAddrs}^{[0,\mathsf{b}]}$ was flagged malicious, the DRC members can extract the $(\mathsf{addr}_{wd}, \mathsf{enc})$ pair from the PRIVKYC pool. Because proof π_{wd} ensures that the enc is derived from correct cm and $H_p(r_{dep})$, the DRC can apply threshold decryption to obtain the plaintext $\mathsf{cm}||H_p(r_{dep}) = \Pi_{te}.\mathsf{Decrypt}(\{\mathsf{sk}^i_{\mathsf{DRC}}\}_{i\in T}, t, \mathsf{enc})$. The DRC then parse $\mathsf{cm}||H_p(r_{dep})$ to obtain cm and retrieve the corresponding deposit addresses addr_{dep}.

VII. IMPLEMENTATION

A. Implementation of PRIVKYC Pool

We adapt the implementation of an existing on-chain mixer, TC, to build the PRIVKYC pool.

- *zkSnarks and Merkle Trees*: Consistent with existing work on on-chain mixers [15, 16], we employ `Groth16` [25] as our zkSNARK implementation, chosen for its efficient proof size and low verification computational cost. For constructing Merkle trees, we utilize two SNARK-friendly hash functions: the Pedersen hash [26] and the MiMC hash [27]. Additionally, we generate withdrawal proofs off-chain using the `Circom` library and the `snarkjs` library.
- *Digital Signatures*: We use the `soliditySha3` function to compute the message hash, replicating Ethereum's `keccak256` as implemented in the `web3.js` library. To generate the signature, we leverage the `signMessage` function from the `ethers.js` library, which employs the `ECDSA` algorithm on `secp256k1` to sign messages. To perform the signature verification, we use Ethereum's `ecrecover` function to derive the signer's address from the provided signature and verify its authenticity. Additionally, due to `Circom`'s limited support for asymmetric encryption, we implement a simplified elliptic curve method.
- *Testbed*: We perform the experiments on a macOS Monterey machine with an Apple M1 chip (8-core, 3.2 GHz), 8 GB RAM, and 512 GB SSD storage.

B. Evaluation Results

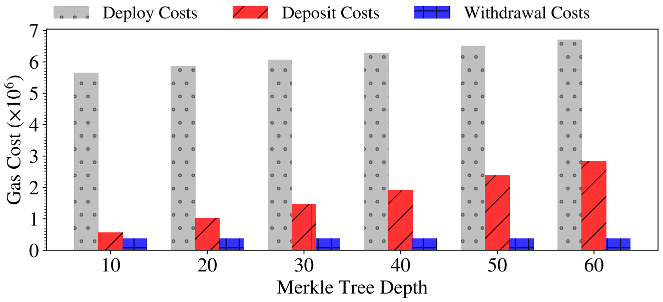

Fig. 2: Gas costs evaluation for the PRIVKYC pool.

1) Gas Costs: The gas cost evaluation for the PRIVKYC pool analyzes the deployment, deposit, and withdrawal costs as a function of the Merkle tree depth (see Fig. 2). The deployment costs remain the most significant contributor to overall gas consumption, requiring around 6×10^6 gas units. This reflects the one-time setup cost for deploying the PRIVKYC pool smart contract. The deposit costs grow with increasing Merkle tree depth, rising from approximately 0.57×10^6 gas units at depth 10 to 2.84×10^6 at depth 60. This increase is due to the higher computational overhead for inserting leaves into deeper Merkle trees. The withdrawal costs remain relatively low and consistent across all depths, averaging around 368,513 gas units. This efficiency is achieved by generating withdrawal proofs off-chain. Overall, the PrivKYC pool demonstrates similar gas efficiency for deployment, deposit, and withdrawal operations as TC. The gas costs of TC for deposit and withdrawal are 1,088,354 and 301,233 at tree depth of 20.

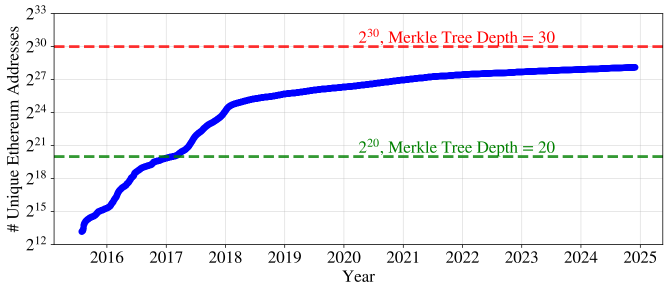

Fig. 3: The number of unique Ethereum addresses and the maximum supported depositors in the PrivKYC pool.

2) Scalability: The scalability of the PrivKYC pool is evaluated by comparing the number of unique Ethereum addresses over time with the maximum depositor capacity supported by the pool (see Fig. 3). The PrivKYC pool's depositor capacity is determined by the Merkle tree depth. The green and red dashed lines in the figure correspond to tree depths of 20 and 30, supporting 2^{20} and 2^{30} depositors respectively. These capacities represent the theoretical upper limit of depositors that the pool can handle. According to historical data from `etherscan.io`, the total number of unique Ethereum addresses grew from approximately 2^{12} to 2^{27} addresses. With a Merkle tree depth of 20, the pool can accommodate only a small set of depositors. However, a Merkle tree depth of 30 ensures that the pool can scale to handle nearly all unique Ethereum addresses. The evaluation highlights that, by supporting deeper Merkle trees, the PrivKYC pool can remain scalable for years to come.

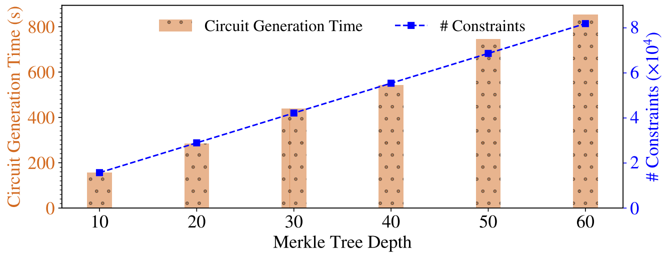

Fig. 4: Off-chain circuit generation time and number of constraints in the circuit with various Merkle tree depths.

3) Off-Chain zkSnarks Circuit Generation: Fig. 4 shows the relationship between the off-chain circuit generation time and the number of constraints in the circuit for various Merkle tree depths. The number of constraints grows linearly with the tree depth, increasing from approximately 1.57×10^4 constraints at depth 10 to 8.19×10^4 constraints at depth 60. In contrast, the circuit generation time increases non-linearly, from approximately 150 seconds at depth 10 to nearly 850 seconds at depth 60. Focusing on depths 20 and 30, which are highlighted in Fig. 3 as critical configurations, the circuit generation times are approximately 280 seconds and 430 seconds respectively. While a Merkle tree depth of 30 significantly increases depositor capacity, it comes with higher build-time costs. This highlights the need to balance depositor capacity with manageable circuit generation times.

VIII. Discussion

Extensions and Applications: A key strength of RegKYC is its flexibility to be extended to support conjunctions or disjunctions of attributes. While this work focuses on a single KYC attribute as an example, RegKYC can instantiate multiple PrivKYC pools, each designed to verify specific combinations of attributes (e.g., nationality $\notin$ high-risk countries & age > 20), thereby addressing more complex compliance requirements (see Appendix A for detailed discussion).

Limitations and Future Work: In this work, we mainly focus on evaluating the performance of the PrivKYC pool on-chain smart contracts and off-chain ZKPs, as these aspects have a direct impact on user experience. The off-chain governance and operational processes of the DRC also require evaluation to ensure the robust handling of deanonymization requests and governance resilience against adversarial manipulation.

The freshness of KYC attributes is also a relevant concern. Attributes such as age and financial credit rating may change over time, which can lead to previously verified KYC attributes becoming outdated. Future research can explore how the PIP may support the dynamic updating and validation of such attributes to ensure that compliance checks remain reliable.

While this work presents a technical exploration of privacy-preserving and KYC compliance solutions in blockchains, the challenge of balancing user privacy with compliance cannot be resolved by technical means alone. A sustainable solution requires a governance framework built on mutual trust and effective incentive mechanisms, where CASPs translate regulatory requirements into attribute-based policies, CEXs maintain accurate KYC data, and users develop compliance awareness.

IX. Conclusion

In this work, we propose RegKYC, a privacy-preserving ABAC framework that balances externally imposed KYC and AML regulatory requirements with user privacy in blockchain systems. RegKYC enables the flexible verification of KYC attributes, enforces compliance policies, and supports authorized deanonymization when accountability is required. This framework benefits multiple stakeholders. Users can achieve compliance while preserving the privacy of their on-chain activities. CASPs can implement tailored compliance policies aligned with their operational needs. External regulators can ensure accountability when necessary. We hope this structured framework provides a foundation to inspire the future development of best practices for integrating regulatory requirements into blockchain systems while protecting user privacy.

References

[1] V. C. Hu, D. R. Kuhn, D. F. Ferraiolo, and J. Voas, "Attribute-based access control," *Computer*, vol. 48, no. 2, pp. 85–88, 2015.

[2] Y. Chen, L. Meng, H. Zhou, and G. Xue, "A blockchain-based medical data sharing mechanism with attribute-based access control and privacy protection," *Wireless Communications and Mobile Computing*, vol. 2021, no. 1, p. 6685762, 2021.

[3] C. Zhonghua, S. Goyal, and A. S. Rajawat, "Smart contracts attribute-based access control model for security & privacy of iot system using blockchain and edge computing," *The Journal of Supercomputing*, vol. 80, no. 2, pp. 1396–1425, 2024.

[4] D. Martens, A. V. Tuyll Van Serooskerken, and M. Steenhagen, "Exploring the potential of blockchain for KYC," *Journal of Digital Banking*, vol. 2, no. 2, pp. 123–131, 2017.

[5] N. Mansoor, K. F. Antora, P. Deb, T. A. Arman, A. A. Manaf, and M. Zareei, "A review of blockchain approaches for KYC," *IEEE Access*, 2023.

[6] S. A. Hussain and Z.-u.-H. Usmani, "Blockchain-based decentralized KYC (know-your-customer)," in *International Conference on Systems and Networks Communications*, 2019.

[7] J. B. Bernabe, J. L. Canovas, J. L. Hernandez-Ramos, R. T. Moreno, and A. Skarmeta, "Privacy-preserving solutions for blockchain: Review and challenges," *IEEE Access*, vol. 7, pp. 164908–164940, 2019.

[8] L. Peng, W. Feng, Z. Yan, Y. Li, X. Zhou, and S. Shimizu, "Privacy preservation in permissionless blockchain: A survey," *Digital Communications and Networks*, vol. 7, no. 3, pp. 295–307, 2021.

[9] A. Pertsev, R. Semenov, and R. Storm, "Tornado cash privacy solution version 1.4," *Tornado cash privacy solution version*, vol. 1, p. 7, 2019.

[10] P. Pauwels, J. Pirovich, P. Braunz, and J. Deeb, "zkKYC in defi: An approach for implementing the zkKYC solution concept in decentralized finance," *Cryptology ePrint Archive*, 2022.

[11] N. Sun, Y. Zhang, and Y. Liu, "A privacy-preserving KYC-compliant identity scheme for accounts on all public blockchains," *Sustainability*, vol. 14, no. 21, p. 14584, 2022.

[12] A. Biryukov, D. Khovratovich, and S. Tikhomirov, "Privacy-preserving kyc on ethereum," in *Proceedings of 1st ERCIM Blockchain Workshop 2018*, European Society for Socially Embedded Technologies (EUSSET), 2018.

[13] V. Buterin, J. Illum, M. Nadler, F. Schär, and A. Soleimani, "Blockchain privacy and regulatory compliance: Towards a practical equilibrium," *Blockchain: Research and Applications*, vol. 5, no. 1, p. 100176, 2024.

[14] M. Dotan, A. Lotem, and M. Vald, "Haze: A compliant privacy mixer," *Cryptology ePrint Archive*, 2023.

[15] D. V. Le and A. Gervais, "AMR: Autonomous coin mixer with privacy preserving reward distribution," in *Proceedings of the 3rd ACM Conference on Advances in Financial Technologies*, pp. 142–155, 2021.

[16] Z. Wang, M. Cirkovic, D. V. Le, W. Knottenbelt, and C. Cachin, "Pay Less for Your Privacy: Towards Cost-Effective On-Chain Mixers," in *5th Conference on Advances in Financial Technologies (AFT 2023)*, vol. 282, pp. 16:1–16:25, 2023.

[17] D. Čapko, S. Vukmirović, and N. Nedić, "State of the art of zero-knowledge proofs in blockchain," in *2022 30th Telecommunications Forum (TELFOR)*, IEEE, 2022.

[18] X. Sun, F. R. Yu, P. Zhang, Z. Sun, W. Xie, and X. Peng, "A survey on zero-knowledge proof in blockchain," *IEEE network*, vol. 35, no. 4, pp. 198–205, 2021.

[19] J. Hasan, "Overview and applications of zero knowledge proof (ZKP)," *International Journal of Computer Science and Network*, vol. 8, no. 5, pp. 2277–5420, 2019.

[20] T. Chen, H. Lu, T. Kunpittaya, and A. Luo, "A review of zk-SNARKs," *arXiv preprint arXiv:2202.06877*, 2022.

[21] S. Atapoor and K. Baghery, "Simulation extractability in groth's zk-SNARK," in *International Workshop on Data Privacy Management*, pp. 336–354, Springer, 2019.

[22] A. Kate, G. M. Zaverucha, and I. Goldberg, "Constant-size commitments to polynomials and their applications," in *International conference on the theory and application of cryptology and information security*, pp. 177–194, Springer, 2010.

[23] Y. Desmedt, "Threshold cryptosystems," in *International Workshop on the Theory and Application of Cryptographic Techniques*, pp. 1–14, Springer, 1992.

[24] Z. Wang, S. Chaliasos, K. Qin, L. Zhou, L. Gao, P. Berrang, B. Livshits, and A. Gervais, "On how zero-knowledge proof blockchain mixers improve, and worsen user privacy," in *Proceedings of the ACM Web Conference 2023*, pp. 2022–2032, 2023.

[25] J. 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, Vienna, Austria, May 8-12, 2016, Proceedings, Part II 35*, pp. 305–326, Springer, 2016.

[26] L. Grassi, D. Khovratovich, C. Rechberger, A. Roy, and M. Schofnegger, "Poseidon: A new hash function for zero-knowledge proof systems.," in *USENIX Security Symposium*, vol. 2021, 2021.

[27] M. Albrecht, L. Grassi, C. Rechberger, A. Roy, and T. Tiessen, "Mimc: Efficient encryption and cryptographic hashing with minimal multiplicative complexity," in *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 191–219, Springer, 2016.

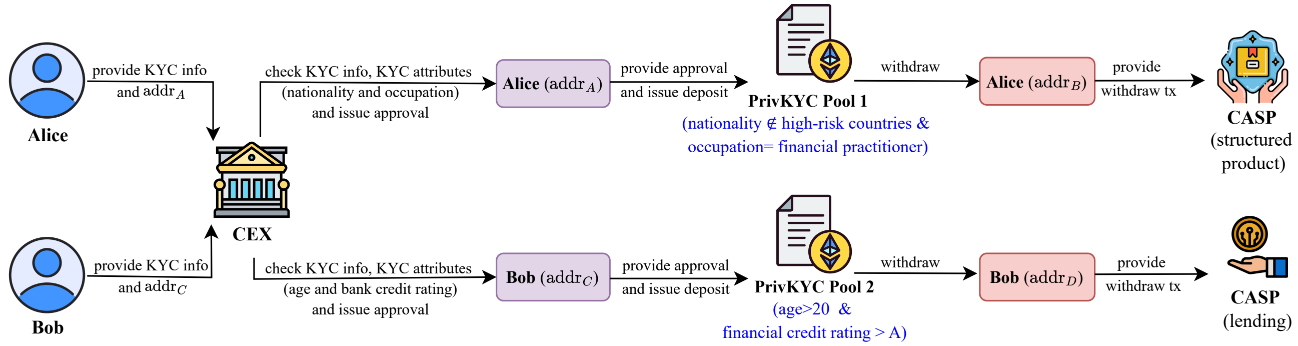

Fig. 5: Illustration examples of how REGKYC supports different combinations of KYC attributes.

APPENDIX A
EXTENSION AND APPLICATION

One of the key strengths of REGKYC lies in its flexibility to verify multiple KYC attribute combinations. This facilitates compliance with complex regulatory frameworks. By initializing tailored PRIVKYC pool instances for specific attribute sets, CASPs can enforce fine-grained compliance policies, such as the combination of nationality and age requirements. This ensures that only eligible users are granted access.

a) Alice's Scenario: Nationality + Occupation Verification: Alice seeks to access a structured financial product offered by a CASP. The CASP mandates that users meet two compliance requirements: they must not be from a high-risk country and must be financial practitioners (e.g., working in the finance sector). Alice submits her KYC information, which includes her nationality and occupation, to a CEX along with her blockchain address addr_A. After verifying Alice's information, the CEX issues a digitally signed `approval` certifying that Alice satisfies the nationality and occupation requirements. Alice deposits her approval and funds into a PRIVKYC pool designed to validate these attributes. Once validated, Alice can then use the new compliant address addr_B to interact with the target CASP.

b) Bob's Scenario: Age + Credit Rating Verification: Bob intends to use a lending service provided by a CASP. To ensure compliance, the CASP requires users to meet two criteria: they must be older than 20 and have a financial credit rating of A or higher. Bob submits his KYC information, including proof of age and a certified credit rating, along with his blockchain address addr_C to the CEX. The CEX verifies that Bob meets the requirements and issues a digitally signed `approval`, which certifies that Bob's age is greater than 20 and his credit rating is above A. Bob then deposits this approval and funds to a PRIVKYC pool configured to validate these attributes. After successful validation, Bob can use his new compliant address addr_D to interact with the lending CASP. REGKYC 's support for attribute combinations enables the integration of traditional financial credit ratings into the blockchain system, allowing CASPs to evaluate user creditworthiness based on established financial systems. This enhances risk assessment, maintains privacy and compliance, and expands CASPs' accessibility to more compliant users.

c) Formalized Process: We express the formalized attribute combination by taking Bob's scenario as an example.

- Setup: System parameters are initialized. A PRIVKYC pool is created to handle the compliant attribute combinations of $\mathsf{Attr} = \{\text{age} > 20, \text{credit rating} > A\}$.
- Approval from CEX: Bob provides addr_C,KYC information, and Attr to the CEX. The CEX verifies Bob's KYC information and checks the validity of Attr. Once verified, the CEX generates a digital signature using its private key $\mathsf{sk}_{\mathsf{CEX}}$. This results in $\sigma_{\mathsf{addr}_C} \leftarrow \mathsf{Sign}(\mathsf{sk}_{\mathsf{CEX}}, H(\mathsf{KYC})||H(\text{age} > 20)||H(\text{credit rating} > A)||\mathsf{addr}_C)$ as Bob's approval.
- Deposit into the PRIVKYC pool: Bob computes $\mathsf{cm} = H_{2p}(k_{dep}||r_{dep})$ and submits tx_{dep} that includes $(H(\mathsf{KYC}), H(\mathsf{Attr}), \mathsf{addr}_C, \sigma_{\mathsf{addr}_C}, \mathsf{cm})$. The pool then performs verfication $\pi_{sig}.\mathsf{Verify}(\mathsf{pk}_{\mathsf{CEX}}, H(\mathsf{KYC})||H(\text{age} > 20)||H(\text{credit rating} > A)||\mathsf{addr}_C, \sigma_{\mathsf{addr}_C})$ to verify the approval. If valid, cm is added to the pool's Merkle tree.
- Withdraw from the PRIVKYC pool: Bob computes the nullifier nf_{wd} and uses DRC's public key to encrypt $\mathsf{cm}||H_p(r_{dep})$. Then he generates a proof π_{wd} and composes $(\mathsf{addr}_D, \mathsf{enc}, \mathsf{nf}_{wd}, \mathsf{Root}_{dep}, \pi_{wd})$ as the input data to issue the withdrawal transaction tx_{wd}. The pool permits a withdrawal to Bob's fresh address if the proof π_{wd} is valid.
- Interaction with CASPs: Bob uses addr_D to interact with the lending CASP. The CASP validates tx_{wd} to ensure that it is dealing with a verified and compliant user with a compliant attribute combination of age > 20 and credit rating $> A$.
- Deanonymization (if required): If addr_D is flagged for malicious activity, the DRC decrypts enc to retrieve cm. The DRC identifies addr_C by matching cm in the pool's records and requests the CEX to provide Bob's KYC information.

Overall, REGKYC's ability to support the verification of attribute combinations offers flexibility for CASPs. It allows CASPs to design attribute requirements tailored to the specific nature of the services they provide, ensuring that compliance measures align closely with their operational needs. Furthermore, this adaptability not only addresses current compliance needs but also positions CASPs to proactively meet stricter regulatory demands that may arise.