Beyond the Blockchain Address: Zero-Knowledge Address Abstraction

Sanghyeon Park
Seoul National University
Seoul, Republic of Korea
lukepark@snu.ac.kr

Jeong Hyuk Lee
Hanyang University
Seoul, Republic of Korea
ahoo791@hanyang.ac.kr

Seunghwa Lee
Kookmin University
Seoul, Republic of Korea
ttyhgo@kookmin.ac.kr

Jung Hyun Chun
Hanyang University
Seoul, Republic of Korea
jungdev@hanyang.ac.kr

Hyeonmyeong Cho
UNIST
Ulsan, Republic of Korea
heyitsme@unist.ac.kr

MinGi Kim
Sungkyunkwan University
Seoul, Republic of Korea
lupin1001@skku.edu

Hyun Ki Cho
Sogang University
Seoul, Republic of Korea
blockchain@sogang.ac.kr

Soo-Mook Moon
Seoul National University
Seoul, Republic of Korea
smoon@snu.ac.kr

***Abstract*—The integration of web2 identities into the web3 blockchain has been a challenging area of research. Existing methods use a mapping approach, linking web2 identities to blockchain addresses, resulting in limitations such as identifier fragmentation across different blockchains.**

In this paper, we propose a novel solution, zero-knowledge Address Abstraction (zkAA), which eliminates the need for mapping and enables the direct use of web2 identity in the blockchain. Our solution leverages zero-knowledge proofs verified by smart contracts, allowing users to operate in the blockchain using their web2 identity without any address-related restrictions. The identity used in zkAA is based on the certificate issued by a trusted institution and registered as a hidden form in the blockchain.

Our zkAA solution is implemented as a smart contract on top of the Ethereum blockchain, providing easy integration with existing systems without the need for a hardfork. We conducted an implementation and evaluation of zkAA on Ethereum, where we found that the cost of registering an abstract identity is $7.66 and the cost of publishing an abstracted transaction is $4.75. On Polygon, the cost is much more favorable, with a cost of $0.02 for registering and $0.01 for publishing, as of January 13, 2023. In addition, we found that the time taken for generating proofs for registering is approximately 9.57 seconds and the time cost for publishing is approximately 3.49 seconds on an average local machine. This suggests that the client has adequate time to generate proofs within a single block.

***Keywords*—identity management; blockchain; decentralized applications; privacy-preserving; zero-knowledge proofs;**

I. Introduction

The rapid growth of blockchain technology has given rise to the proliferation of decentralized applications (dApps). However, despite their decentralized nature, dApps often require centralized authentication methods, such as social login or Know-Your-Customer (KYC) processes, for efficient operation and to mitigate malicious activities, such as Sybil attacks [12]. In integrating web2 identities into the web3 blockchain, previous approaches have encountered several challenges. These approaches center around mapping external identities to blockchain-specific addresses, which can result in difficulties in effectively utilizing external identities on the blockchain and increased complexity in identification from dApps. This becomes particularly problematic in multi-chain dApps, where the management of multiple blockchain-specific addresses becomes a challenge, leading to fragmented digital identities for both service providers and users. The use of blockchain-specific addresses also results in the public linkage of external identities and addresses on the blockchain, which can result in privacy leaks given the transparency of the blockchain and its ability to trace the full history of an account distinguished by its address.

A. Address Abstraction

To address these challenges, this paper proposes the Address Abstraction (AA) scheme, which allows for the direct use of external identities $cert$, such as JSON Web Tokens (JWTs) [20], as blockchain identifiers, without mapping them to blockchain-specific addresses. In the AA scheme, such as the hash output $\mathcal{H}(cert)$, is used as the web3 identifier. This non-blockchain-specific identity scheme enables the creation of a unified multi-chain identity, regardless of the signature and address system, and enables users to leverage the benefits of both web2 and web3 technologies seamlessly.

However, straightforward implementations of the AA scheme through hash-only schemes or Merkle tree approaches can result in privacy and unsustainable issues, as they lack privacy-preserving cryptography. To address these challenges, this paper proposes zkAA, a zero-knowledge Address Abstraction solution that leverages zero-knowledge proofs [16], [17] for identity verification on the blockchain. In a nutshell, zkAA introduces a novel approach to incorporating external identities into the blockchain, using $cert$s and zero-knowledge proofs, without mapping to blockchain addresses. This eliminates the need for private key storage, provides an effective solution for identity management, and enables the development of seamless multi-chain and web2-web3 hybrid dApps. Table I provides a comparison between traditional blockchain addresses and the proposed identifier, denoted as $\mathcal{H}(cert)$.

TABLE I: Comparison of Address and $\mathcal{H}(cert)$ as Identifiers on the Blockchain

Identifier	Definition	Purpose	Issuing Authority
Address	A public identifier employed for conducting on-chain transactions.	To furnish a means of identifying and managing digital assets in a blockchain.	Generated by individual users, not issued by a central authority.
$\mathcal{H}(cert)$ in zkAA	A solution utilizing certificates $certs$ as a form of identification within a blockchain.	To tackle the challenges associated with mapping existing external identities into blockchain-specific identities.	$certs$ can be issued and verified by trusted Web2 certificate authorities.

Identifier	Underlying Cryptography	Secret	Chain-Specific	Cost
Address	Public Key Cryptography (& Hash)	Private Key	Yes.	Typically low, with only minimal fees incurred for transaction processing.
$\mathcal{H}(cert)$ in zkAA	Zero-Knowledge Proofs & Hash	Certificate, such as JSON Web Token	No. Identifiers remain consistent across multiple chains.	The cost of implementation is contingent upon the specifics of the implementation, but, in general, the use of ZKPs increases the cost.

The implementation of zkAA is achieved through the use of smart contracts on top of existing blockchain infrastructure, making it easy to integrate with existing systems and applications and eliminating the need for a hardfork process.

B. Contributions

The focus of this paper is to present a unique approach to address abstraction in blockchain systems, utilizing zero-knowledge proofs. The key contributions of this paper are as follows:

- We introduce a new identity system, address abstraction, to overcome the limitations of traditional blockchain-specific identity and signature systems.
- We present a novel scheme for address abstraction named zkAA, zero-knowledge Address Abstraction, which leverages zero-knowledge proofs to offer strong privacy guarantees and practicality.
- The paper demonstrates the integration of zkAA with multi-modal decentralized applications, including multi-chain and web2-web3 hybrid systems, to provide a seamless user experience in blockchain systems.
- The paper presents the implementation and evaluation of zkAA on the Ethereum [46], along with a comprehensive analysis of its performance and efficiency.

The organization of this paper is as follows: In Section II, a comprehensive background is presented to provide a clear understanding of the concepts involved in the material presented. In Section III, the necessary notations and definitions are defined to ensure consistency throughout the paper. In Section IV, the design and implementation of a naïve solution for address abstraction are described and its limitations are analyzed. In Section V, our proposed solution, Zero-knowledge Address Abstraction (zkAA), is introduced. zkAA leverages the power of zero-knowledge proofs to overcome the limitations of the naïve solution. In Section VI, a sequence diagram is provided to outline the potential applications of zkAA. In Section VII, experiments are conducted to evaluate the performance and overhead of zkAA. In Section VIII, the potential for future work and enhancement of zkAA is discussed. In Section IX, related work in the field of identity management in web3 is reviewed. Finally, in Section X, our proposed scheme is summarized and its strengths are highlighted.

II. Background

A. JSON Web Tokens

JSON Web Tokens (JWTs) [20] are widely used open standards for securely transmitting claims between parties. They are commonly used for authenticating users and passing user information between systems, especially as access tokens in API authentication. JWTs are signed using public/private key cryptography algorithms to ensure their authenticity and integrity. Elliptic curve-based algorithms, such as EdDSA [2], [21], are particularly secure and efficient for this purpose. The signature created through this process confirms that only the entity holding the private key could have signed the token, thereby certifying the authenticity of the source of the JWT.

This paper presents an effort to integrate a certificate, such as JWT, into the blockchain ecosystem, without mapping to blockchain addresses. The utilization of certificates issued and verified by trusted institutions offers a convenient and effective solution for identity verification, particularly in scenarios where mapping to blockchain addresses is not necessary.

B. Blockchain and Smart Contracts

The advent of blockchain technology and smart contracts has established itself as a transformative solution for secure and efficient data storage and contract execution. A blockchain is a distributed ledger that employs cryptographic techniques to securely record transactions across a network of nodes, where the integrity of the data is maintained through the consecutive creation of blocks that form a chain, making it resistant to modification [30]. One of the leading blockchain platforms is Ethereum [46], which provides an infrastructure for the creation and operation of smart contracts. The Ethereum Virtual Machine (EVM) operates in a decentralized environment, executing smart contracts through a global network of public nodes and using the cryptocurrency Ether (ETH) as the medium of exchange for transaction fees and computational services. Smart contracts have also found applications in identity management, as demonstrated by decentralized identity managements II-C.

In this work, we introduce a new identity management system, named zkAA, that integrates the use of smart contracts to achieve privacy-preserving verification of web2 certificates. Our implementation is based solely on the functionality of

Ethereum Virtual Machine (EVM), allowing for seamless integration into existing blockchain networks without the need for a hardfork, thus increasing feasibility.

C. Decentralized Identity Management

Decentralized identity management refers to the utilization of blockchain technology to allow entities to create, store, and manage their own digital identity, without relying on centralized authorities or databases. Examples of decentralized identity management systems that run on the blockchain include Proof of Attendance Protocol (POAP) and SoulBound Tokens (SBTs). POAP [19] is a platform that leverages blockchain technology to create collectible tokens representing personal experiences, while SBTs [43] serve as nontransferable Non-Fungible Tokens (NFTs) that form the foundation of the Decentralized Society trend in web3.

However, these solutions are based on the web3 identifier, also known as the blockchain address. In order to directly use a web2 identity on a web3 platform, mapping between the two identities is necessary. Whereas in the case of our proposed identity system, address abstraction, the web2 identity is utilized directly on the blockchain, without the need for mapping to a web3 identity.

D. Account Abstraction

In the Ethereum blockchain, Account Abstraction is a technique that aims to unify External Owned Accounts (EOAs) and Contract Accounts (CAs). EOAs are traditional blockchain accounts controlled by a private key, enabling users to sign transactions and interact with the network. On the other hand, CAs do not have a corresponding private key but possess program codes and storage. Account Abstraction enables integration of the features of both EOAs and CAs, allowing for programmability features in EOAs and signing capabilities in CAs. This integration can be achieved either through a hardfork on the Ethereum network [44] or a contract-based wallet solution without the need for a hard fork [7].

In contrast, our proposed Address Abstraction is a new concept in the blockchain. Our proposed solution abstracts the user identifier from the conventional blockchain address to an abstracted unified identifier. This simplifies identity management and enhances inter-compatibility across various blockchains by enabling users to interact with the blockchain using a single credential, thereby eliminating the limitations imposed by the underlying blockchain address systems that may vary between different blockchains.

E. Zero-Knowledge Proofs

The field of cryptography and blockchain technology has seen a surge in interest in Zero-Knowledge Proofs (ZKPs) as a solution to privacy and security challenges. Zero-knowledge proofs were first introduced in [16] as a mechanism for verifying the authenticity of information without requiring the revelation of the underlying data. Since the initial introduction of zero-knowledge proofs, the field has undergone significant advancements, including the development of zero-knowledge Succinct Non-Interactive Arguments of Knowledge (zk-SNARKs) [17]. This technology allows for the efficient and succinct verification of information, making it an ideal solution for blockchain-based systems. A popular tool used in the development of zero-knowledge proofs is ZoKrates [13], a toolkit for writing, compiling, and executing ZKP-based smart contracts on the Ethereum blockchain.

The zero-knowledge protocol employed in this paper includes three functions: $\mathsf{zkp.Setup}$, $\mathsf{zkp.Prove}$, and $\mathsf{zkp.Verify}$.

- $crs \leftarrow \mathsf{zkp.Setup}(\mathcal{R})$: This function takes as input a relation $\mathcal{R}$ and returns a Common Reference Strings (CRSs), crs, which are public parameters for proof generation and verification.
- $\pi \leftarrow \mathsf{zkp.Prove}(crs, io; w)$: This function is utilized to generate a proof π based on the input values of crs, public input/output io, and private witness w. The proof is generated in such a manner that the witness w remains hidden during the verification process.
- $0/1 \leftarrow \mathsf{zkp.Verify}(crs, io, \pi)$: This function verifies the validity of a proof π by taking as input the crs, input/output io, and proof π. The function returns a binary outcome, either 0 or 1, indicating whether the verification was successful (1) or failed (0).

III. Definitions

A. Notation

In this paper, we use the following notations. The adversary is represented by $\mathcal{A}$ and its output is represented as $\mathcal{A}(x)$, where x is the input. The function $\mu(x)$ represents a negligible function, which approaches zero. The order-related symbol, $a \succ b$, is used to denote that a cannot occur before b. A web2 certificate, such as a JSON Web Token (JWT), is represented by $cert$, and it is assumed that a trusted third party provides the $cert$ corresponding to the user's private personal data pwd, such as a password. The public key of the trusted institution is denoted as pk. The function $\mathcal{H}$ is utilized to generate a universally consistent identifier id from the certificate $cert$. For example, this is achieved through the application of a cryptographic hash function, expressed as $id \leftarrow \mathcal{H}(cert)$. The identifier or identity of a specific user $user^k$ is represented as id^k. Let $\mathcal{M}$ be a set of all executable messages. The user request, which consists of $(target, funcsig, calldata, value)$, is denoted as $msg \in \mathcal{M}$. These parameters $(target, funcsig, calldata, value)$ make up the context of the user request. The function $funcsig$ is executed on the $target$ address through an internal transaction, with $calldata$ and $value$ ETH. Finally, two relations are defined: $\mathcal{R}_R$ for id registration and $\mathcal{R}_P$ for msg publication using id. The symbol $\perp$ indicates that a value is not considered.

B. Address Abstraction

The proposed concept of address abstraction refers to the abstraction of the identifier system in blockchain, which is

decoupled from the underlying blockchain platform. This approach allows for the registration and usage of identity to be independent of the specific characteristics and features of a given blockchain. In this work, we present a novel definition of address abstraction in relation to abstracted identity and abstracted transaction, which encompasses the functions: SETUP, CERTIFICATE, REGISTER, and PUBLISH. The function of REGISTER facilitates the registration of an identifier as an abstracted address, while the function of PUBLISH allows for the publishing of transactions using the abstracted identifier.

Definition III.1 (Address Abstraction)**.** *Let $\mathcal{R}_R$ and $\mathcal{R}_P$ be relations for registration and publication, respectively. The private data of the user, utilized in web2, is referred to as* pwd*. In order to achieve an abstracted identity id of $user$ and to be able to publish the abstracted transaction msg, the scheme must possess four key functions* (SETUP, CERTIFICATE, REGISTER, PUBLISH)*.*

- $(pp_R, pp_P) \leftarrow \text{SETUP}(\mathcal{R}_R, \mathcal{R}_P)$: This function generates the public parameters pp_R and pp_P for registration and publication, respectively. The inputs to the function are the relations $\mathcal{R}_R$ and $\mathcal{R}_P$.
- $(pk, cert) \leftarrow \text{CERTIFICATE}(pwd)$: This function enables a user to request a certificate from a trusted institution. The user submits their personal information, pwd, and the trusted institution returns a certificate, $cert$, and its public key, pk. The public key serves to identify the certificate authority and verify the validity of the certificate.
- $(0/1, id_{chain}) \leftarrow \text{REGISTER}(pp_R, id, pk, \pi_R)$: This function allows a user to register the abstracted identity, id, derived from the certificate $cert$ through the application of $\mathcal{H}$ (i.e., $id \leftarrow \mathcal{H}(cert)$) on a blockchain, $chain$. The inputs to the function include the public parameters pp_R, id, the public key pk, and a proof π_R that verifies the validity of id. It is not necessary for the verification of id to occur immediately. It is acceptable for the verification to occur prior to the first publishing of the abstracted transaction. The output of the function includes the validation status of the identifier id, where a value of 0 represents an invalid identifier and a value of 1 represents a valid one. Additionally, the function returns the blockchain-specific explicit identity, id_{chain}, which may or may not be equivalent to id, for supporting compatibility with other smart contracts on the same blockchain. The comprehensive information regarding id_{chain} can be found in Section V-C and VI.
- $(0/1, res_m) \leftarrow \text{PUBLISH}(pp_P, id, msg, m, \pi_P)$: This function allows the execution of a submitted request msg upon the provision of valid proof π_P about an identifier id, based on the public parameters pp_P. The input of a monotonic increasing counter m is required to ensure that the submitted msg is processed chronologically. The result of the execution of the m-th message is represented by res_m. If the execution is success, returns $(1, res_m)$, or $(0, \emptyset)$, otherwise.

The address abstraction must conform to two properties to ensure its functionality: **Unique Identifier** (III.2) and **Immutable Request** (III.3). These properties are crucial in maintaining the security and integrity of the address abstraction system.

Unique Identifier. A Unique Identifier is a property that ensures a user's identity is unique (Injectiveness), controlled only by the user who holds the secret (Unforgeability), and publicly verifiable (Correctness).

Definition III.2. *The system* (SETUP, CERTIFICATE, REGISTER, PUBLISH) *has unique identifier id^k for $user^k$, if id^k meets the* Injectiveness*,* Unforgeability*, and* Correctness*.*

$$\text{for all } (cert^x, cert^y),\ cert^x \neq cert^y \implies id^x \neq id^y \quad \text{(Injectiveness)}$$

$$\begin{aligned}&\text{for all adversaries } \mathcal{A},\\ &\Pr[\pi_P^{\mathcal{A}} \leftarrow \mathcal{A}(1^\kappa)^{O_{Prove}(\cdot), O_{Cert}(\cdot)};\\ &(s_P, \perp) \leftarrow \text{PUBLISH}(pp_P, id, msg, m, \pi_P^{\mathcal{A}}) :\\ &s_P = 1, (id, \pi_P^{\mathcal{A}}) \notin O_{Prove} \wedge cert \notin O_{Cert} \text{ s.t. } id = \mathcal{H}(cert)]\\ &\qquad \leq \mu(\kappa) \quad \text{(Unforgeability)}\end{aligned}$$

where $O_{Prove}(\cdot)$ is an oracle that can receive the any identifier id and a proof π_P for any unqueried π_P, and $O_{Cert}(\cdot)$ is an oracle that can generate a $cert$ for any pwd.

$$\begin{aligned}&\Pr[(pp_R, pp_P) \leftarrow \text{SETUP}(\mathcal{R}_R, \mathcal{R}_P);\\ &(pk, cert) \leftarrow \text{CERTIFICATE}(pwd); id \leftarrow \mathcal{H}(cert);\\ &(s_R, \perp) \leftarrow \text{REGISTER}(pp_R, id, pk, \pi_R);\\ &(s_P, \perp) \leftarrow \text{PUBLISH}(pp_P, id, msg, m, \pi_P) : s_R \wedge s_P = 1]\\ &\qquad = 1 \quad \text{(Correctness)}\end{aligned}$$

A user possessing a valid $cert$ can always generate a proof π_R and π_P for all executable messages $msg \in \mathcal{M}$ and a monotonically increasing counter m, to the verifier.

Immutable Request. This property means that requests made under a registered identity can only perform the specified actions and cannot be altered in transit. This is crucial in avoiding front-running attacks [26], where malicious actors observe the transaction and try to modify requests to gain an advantage.

Definition III.3. *Requests on the system* (SETUP, CERTIFICATE, REGISTER, PUBLISH) *is immutable, if the system ensures the constraints* Chronicle *and* Tamper Resistance *execution of msgs.*

$$\text{for all } m \geq 0, msg_{m+1} \succ msg_m \quad \text{(Chronicle)}$$

for $\forall msg' \neq msg,\ msg' \in \mathcal{M}$,
$\Pr[(pp_R, pp_P) \leftarrow \textsc{Setup}(\mathcal{R}_R, \mathcal{R}_P);$
$(pk, cert) \leftarrow \textsc{Certificate}(pwd); id \leftarrow \mathcal{H}(cert);$
$(s_R, \perp) \leftarrow \textsc{Register}(pp_R, id, pk, \pi_R);$
$msg' \leftarrow \mathcal{A}(pp_P, id, m, \pi_P);$
$(s_P, \perp) \leftarrow \textsc{Publish}(pp_P, id, msg', m, \pi_P) : s_P = 1]$
$\leq \mu(\kappa)$ (Tamper Resistance)

C. Privacy-Preserving Address Abstraction

Privacy-Preserving. The property of a system that protects users' personal information, such as JWT, from public disclosure. It guarantees that the information is kept confidential and does not reveal sensitive details to unauthorized parties.

IV. Naïve Approach to Address Abstraction

Address abstraction, we proposed, is a novel concept in blockchain that aims to simplify identity management by abstracting away the underlying cryptography. The fundamental idea behind address abstraction is to utilize the web2 certificate $cert$ to generate a unique, secure, and concise identifier $id \leftarrow \mathcal{H}(cert)$ for users on the blockchain. In this section, we will introduce two naïve approaches to address abstraction, namely the hash-based approach and the Merkle tree approach.

A. Hash-based Approach

In this section, we present a hash-based scheme for identifying users in a blockchain. This approach leverages the properties of cryptographic hash functions, such as SHA256 [33], [40], to generate a unique identifier. Our method uses a certificate $cert$ and a hash function $\mathcal{H}$ to create a unique identifier, id denoted as $id = \mathcal{H}(cert \parallel msg \parallel r)$, for each user. The identifier is formed by the hash of the concatenation of $cert$, a message msg, and a random number r for added security. Our approach is similar to the commit-and-reveal scheme, where the user first commits to a value by submitting its hash representation id and later reveals the original values $cert$, msg, and r.

The hash-based approach consists of four functions according to the definition of address abstraction: Setup, Certificate, Register, and Publish. These functions are described below:

0) $(\perp, \perp) \leftarrow \textsc{Setup}(\mathcal{R}_R, \mathcal{R}_P)$: The hash-based scheme does not require the generation of any additional public parameters.
1) $(pk, cert) \leftarrow \textsc{Certificate}(pwd)$: A user obtains a signed certificate, such as JSON Web Token, from a trusted institute by providing their private information pwd. The certificate can be verified using the public key pk.
2) $(0/1, id_{chain}) \leftarrow \textsc{Register}(\perp, id, pk, \perp)$: The user calculates a one-time identifier $id = \mathcal{H}(cert \parallel msg \parallel r)$. The validity of id is verified later, just before the first execution of Publish in the same transaction.
3) $(0/1, res_m) \leftarrow \textsc{Publish}(\perp, id, msg, \perp, (cert, msg, r))$: The user reveals the preimages ($cert$, msg, r) as proof, and the validity of id is calculated by executing the hash function $\mathcal{H}$ on the smart contract. The validity of $cert$ is also confirmed by checking it against the public key pk saved from Register. If both Register and Publish return a result of 1, msg is executed and the result res_m is returned. Note that the hash-only scheme is designed for one-time use, so the monotonic increasing counter m is not necessary.

The hash-based approach satisfies the properties of address abstraction, unique identifier and immutable request, as follows:

- **Unique Identifier Generation.** In this hash-based approach, a unique identifier, id, is generated as $id = \mathcal{H}(cert \parallel msg \parallel r)$. The original certificate is not disclosed publicly during the Register. During Publish, only the entity that possesses the original certificate can provide its preimage and claim the use of the corresponding identity. The certificate is recorded on the blockchain in a public manner, making it available for verification by any party or contract at any time.
- **Immutable Request Execution.** The user request msg is executed exactly as it was recorded during the Register. The blockchain records the request message in the form of id, ensuring that any modifications to the original request would compromise the validity of the id and thus prevent its execution.

However, it is important to note that this approach has its limitations. Once the certificate is revealed and recorded in the blockchain, it cannot be used again in the future, as its preimage is publicly known. This method is only suitable for cases where the identity does not need to be kept private and can only be used for a single transaction. Additionally, the execution message is enrolled at the time of registration and cannot be changed during the publication phase. To execute a different request, the user must go through the registration phase again to call Register, incurring additional time and cost overhead for both the user and service provider.

B. Using Merkle Tree

Figure 1 presents a Merkle tree-based approach as an alternative to implement address abstraction. The $root$ of a Merkle tree and its accompanying Merkle proofs are employed to execute a requested message msg at a specified order, $1 \leq m \leq n$, where n is the maximum number of messages. The identifier in this approach, with a finite lifetime of n, is the $root$ of the Merkle tree.

The Merkle tree approach to address abstraction is comprised of four functions: Setup, Certificate, Register, and Publish. The functions Setup and Certificate work similarly to the hash-only scheme, with the main differences being in the Register and Publish functions.

2) $(0/1, id_{chain}) \leftarrow \textsc{Register}(\perp, id, pk, \perp)$: The user submits the $root$ of the Merkle tree, which consists of a series of leaves including the certificate $cert$, the requested message msg_m, and the random number r_m

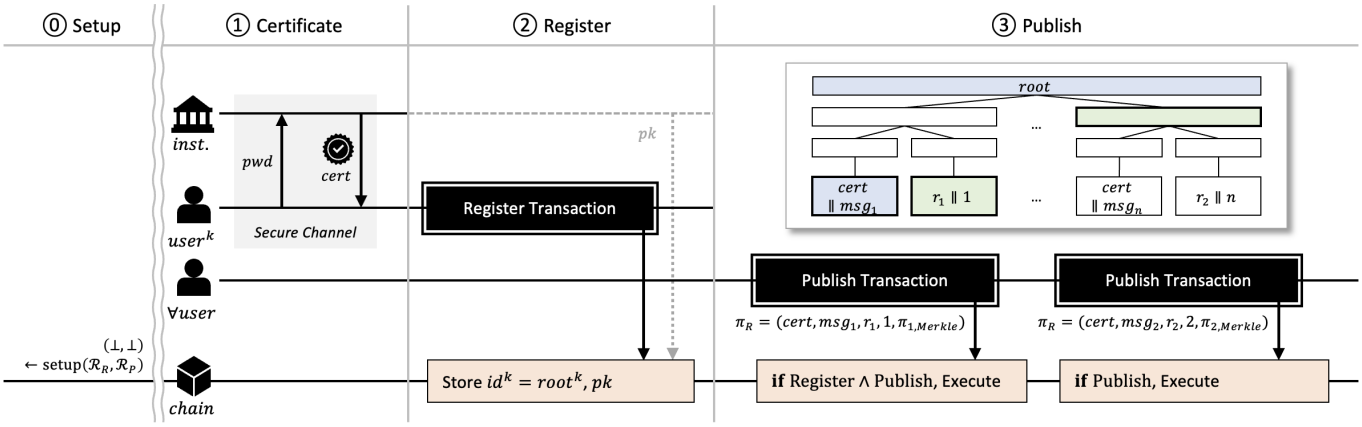

Fig. 1: Merkle Tree Address Abstraction

concatenated with the order of the message m. The leaves of the Merkle tree are organized in the following order: $[cert \parallel msg_1, r_1 \parallel 1, cert \parallel msg_2, r_2 \parallel 2, ..., cert \parallel msg_n, r_n \parallel n]$. The random number r_m serves as a security feature to randomize the input data. In this step, the user submits all the messages to be executed in the future PUBLISH step in a specified order. The validity of the identity id is verified just before the first execution of PUBLISH in the same transaction.

3) $(0/1, res_m) \leftarrow$ PUBLISH$(\perp, id, msg, m, (cert, msg_m, r_m, m, \pi_{m,Merkle}))$: The user reveals the certificate $cert$, the requested message msg_m, the corresponding random number r_m, the order of the message m, and the Merkle proof $\pi_{m,Merkle}$. In the first execution of this step, when $m = 1$, the validity of the certificate $cert$ is verified using the public key pk to ensure that it was signed by a valid trusted institute. The contract then verifies the Merkle proof $\pi_{m,Merkle}$ by computing hash functions and comparing the results with the registered $root$ of the Merkle tree. If both REGISTER and PUBLISH return a result of 1, the first message msg_1 is executed and the result res_1 is returned. For subsequent executions of PUBLISH with $m > 1$, if the function returns a result of 1, the message msg_m is executed and the result res_m is returned.

The Merkle tree approach satisfies the following two properties of address abstraction:

- **Unique Identifier.** The $root$ of the Merkle tree is a unique identifier for the entity that holds the original $cert$. During the PUBLISH, the entity presents the $cert$, the corresponding random number r_m, the execution message msg, and the Merkle proof $\pi_{m,Merkle}$ for the order of m. These data are verified by the contract, and upon verification, the entity is allowed to use the identity represented by the $root$. The data is recorded in the blockchain and can be publicly verified by anyone or a contract after the corresponding round m data is revealed.
- **Immutable Requests.** The execution messages, or msgs, are enrolled during the REGISTER and are executed in chronological order, determined by the monotonic increasing nonce m. This ensures that the execution messages cannot be altered and are executed in the same manner as they were enrolled at the time of the $root$ registration.

However, it is important to note that this method, like the hash-based approach, also reveals the $cert$, which makes it only suitable for cases where the identity represented by the $cert$ does not need to be kept private. Additionally, the execution messages must be enrolled during the time of the root registration and cannot be changed during the PUBLISH step. Changing the execution messages would incur significant overhead in terms of time and cost, as the user would need to re-register. Furthermore, the identifier $root$ can only be used a finite number of times, n.

In conclusion, the hash-based approach enables single-use abstracted identity. Meanwhile, the Merkle tree approach employs a Merkle tree structure, with a limited number of identity uses as determined by the number of n. Despite their merits, both these approaches can present privacy and scalability concerns due to the lack of privacy-preserving features.

To tackle these challenges, we present our solution, the zero-Knowledge Address Abstraction scheme, which utilizes zero-knowledge proofs for identity verification on the blockchain, as detailed in the next Section V.

V. ZERO-KNOWLEDGE ADDRESS ABSTRACTION

The management of identities in decentralized systems, particularly blockchain networks, presents a challenge in balancing privacy and security with efficiency and usability. The naïve approaches, hash-based or Merkle tree based address abstraction methods, fail to provide privacy guarantees as the contents of the certificates are disclosed during the PUBLISH phase. Furthermore, they are limited in their application, as they can only be used for single-use or a finite number of

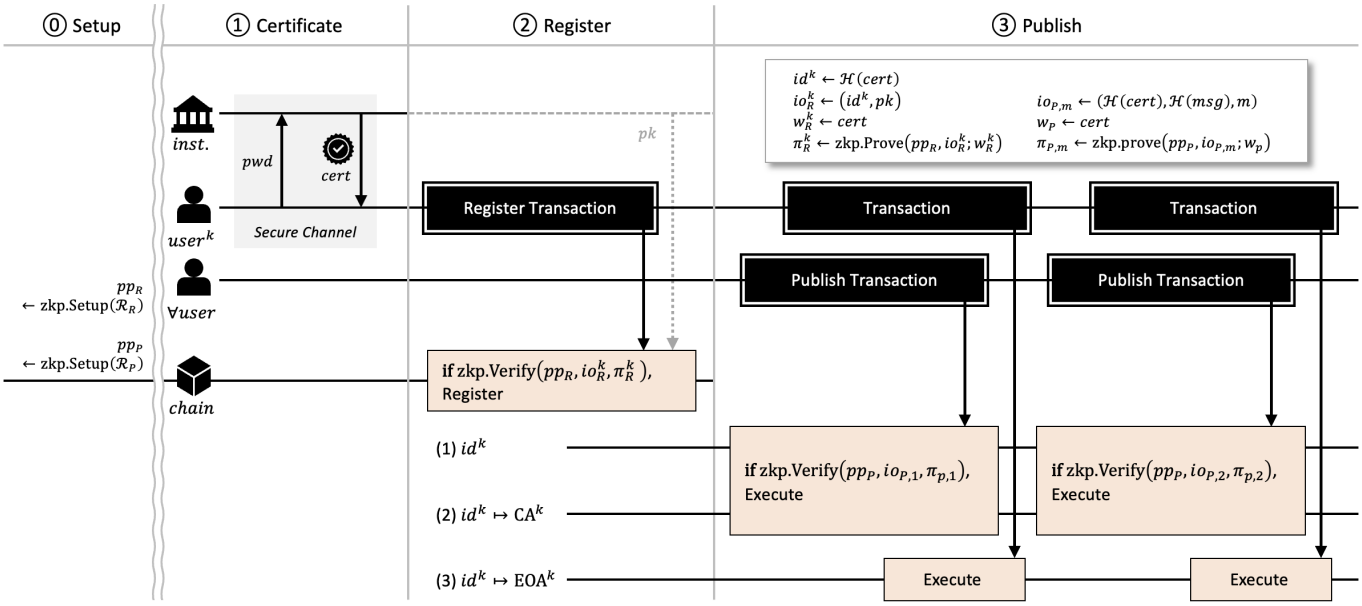

Fig. 2: Zero-Knowledge Address Abstraction

usages on a registered identifier, rendering them unsuitable for unlimited identity usage.

To address these limitations, a privacy-preserving approach is necessary. Our proposed solution leverages zero-knowledge proofs to offer a solution that verifies the validity of a $cert$, while maintaining its privacy. The use of a zero-knowledge address abstraction scheme ensures the preservation of privacy for the $cert$ contents and enables unlimited usage of identities.

A. Design of zkAA

The zkAA protocol is a specific implementation of address abstraction that leverages the power of zk-SNARKs to provide privacy for registering identities on a blockchain network. In the zkAA system, the user's identity is represented by a $cert$ that is hashed to create a unique identifier $id = \mathcal{H}(cert)$. $\mathcal{H}$ is the hash function. The proof π is created to verify the authenticity of the $cert$, thereby proving that the user holds the preimage corresponding to the $\mathcal{H}(cert)$, without disclosing the contents of the $cert$. This enhances privacy as the contents of the $cert$ remain confidential throughout the protocol. The proof is verified on the blockchain, providing a decentralized and tamper-proof method for managing and verifying user identities. The use of zk-SNARKs in the proof process makes the zkAA protocol efficient and succinct, making it suitable for use in the blockchain.

The zkAA protocol consists of four functions by the definition of address abstraction: SETUP, CERTIFICATE, REGISTER, and PUBLISH, which are depicted in Figure 2 and are based on the zero-knowledge protocol described in Section II-E.

0) $(pp_R, pp_P) \leftarrow \text{SETUP}(\mathcal{R}_R, \mathcal{R}_P)$:
The SETUP function generates the common reference strings (crs_R, crs_P), as (pp_R, pp_P), which are used in the identity verification and hash verification processes, respectively. The two trusted setups, $crs_R \leftarrow \mathsf{zkp.Setup}(\mathcal{R}_R)$ and $crs_P \leftarrow \mathsf{zkp.Setup}(\mathcal{R}_P)$, are conducted and the public parameters are used for identity verification in the REGISTER phase and hash verification in the PUBLISH phase.
1) $(pk, cert) \leftarrow \text{CERTIFICATE}(pwd)$:
The user requests a $cert$ from an institute using their personal web2 data pwd.
2) $(0/1, id_{chain}) \leftarrow \text{REGISTER}(pp_R, id, pk, \pi_R)$:
The user calculates the hash of their certificate as identifier $id = \mathcal{H}(cert)$ and creates a proof $\pi_R \leftarrow \mathsf{zkp.Prove}(pp_R, (id, pk); cert)$ to verify that the id was derived from the $cert$ and that the $cert$ was signed by the institute, without revealing the actual $cert$. The user then registers the id and proof π_R on the blockchain, and the zkAA contract verifies the hash and proof through $\mathsf{zkp.Verify}(pp_R, (id, pk), \pi_R)$ to ensure their validity. If the verification is successful, the contract adds the id to its list of valid identities. In the zkAA protocol, the id is immediately finalized after the completion of the REGISTER function.
3) $(0/1, res_m) \leftarrow \text{PUBLISH}(pp_P, id, msg, m, \pi_P)$:
The user can publish abstracted transactions from any account they choose, as long as they have a valid certificate $cert$ and can calculate its corresponding hash id, which must match the registered id on the blockchain. The abstracted transaction includes a proof $\pi_{P,m} \leftarrow \mathsf{zkp.Prove}(pp_P, (id, \mathcal{H}(msg), m); cert)$ that the zkAA contract verifies through $\mathsf{zkp.Verify}(pp_P, (id, \mathcal{H}(msg), m), \pi_{P,m})$ to ensure that the sender is the legitimate owner of the identity represented by id and that the message msg and its sequence m are valid. The requested message is executed

Algorithm 1 ② REGISTER

pre-defined registration type $\tau \in \{\tau_1, \tau_2, \tau_3\}$

Require:
pp_R, $cert$, institute's public key pk,
procedure CALLREGISTER^k(pp_R, $cert$, pk)
  $id^k \leftarrow \mathcal{H}(cert)$
  $\pi_R^k \leftarrow$ zkp.prove$(pp_R, (id, pk); cert)$
  call REGISTER(pp_R, id, pk, π_R^k)

Require: pp_R, id, pk, π_R
Ensure: $0/1, id_{chain}$
transaction REGISTER(pp_R, id, pk, π_R)
  `assert`(id has not yet been registered)
  `assert`(zkp.Verify$(pp_R, (id, pk), \pi_R)$)
  register id within zkAA contract
  set $nonce_{id} \leftarrow 0$
  switch τ **do**
    case τ_1 ▷ (id)
      return $(1, id)$
    case τ_2 ▷ $(id \mapsto$ CA$)$
      create contract wallet controlled by id
      map id into address of contract wallet CA
      return $(1,$ address(CA)$)$
    case τ_3 ▷ $(id \mapsto$ EOA$)$
      create NFT contains id
      give ownership to `msg.sender`
      return $(1,$`msg.sender`$)$
  return $(0, \perp)$

and the result res_m is returned if the proof $\pi_{P,m}$, m, and msg are valid.

REGISTER. The pseudocode for the REGISTER is depicted in Algorithm 1. The REGISTER is a crucial step in the zkAA system as it verifies the validity of the identities recorded in the blockchain. During this phase, three verifications are performed: verification of uniqueness, verification of hash, and verification of $cert$ signatures. The system checks that each identity has a unique representation in the system by verifying that the same value for id is not already registered in the blockchain. Then, the system verifies that id is indeed a hash made from the original $cert$ and that the $cert$ has been signed by the institute through a zero-knowledge proof, without revealing the $cert$ itself. The public key pk of the institute is used to verify about signature on-chain. In this paper, it is assumed that the correct public key is registered through reliable Oracle, governance, or majority voting. If the above verifications are successful, the id is registered in one of three ways: (1) registering id as a standalone value, (2) mapping the smart contract wallet with id, or (3) NFT-izing id and assigning its ownership to an Externally Owned Account (EOA). In the case of options (1) and (2), a $nonce$, a unique monotonic increment counter, is initialized to zero for

Algorithm 2 ③ PUBLISH

pre-defined registration type $\tau \in \{\tau_1, \tau_2, \tau_3\}$

Require:
pp_P, $cert$, nonce m,
$msg \leftarrow (target, funcsig, calldata, value)$
procedure CALLPUBLISH(pp_P, $cert$, msg, m)
  $id \leftarrow \mathcal{H}(cert)$
  $\pi_{P,m} \leftarrow$ zkp.Prove$(pp_P, (id, \mathcal{H}(msg), m); cert)$
  if $\tau \in \{\tau_1, \tau_2\}$ **then**
    call PUBLISH(pp_P, id, msg, m, $\pi_{P,m}$)
    if $msg.value > 0$ **then**
      Send $msg.value$ `ETH` to `zkAA` contract
  else ▷ $(id \mapsto$ EOA$)$
    execute msg through normal transaction

Require: pp_P, id, msg, m, π_P
Ensure: 0/1, res_m
transaction PUBLISH(pp_P, id, msg, m, π_P)
  `assert`(id must have been registered)
  `assert`$(m == nonce_{id} + 1)$
  `assert`(zkp.Verify$(pp_P, (id, \mathcal{H}(msg), m), \pi_P)$)
  update $nonce_{id} \leftarrow m$
  switch τ **do**
    case τ_1 ▷ (id)
      $res \leftarrow$ **execute** msg with identifier id
      return $(1, res)$
    case τ_2 ▷ $(id \mapsto$ CA$)$
      $res \leftarrow$ **call** `ExecuteMsg`(msg) of CA
      return $(1, res)$
  return $(0, \perp)$

Require: msg
Ensure: execution result of msg
function EXECUTEMSG(msg)
  `assert`(`msg.sender` == address(zkAA contract))
  $res \leftarrow$ **execute** msg
  return res

each identity managed by the smart contract to maintain the sequence of abstracted transactions. The information regarding the details can be found in Subsection V-C and VI. In the zkAA scheme, the validation of the $cert$ occurs during the REGISTER, as opposed to the naïve address abstraction implementations where $cert$ validation occurs during the PUBLISH. This forward-awareness of $cert$ authenticity offers the added advantage of allowing dApps to provide user-specific services that rely on zkAA-driven identities in advance.

PUBLISH. The pseudocode for the PUBLISH is presented in Algorithm 2. The user must produce evidence of their knowledge of the original preimage, which must correspond to the registered hash value id^k in the contract. Upon successful verification of the proof as true, the actions specified in msg will be executed. A non-zero quantity of the native asset

($value$) may also be required to be included in the abstracted transaction. The presence of msg and m within a proof $\pi_{R,m}$ serves to safeguard the intended actions against tampering, including potential front-running attacks [26], even if the proof were to become public in the mempool. In addition, to generate a valid proof, the $nonce$ must be initiated with a value of 0 and incremented by 1 for each subsequent transaction. The $nonce$ must be m after m-th abstracted transaction. In the case of type 3, as specified in line 9 of Algorithm 2, the user executes normal transactions rather than abstracted transactions. As a result, this approach is more cost-efficient compared to the cases of types 1 and 2.

B. Properties of zkAA

The zero-knowledge Address Abstraction (zkAA) is a privacy-preserving scheme for address abstraction, privacy-preserving address abstraction, that offers unique and distinguishable identities for users, with immutability and flexibility in the requests. Unlike previous methods, the original certificate is kept confidential throughout the entire process, providing a secure and tamper-proof identifier that can be used repeatedly without being consumed.

The zkAA process includes two essential features of address abstraction, **Unique Identifier** and **Immutable Requests**, and one feature for privacy-preserving address abstraction, **Privacy-Preserving**. In addition, three additional convenience features **Unlimited Usage**, and **Message Flexibility** are provided. In the following, we provide a detailed discussion of these five properties:

- **Unique Identifier.** The zkAA process generates a unique identifier, $\mathcal{H}(cert)$, that represents a user's identity. This identifier is ensured to be unique and distinguishable due to the privacy-preserving nature of the approach, as the original $cert$ is known only to the user (and the trusted institute). The validity of the identifier can be publicly verified through proofs π_R and π_P, ensuring that anyone can confirm that it represents a legitimate identity without revealing the original $cert$ data.
- **Immutable Request.** The m-th user request, msg, is included in the proof $\pi_{P,m}$, making it immutable and resistant to tampering. The sequential number $nonce$ also ensures that the abstracted transactions are executed in the correct order, without any changes or missing in the sequence.
- **Privacy-Preserving.** Privacy is a key aspect of the zkAA approach, as it is designed to protect users' personal information from public disclosure. The approach guarantees that public information does not reveal the original $cert$, both now and in the future.
- **Unlimited Usage.** The zkAA approach offers unlimited usage, allowing the unique identifier to be used repeatedly without being consumed. This sets it apart from other techniques such as hash-only or Merkle tree methods, which are limited in their use.
- **Message Flexibility.** The user request, msg, can be incorporated into the PUBLISH without prior registration, providing a high level of flexibility in the requests. This reduces the overhead at changing msg.

The zkAA offers a suitable and secure solution for privacy-preserving address abstraction, through the use of a zero-knowledge protocol. Moreover, the approach provides a unique identifier that can be used for unlimited purposes, with flexible messages.

C. Identifier Types of zkAA

Our paper proposes the use of $cert$-based encoded identifiers for improved identity management. The identifier is created by taking the hash of a $cert$, $\mathcal{H}(cert)$, which solves the issue of address variability across different blockchains and web2 platforms by providing a unified solution.

The REGISTER of the zkAA protocol includes three distinct methods for creating user identities: hash, contract wallet, and NFT. Each method has its own benefits and drawbacks, which are explained below.

1) Hash: In this method, the hash of the $cert$ is submitted to the zkAA contract on the blockchain, and the resulting $\mathcal{H}(cert)$ serves as the unique identifier for the user both implicitly as id and explicitly as id_{chain}. This allows for abstracted, anonymous transactions that can be signed from any ephemeral address by a valid user.

2) Contract Wallet: A new contract wallet is created on the blockchain and linked to the $\mathcal{H}(cert)$ identity, hence id_{chain} is the address of the contract wallet, denoted as $address(\text{CA})$. This approach offers compatibility with existing dApps by using a traditional address system but sacrifices anti-traceability due to the contract wallet's address. The contract wallet is controlled by proof π_P, and the abstracted transaction can be performed from any ephemeral address. This method as ever uses the same secret $cert$ across all blockchains, eliminating the need to store individual private keys for each address. In order to forward a request to a $target$, the contract wallet must implement the `ExecuteMsg`(msg) function, which can be invoked by the zkAA contract as described in Algorithm 2, line 23.

3) NFT: The user can create a Non-Fungible Token (NFT) to represent the $\mathcal{H}(cert)$ identifier, which can then be linked to a specific Externally Owned Account (EOA). The explicit identifier id_{chain} is the same as the user's address. This method, known as NFT-izing id, resembles the traditional mapping approach for integrating web2 identities into web3. Unlike the hash and contract wallet methods, the user can interact with decentralized applications using traditional transactions instead of abstracted transactions that require zero-knowledge proofs. However, this method requires certification verification to be performed within the dApp-specific, which can lead to an additional and explicit flow by the dApp. This method offers a trade-off between cost and convenience, as it incurs lower transaction costs compared to the hash and contract wallet methods. However, it also shares the limitations of the mapping approach in facilitating the seamless integration of decentralized applications, as discussed in Section I.

In each of these methods, the dApp requires a different flow for implementation. The second method, using the address of the contract wallet, can be employed without any modification to the existing dApp. However, the first method, which is based on $\mathcal{H}(cert)$, necessitates careful consideration during the design of the dApp to ensure that the identifier is properly recognized. The third method, NFT-izing id, may be compatible with existing systems that support POAP or SBT, but it still demands verification actions to be performed by the dApp. A more in-depth examination of these methods, along with examples, can be found in the subsequent Section VI.

VI. Applications

The zkAA scheme, a zero-knowledge proof-based identity management solution, has numerous potential applications in the field of decentralized applications (dApps) on blockchain technology. By eliminating the correlation between the identifier and the private key, the zkAA scheme offers a more secure and private approach for identity management in dApps. This scheme exhibits several feasible use cases, including mitigating Sybil attacks during NFT minting, fulfilling Know-Your-Customer (KYC) requirements, and securing wallet recovery. One particularly promising application of zkAA is its use in social login for dApps, such as online games. Currently, social login in web2 games is typically facilitated through a federated identifier. With the implementation of the zkAA scheme, social login can be extended to the blockchain, eliminating the need for mapping to or use of a web3 identifier. Moreover, as the secret is a certificate rather than a private key, dApps can be built with a hybrid architecture between web2 and web3, obviating the need for mapping and the storage of multiple secrets. This hybrid approach provides a cost-effective and high-performance solution, especially suitable for online games. The zkAA scheme also features multi-chain functionality, which is vital for games that use popular NFTs across multiple chains.

In this section, we will delve into the different ways in which the zkAA scheme can be employed in dApps, with a focus on its use in social login for blockchain-based games as a concrete example. This case study provides insight into the potential applications and benefits of the zkAA scheme for decentralized applications and highlights its potential for enhancing security and performance in identity management within the blockchain sphere.

A. Using the Hash Value as an Identifier

In the scenario depicted in Figure 3, a user engages in a social login process to gain access to and play games that are partially or fully facilitated by smart contracts. The identifier for the user is represented by the hash of their web2 identity, which is denoted as $id = \mathcal{H}(cert)$.

The user navigates to the website of a game and clicks on the login button, which redirects them to a social login provider (e.g., Google or GitHub) for authentication (Step A-1). Upon successful authentication, the user is redirected back to the game's website, with a $cert$ that contains information

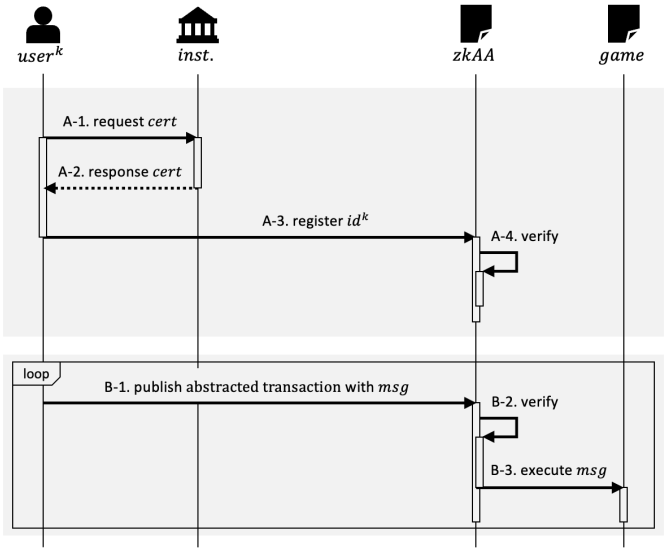

Fig. 3: Using $\mathcal{H}(cert)$ as identity.

regarding their identity (Step A-2). The user's browser then computes the hash value $id = \mathcal{H}(cert)$ and creates a zero-knowledge proof to demonstrate that the $cert$ was signed by the social login provider and that id is equal to $\mathcal{H}(cert)$. This proof, along with the hash value id, is then published to the blockchain (Step A-3). The contract verifies the proof, and if it is valid, submits the hash value $\mathcal{H}(cert)$ as the user's identifier (Step A-4). The explicit identifier, id_{chain}, is equivalent to the unique identifier, id. The user is now able to publish abstracted transactions on the blockchain (Step B-1). Then, the validity of the abstracted transaction is verified by the contract (Step B-2). Finally, the original request denoted as msg is executed (Step B-3).

This hash value identifier can be utilized by the user to interact with other decentralized applications, as long as these dApps are hash-value-identifier-aware rather than using a traditional address-based identifier.

B. Using Contract Wallet

Figure 4 illustrates an alternative method of implementing address abstraction using zero-knowledge proofs, which involves mapping the user's $\mathcal{H}(cert)$ identity to a contract wallet on the blockchain. The explicit identifier id_{chain} in this scenario is the address or Contract Account (CA) of the contract wallet.

The steps involved in this process are similar to those depicted in Figure 3. The only difference is the presence of a contract wallet. In Step A-4 of Figure 4, the zkAA contract verifies the proof and, upon validation, creates a mapping from the user's id to the newly minted contract wallet. The contract wallet interacts with the game contract via abstracted transactions (Steps B-3 and B-4) that utilize the original request msg.

In this scenario, the user's request msg is forwarded to the contract wallet, which then calls the $target$ contract. As a

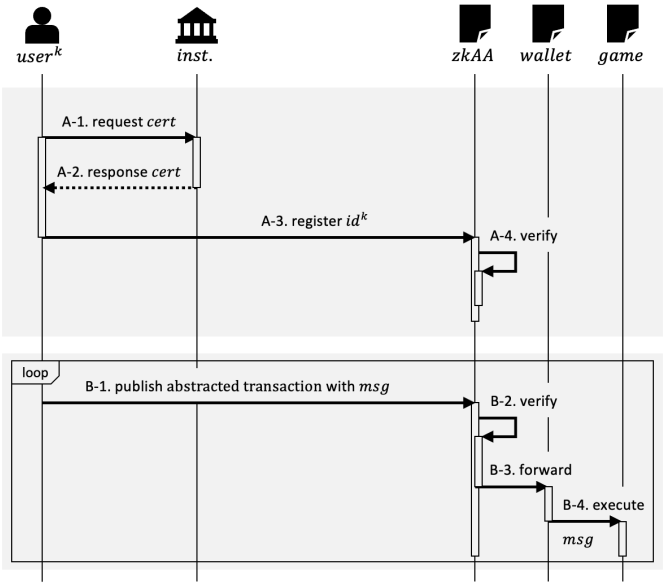

Fig. 4: Mapping $\mathcal{H}(cert)$ to contract wallet.

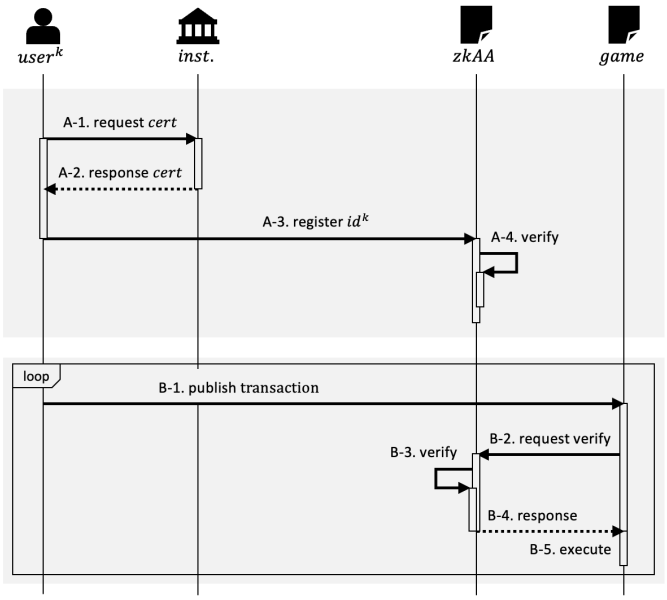

Fig. 5: Using $\mathcal{H}(cert)$ as NFT.

result, the caller's address in the request is always the address of the contract wallet. This implementation provides maximum compatibility with all decentralized applications that utilize the traditional address system as an identifier.

C. Using Non-Fungible Token

The utilization of Non-Fungible Tokens (NFTs) as a means of identification on a blockchain is presented in this scenario, as illustrated in Figure 5. NFTs are unique digital assets that represent ownership on a blockchain. These NFTs can be utilized as identifiers to allow users to engage with decentralized applications, such as POAP [19] or SBTs [43]. In this scenario, the zkAA contract creates a new NFT that is linked to the user's identity during the REGISTER phase (Step A-4).

One notable difference in this approach is that users do not have to publish abstracted transactions, as is the case in the previous two scenarios. Instead, users can publish normal transactions, as is typical in blockchain systems (Step B-1). The application contract must verify the NFT ownership by requesting validation from the zkAA contract (Step B-2). Upon successful verification of ownership (Steps B-3 and B-4), the request is executed normally (Step B-5). The NFT acts as an identifier for the user and enables them to interact with other decentralized applications. In this case, the explicit identifier id_{chain} is the address of the owner.

However, there are also challenges associated with using NFTs as identifiers. In this approach, the service contract must verify the user's identity through NFT ownership, increasing the complexity of dApp development and execution. Additionally, pre-existing dApps that do not consider NFT ownership checks may not be compatible with this method. In summary, the use of NFTs as identifiers presents a trade-off between the lack of abstraction and cost-efficiency, as users can send normal transactions rather than more costly abstracted transactions that require zero-knowledge proofs.

D. Multi-chain Decentralized Applications

The traditional approach of managing user identities in decentralized applications across multiple blockchain networks has posed a significant challenge, as the address systems used by different blockchain networks, such as Ethereum [46], Cosmos [23], and Polkadot [45], among others, are incompatible and result in difficulties in identifying the same user across multiple dApps.

The zero-knowledge address abstraction method offers a promising solution to this challenge. It employs a non-revealing $cert$ and its hash value $\mathcal{H}(cert)$ to establish a uniform, blockchain-agnostic identifier, thereby overcoming the difficulties posed by disparate address systems, as depicted in Figure 6. This not only enables the easy creation of multi-chain applications but also facilitates interoperability between different blockchain networks, allowing for the development of decentralized cross-chain applications while maintaining a consistent and secure user identity.

For instance, when transferring assets between different blockchain networks, the conventional approach relies on a bridge, such as [28], [47]. This method requires the input of a receiver address, which may not be compatible with the current blockchain, leading to potential errors in address input and resulting in a higher risk of assets being lost. However, the zero-knowledge address abstraction approach offers a solution to this issue by employing a uniform identifier for each user. This eliminates the risk of mistyping addresses and reduces the need to store multiple private keys for various blockchains, thus improving overall security.

In conclusion, zkAA presents a promising solution for the challenge of cross-chain identity management. Its blockchain-

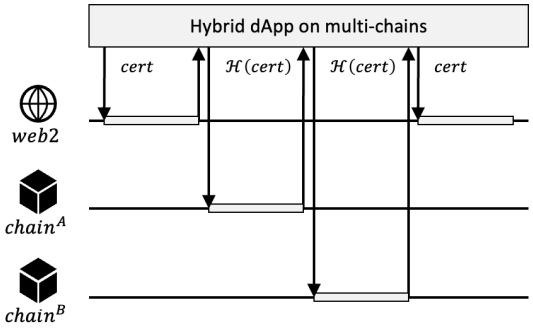

Fig. 6: Invariant identity across multi-chains and web2.

agnostic identity and compatibility with multi-chain applications make it a versatile and efficient solution for various use cases in the decentralized ecosystem.

E. Web2-Web3 Hybrid Applications

The integration of decentralized technologies with traditional web-based systems, referred to as web2-web3 hybrid applications, has become a prevalent solution for high-performance and gas-efficient applications, particularly in games. This integration presents an opportunity to offer an improved user experience by combining the ease of use of fast centralized systems with the security and decentralization of blockchain.

One of the major obstacles in the development of hybrid applications is the management of user identities, which are fragmented by each web2 and web3's unique identifier system. The zkAA approach offers a potential solution for this challenge by enabling a uniform representation of identifiers across both web2 and web3, as shown in Figure 6. The representation is based on the same $cert$ and eliminates the need for a service provider to manage multiple identities across various systems, simplifying the management.

In summary, the zkAA approach provides a promising solution for the management of user identities in web2-web3 hybrid applications. Its compatibility with both web2 and web3 systems makes it a versatile solution that can support the growth and development of hybrid applications in the decentralized ecosystem.

VII. Experiments

In order to gauge the efficiency and effectiveness of the proposed zero-knowledge address abstraction scheme, we conduct experiments that measure the gas cost for verification on the Ethereum [46] and Polygon [41], as well as the time required for creating witnesses and proofs locally.

A. Implementations

In this work, we have adopted EdDSA [2] for the institute to sign the certificate $cert$. To generate the signature, we apply the SHA512 digest to the $cert$ as input. The resulting signature is then used to compute the identifier id, from $cert$, through the application of the SHA256 hash function as $\mathcal{H}$, to the 1024-bit padded input derived from the signature. The registration phase involves the verification of EdDSA and SHA256 in a zero-knowledge manner, ensuring the user's privacy by not revealing the $cert$. In contrast, the publication phase employs a different zero-knowledge circuit that performs SHA256 verification on the preimage $cert$ while keeping it confidential.

The implementation of the proposed scheme is done using the ZoKrates [13] toolbox, which is based on the Groth16 [17] zero-knowledge proof system. We also utilize the ALT-BN128 curve for efficient on-chain verification through the use of pre-compiled contracts in Ethereum, which are created at addresses `0x6` for ADD, `0x7` for MUL, and `0x8` for pairing check [6], [36]. The implementation consists of two zero-knowledge circuits, with 169188 constraints in the registration circuit and 75945 constraints in the publication circuit. The experiments were conducted using a local machine, an Apple M1 Pro with 16GB of memory.

For the benefit of the research community, the implementation and experimental codes have been made publicly accessible on GitHub[1], allowing for replication and further study.

B. Verification Costs

Table II presents the examination of the costs associated with on-chain verification. The contract was optimized using the Solidity 0.8.17 version with the option (200 runs). On the Ethereum blockchain, the average gas price was 14 gwei ($14 * 10^-9$ ETH), with an ETH price of 1412.49 USD. On the Polygon network, the average gas price was 51.6 gwei ($51.6 * 10^-9$ MATIC), with a MATIC price of 0.91 USD. These data were derived as of January 13, 2023.

The deployment of the verification contracts is a one-time task that incurs a cost in gas. Specifically, the deployment of the registration verification contract requires 1410472 gas, translating to approximately $27.89 on Ethereum. The deployment of the publication verification contract requires 921902 gas, translating to approximately $18.23 on Ethereum. Although the initial cost of deploying the contracts may be substantial, it is a one-time expense. The cost of verifying the certificate identifier, id, is incurred at the time of registration and publication. On average, the cost of registering the identifier is 387494 gas, translating to approximately $7.66 on Ethereum. The cost of publishing the identifier is 239953 gas, translating to approximately $4.75 on Ethereum. On the Polygon network, the cost is significantly lower, at $0.02 and $0.01 for registration and publication, respectively. The cost of verifying the identifier during publication, while dependent on the user's requirements, can be considered economical, especially on low-cost blockchains like Polygon.

C. Overhead on Generating Proofs

Table III presents the average time taken to generate the witnesses and proofs for creating input values of the verification logic. During the registration process, it was observed

[1]https://github.com/lukepark327/zkAA

TABLE II: Gas cost on Ethereum and Polygon

Methods	min	max	avg	USD (avg)	
				Ethereum	Polygon
Registration					
verifyTx	387456	387516	387494	$ 7.6626	$ 0.0182
Deployment	-	-	1410472	$ 27.8919	$ 0.0662
Publication					
verifyTx	239906	239966	239953	$ 4.7450	$ 0.0113
Deployment	-	-	921902	$ 18.2305	$ 0.0433

TABLE III: Time taken for generating proofs (sec)

Methods	min	max	avg (sd)	med
Registration				
witness	3.7511	3.9998	3.8381 (0.0474)	3.8300
proof	5.4607	11.8173	5.7365 (0.6323)	5.6348
Publication				
witness	1.7636	1.9181	0.8132 (0.0285)	1.8098
proof	2.5685	2.9915	2.6759 (0.0640)	2.6668

that on average, 9.5746 seconds were spent creating the proofs. Conversely, the average time taken to generate the proofs for the publication of abstracted transactions was 3.4891 seconds. Although this time overhead is non-negligible, it is still within the acceptable limit given that the block interval of Ethereum is approximately 12 seconds, allowing ample time for calculation and broadcasting.

VIII. Discussion

A. Trusted Setup

Many tools for implementing zk-SNARKs, such as [13], [29], provide various implementations, including those usually using Quadratic Arithmetic Programs (QAPs) [15], [17], [18], [31]. Our proposed zero-knowledge address abstraction also utilizes the ZoKrates library [13] for its verification smart contract. However, the common use of QAP-based zk-SNARKs requires a trusted setup. While the trusted setup process can be mitigated through Multi-Party Computation, it requires the active participation of all parties, presenting a limitation.

To overcome this challenge, recent advancements in zero-knowledge proofs have aimed to introduce zk-SNARKs without the need for a trusted setup, making the system more decentralized and trustworthy [1], [3], [4], [38]. However, these systems lack the ability to provide constant verification, and as a result, suffer from substantial verification gas costs on-chain, or even experience limitations in verification due to block gas constraints. As such, a trade-off exists between the absence of a trusted setup and cost-efficiency.

B. Commit-and-Prove

In this work, we have leveraged the use of zk-SNARKs to construct zkAA. However, the field of zero-knowledge proof systems is rapidly evolving and there exists room for improvement in terms of efficiency. One promising direction for future work is the integration of commit-and-prove (CP) schemes [8], [14] into zkAA. CP is a type of SNARK that allows for knowledge to be verified through the use of pre-uploaded commitments. In CP-based SNARKs, the process of verifying hashes, which is typically the most computationally expensive part of generating a proof, is simplified through the use of Pedersen commitments [32]. This leads to a decomposition of computations into smaller, specialized proofs, thus potentially increasing the efficiency of proof generation. While the use of CP is expected to significantly increase the efficiency of proof generation, it may result in a slight increase in the cost of verification. This is due to the added complexity of verifying both the Pederson commitment and the original proof. However, it is important to note that the complexity of zk-SNARK verification remains constant.

In conclusion, our future work includes the exploration of CP-based SNARKs and a comprehensive investigation into their potential benefits and trade-offs. We believe that the integration of CP into zk-SNARKs holds the potential to greatly improve the efficiency and widespread adoption of zkAA.

C. Privacy and Transaction Fees

In this work, we have considered privacy as a key aspect of the zkAA system. The zkAA approach of detaching identity from the original blockchain address system offers the potential for improved privacy by allowing users to generate fresh ephemeral addresses for each transaction. This approach allows users to maintain their identity while keeping their history of transactions private. However, the creation of a fresh temporary address also results in the absence of native assets, making it difficult for users to pay transaction fees. The public link created when sending assets to a temporary address may also compromise the privacy of the transaction.

To mitigate these issues, various solutions have been explored, including the study of Stealth Addresses [5]. One potential solution is the use of account abstraction's paymaster [7], which is applicable only in the case of ERC20s. Another option to enhance privacy fundamentally is by utilizing zero-knowledge proofs to hide fee-sending transactions. However, this approach increases implementation complexity and overhead. Based on that, we plan to further investigate solutions for preserving transaction privacy in the zkAA system in future work.

D. Oracle Problem in Institute's Public Key

The verification of the validity of a certificate issued by an institute requires the saving of the institute's public key in a smart contract. This, however, poses a challenge in terms of the trust as it involves an oracle problem in the blockchain [10]. To overcome this issue, the public key can be stored through a governance process involving stakeholders. The other promising solution to address this challenge is to leverage the latest research in oracle problems, such as the Town Crier protocol [50].

IX. Related Work

A. Linking Web2 and Web3 Identities

Several approaches have been proposed to link web2 and web3 identities, with the aim of facilitating the transition from web2 to web3. The WebttCom framework [49] relies on a mapping between web2 identities and blockchain-specific identities to achieve this goal. Similarly, Holonym [22] utilizes a mapping approach, using zk-SNARKs to prove knowledge of the JSON Web Tokens (JWTs) preimage and signatures on-chain. Other works such as [35], [42], [48] also focus on linking web2 and web3 identities, but they still rely on blockchain-specific addresses. However, this mapping approach requires careful management of both web2 and web3 secrets of identities to ensure their secure use. This not only imposes a responsibility on users to manage their secrets securely, but also places a burden on service providers to securely keep and utilize both types of identities.

In contrast, our proposed system, zkAA, eliminates the need for an explicit mapping between web2 and web3 identities. The web2 identity, which is represented as a $cert$, can be directly used as a secret to generate a unique proof, without the need for considering blockchain-specific addresses. This innovative approach enables the development of multi-chain or web2-web3 hybrid applications, a capability that is not currently available in existing frameworks.

B. Zero-Knowledge Identity Systems

Several studies have leveraged zero-knowledge proofs (ZKPs) to preserve privacy in identity and credential systems. For instance, [9], [35], [37], [48] have utilized ZKPs to maintain the confidentiality of identities or credentials. [25] leverages ZKPs to conceal users' activities and service history from even malicious identity providers. [42] enables face matching for mapping actual individuals to accounts, where ZKPs are used for this mapping confidentially. Holonym [22] and Notebook [27] also utilize ZKPs to bridge the gap between web2 and web3 identities. Holonym verifies the signatures and hashes of JWTs using zk-SNARKs. Notebook, a zero-knowledge identity infrastructure, verifies credentials signed by third-party organizations as proof of humanity, storing such credentials on blockchain addresses.

However, both Holonym and Notebook rely on the original blockchain address as the identifier, rather than using itself as a web2 identity. In contrast, our proposed system, zkAA, utilizes the web2 identity as the direct identifier through $\mathcal{H}$, independent of the original blockchain address. Additionally, zkAA enhances privacy by eliminating the risk of data leakages, as the secret $cert$ is not publicly linked to any address.

C. Web3 Login Solutions

In the decentralized application (DApp) arena, Web3 Login Solutions, such as Web3Auth [24] and Ramper [34], have been proposed to enhance the user experience and simplify the onboarding process for both mainstream and crypto-savvy users. These solutions aim to provide seamless authentication through traditional web2 social login mechanisms. Web3Auth employs Shamir Secret Sharing (SSS) [39] and Threshold Cryptography [11] to divide the private key into multiple key shares. On the other hand, Ramper employs a third-party Key Management System (KMS), a hardware security module (HSM), and encrypted cloud storage. Transactions can be signed within a Trusted Execution Environment (TEE), ensuring that Ramper has no access to the private key and is unable to view it at any time.

However, these approaches still rely on the blockchain-specific identity system and protect their secrets through SSS or KMS/HSM. In contrast, zkAA fundamentally eliminates the need for storing private keys and instead utilizes the original $cert$ for social login. As a result, zkAA is inherently non-custodial and does not require the use of specialized hardware such as TEE, or trust in other trusted third parties.

X. Conclusion

In this paper, we introduce a novel approach for privacy-preserving identity management in blockchain systems, which leverages the concept of zero-knowledge proofs. Our proposed protocol, known as zkAA, enables users to securely associate their web2 identity with their presence on the web3 platform, without revealing any sensitive information.

The zkAA protocol offers several advantages over previous studies in the field. Firstly, users only need to keep their web2 identity and are free to keep their private keys in any manner they choose. Secondly, the same identifier can serve across different blockchains, eliminating the need to manage multiple addresses. Thirdly, the seamless integration of web2 and web3 services is possible due to the use of the same value for both the web2 identity and the web3 identifier. Lastly, privacy is protected as the user's certificate is not disclosed through the blockchain.

In conclusion, our contribution, the zkAA protocol, provides a secure, efficient, and privacy-preserving solution for identity management in blockchain systems. The seamless integration between web2 and web3 identities offers significant improvements to the user experience in the blockchain.

Acknowledgment

This work was supported by Coinplug corporations.

References

[1] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. *Cryptology ePrint Archive*, 2018.

[2] Daniel J Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. High-speed high-security signatures. *Journal of cryptographic engineering*, 2(2):77–89, 2012.

[3] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 315–334. IEEE, 2018.

[4] Benedikt Bünz, Ben Fisch, and Alan Szepieniec. Transparent snarks from dark compilers. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 677–706. Springer, 2020.

[5] Vitalik Buterin. An incomplete guide to stealth addresses . https://vitalik.ca/general/2023/01/20/stealth.html.

[6] Vitalik Buterin and Christian Reitwiessner. EIP-197: Precompiled contracts for optimal ate pairing check on the elliptic curve alt_bn128. https://eips.ethereum.org/EIPS/eip-197, 2017.

[7] Vitalik Buterin, Yoav Weiss, Kristof Gazso, Namra Patel, Dror Tirosh, Shahaf Nacson, and Tjaden Hess. EIP-4337: Account Abstraction Using Alt Mempool [DRAFT]. https://eips.ethereum.org/EIPS/eip-4337, 2021.

[8] Matteo Campanelli, Dario Fiore, and Anaïs Querol. Legosnark: Modular design and composition of succinct zero-knowledge proofs. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 2075–2092, 2019.

[9] Jing Chen, Zeyi Zhan, Kun He, Ruiying Du, Donghui Wang, and Fei Liu. Xauth: Efficient privacy-preserving cross-domain authentication. *IEEE Transactions on Dependable and Secure Computing*, 19(5):3301–3311, 2021.

[10] Matija Damjan. The interface between blockchain and the real world. *Ragion pratica*, (2):379–406, 2018.

[11] Yvo G Desmedt. Threshold cryptography. *European Transactions on Telecommunications*, 5(4):449–458, 1994.

[12] John R Douceur. The sybil attack. In *Peer-to-Peer Systems: First InternationalWorkshop, IPTPS 2002 Cambridge, MA, USA, March 7–8, 2002 Revised Papers 1*, pages 251–260. Springer, 2002.

[13] Jacob Eberhardt and Stefan Tai. Zokrates-scalable privacy-preserving off-chain computations. In *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 1084–1091. IEEE, 2018.

[14] Dario Fiore, Cédric Fournet, Esha Ghosh, Markulf Kohlweiss, Olga Ohrimenko, and Bryan Parno. Hash first, argue later: Adaptive verifiable computations on outsourced data. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1304–1316, 2016.

[15] Ariel Gabizon, Zachary J Williamson, and Oana Ciobotaru. Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. *IACR Cryptol. ePrint Arch.*, 2019:953, 2019.

[16] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In Robert Sedgewick, editor, *STOC*, pages 291–304. ACM, 1985.

[17] Jens Groth. On the size of pairing-based non-interactive arguments. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 305–326. Springer, 2016.

[18] Jens Groth and Mary Maller. Snarky signatures: Minimal signatures of knowledge from simulation-extractable snarks. In *Advances in Cryptology - CRYPTO 2017*, volume 10402, pages 581–612. Springer, 2017.

[19] POAP Inc. Proof of Attendance Protocol. https://poap.xyz/.

[20] M. Jones, J. Bradley, and N. Sakimura. JSON Web Token (JWT). https://www.rfc-editor.org/rfc/rfc7519, May 2015.

[21] Simon Josefsson and Ilari Liusvaara. Edwards-curve digital signature algorithm (eddsa). Technical report, 2017.

[22] Nanak Nihal Khalsa, Caleb Tuttle, Lily Hansen-Gillis, Kushal Kahar, and Shady El Damaty. Holonym: A decentralized zero-knowledge smart identity bridge. 2022.

[23] Jae Kwon and Ethan Buchman. Cosmos whitepaper. *A Netw. Distrib. Ledgers*, page 27, 2019.

[24] Torus Labs Pte Ltd. Web3Auth. https://web3auth.io/.

[25] Duc Anh Luong and Jong Hwan Park. Privacy-preserving identity management system on blockchain using zk-snark. *IEEE Access*, 11:1840–1853, 2023.

[26] Jerry W Markham. Front-running-insider trading under the commodity exchange act. *Cath. UL Rev.*, 38:69, 1988.

[27] Nathaniel Masfen-Yan, Solal Afota, Dhruv Mangtani, and Sacha Arroues-Paykin. Notebook: A zero-knowledge identity infrastructure layer.

[28] Multichain. Multichain. https://multichain.org/.

[29] Jose L Muñoz-Tapia, Marta Belles, Miguel Isabel, Albert Rubio, and Jordi Baylina. Circom: A robust and scalable language for building complex zero-knowledge circuits. 2022.

[30] Satoshi Nakamoto and A Bitcoin. A peer-to-peer electronic cash system. *Bitcoin.–URL: https://bitcoin. org/bitcoin. pdf*, 4(2), 2008.

[31] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy*, pages 238–252. IEEE, 2013.

[32] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology—CRYPTO'91: Proceedings*, pages 129–140. Springer, 2001.

[33] Wouter Penard and Tim van Werkhoven. On the secure hash algorithm family. *Cryptography in context*, pages 1–18, 2008.

[34] Inc Ramper. ramper. https://www.ramper.xyz/.

[35] Deevashwer Rathee, Guru Vamsi Policharla, Tiancheng Xie, Ryan Cottone, and Dawn Song. Zebra: Anonymous credentials with practical on-chain verification and applications to kyc in defi. *Cryptology ePrint Archive*, 2022.

[36] Christian Reitwiessner. EIP-196: Precompiled contracts for addition and scalar multiplication on the elliptic curve alt_bn128. https://eips.ethereum.org/EIPS/eip-196, 2017.

[37] Michael Rosenberg, Jacob White, Christina Garman, and Ian Miers. zk-creds: Flexible anonymous credentials from zksnarks and existing identity infrastructure. *Cryptology ePrint Archive*, 2022.

[38] Srinath Setty. Spartan: Efficient and general-purpose zksnarks without trusted setup. In *Advances in Cryptology – CRYPTO 2020: 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17–21, 2020, Proceedings, Part III*, page 704–737, Berlin, Heidelberg, 2020. Springer-Verlag.

[39] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[40] Secure Hash Standard. National institute of standards and technology (nist), fips publication 180-2, aug 2002.

[41] Polygon technology. Polygon. https://polygon.technology/.

[42] Taotao Wang, Shengli Zhang, and Soung Chang Liew. Linking souls to humans with zkbid: Accountable anonymous blockchain accounts for web 3.0 decentralized identity. *arXiv preprint arXiv:2301.02102*, 2023.

[43] E Glen Weyl, Puja Ohlhaver, and Vitalik Buterin. Decentralized society: Finding web3's soul. *Available at SSRN 4105763*, 2022.

[44] Sam Wilson, Ansgar Dietrichs, Matt Garnett, and Micah Zoltu. EIP-3074: AUTH and AUTHCALL opcodes [DRAFT]. https://eips.ethereum.org/EIPS/eip-3074, 2020.

[45] Gavin Wood. Polkadot Whitepaper. https://polkadot.network/PolkaDotPaper.pdf.

[46] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, 2014.

[47] Wormhole. Potral. https://www.portalbridge.com/.

[48] Xiaohui Yang and Wenjie Li. A zero-knowledge-proof-based digital identity management scheme in blockchain. *Computers & Security*, 99:102050, 2020.

[49] Guangsheng Yu, Xu Wang, Qin Wang, Tingting Bi, Yifei Dong, Ren Ping Liu, Nektarios Georgalas, and Andrew Reeves. Towards web3 applications: Easing the access and transition. *arXiv preprint arXiv:2210.05903*, 2022.

[50] Fan Zhang, Ethan Cecchetti, Kyle Croman, Ari Juels, and Elaine Shi. Town crier: An authenticated data feed for smart contracts. In *Proceedings of the 2016 aCM sIGSAC conference on computer and communications security*, pages 270–282, 2016.