# zkSABER: Zero-knowledge Succinct Authentication using Biometric Embedding Representation

Tomoki Chinen*, Christopher Wiraatmaja*, Yu Nakahata*, Takanori Hara*, Shoji Kasahara*
*Nara Institute of Science and Technology, Ikoma, Nara, Japan
Emails: chinen.tomoki.ct3@naist.ac.jp, wiraatmaja.christopher@naist.ac.jp,
yu.nakahata@is.naist.jp, hara@ieee.org, kasahara@ieee.org

***Abstract*—Executing biometric matching between two embedding vectors on the blockchain remains a challenging problem due to inherent privacy concerns and the computational constraints imposed by block gas limits. To address these challenges, we propose zk-SABER, a succinct blockchain-based biometric authentication scheme that allows constant proof size and verification cost with respect to the embedding vector length. Our design combines a Merkle Tree and a biometric matching algorithm within a zkSNARK circuit to prove that a user's biometric trait matches one of the registered templates in an anonymous manner. To ensure compatibility with state-of-the-art Deep Neural Network (DNN) models, we introduce a complete quantization pipeline that converts floating-point embeddings into zkSNARK-friendly representations. Our experiment results show constant transaction gas cost and proof size, regardless of the embedding vector length, thereby demonstrating the practicality of zk-SABER for real-world blockchain environments.**

***Index Terms*—anonymous, biometric authentication, zero-knowledge proof, blockchain, biometric embedding, quantization**

## I. Introduction

Authentication verifies a user's identity and serves as a fundamental component of access control, which prevents unauthorized access [1]. Traditional access control systems rely on centralized administrators, introducing a single point of failure. To address this, Blockchain-Based Access Control (BBAC) has been proposed, leveraging the decentralized and immutable nature of blockchain to mitigate server-side threats such as Man-in-the-Middle (MiTM) and Distributed Denial of Service (DDoS) attacks [2]. Authentication methods are generally categorized as knowledge-based, token-based, or biometric-based. While the first two have been well-studied in blockchain contexts [3]–[6], biometric-based authentication remains challenging due to its privacy risks. Unlike passwords or tokens, biometric data is tied to individuals and cannot be replaced, making privacy-preserving verification critical.

Several studies have applied zkSNARKs [7], [8] for privacy-preserving biometric authentication. Guo et al. [9] verified fingerprint features stored on trusted servers using zkSNARKs, while Kothari et al. [10] encrypted biometric data and stored it on IPFS. More recent methods use DNNs to extract embedding vectors, followed by privacy-preserving processing. Blanton et al. [11] used XOR-based secret sharing with MPC, assuming non-colluding servers. In contrast, BioZero by Lai et al. [12] employs Pedersen Commitments and zkSNARKs to privately compute distances using homomorphic properties. BioZero eliminates the need for trust in multiple parties and instead builds on well-established cryptographic assumptions, notably the perfect hiding property of Pedersen Commitments. However, BioZero's linear proof size and verification complexity can be prohibitive under blockchain gas limits, especially with longer embedding vectors that offer higher matching accuracy.

To address this challenge, we propose a privacy-preserving biometric authentication scheme with constant verification and proof size relative to the embedding length. We refer to this efficiency as succinctness, a term commonly used in the zkSNARK literature to describe protocols with sub-linear proof size and verification complexity. This naturally opens up possibilities to utilize more complex deep learning models that use higher embedding dimensions. Our method introduces a quantization pipeline that transforms floating-point DNN embeddings into zkSNARK-compatible inputs. These quantized embeddings, along with Merkle proofs, are used as private witnesses in a zkSNARK circuit, proving the sampled biometric trait matches one of the stored registered templates using set-membership proof [13]–[15]. The circuit then computes the Cosine Distance between the sampled and registered embeddings, ensuring zero knowledge throughout the authentication process. We summarize our contribution as follows:

- We propose zkSABER, a succinct blockchain-based biometric authentication scheme that allows constant transaction gas cost and proof size for any embedding vector length.
- Our proposed authentication scheme is fully zero-knowledge with the help of zkSNARK and a cryptographic hash function, ensuring the protection of the user's biometric embedding information and their access history.
- We provide a complete quantization pipeline in our method, introducing versatility to our proposal when implementing the state-of-the-art DNN model.

We organized the remainder of this paper as follows. In Section II, we introduce some important preliminaries that will be used throughout the paper. We then describe our proposal in Section III. We assert the security and the privacy claim in Section IV. We then explain the details of our implementation in Section V and present our experiment results in Sections VI. Finally, we conclude this paper in Section VII.

## II. PRELIMINARIES

### A. Zero-Knowledge Proofs

A zero-knowledge proof is a cryptographic protocol that enables a Prover to convince a Verifier that a statement is true without revealing any additional information beyond the statement's validity. Introduced by Goldwasser et al. in 1989 [16], this concept has become foundational in privacy-preserving technologies, particularly because all problems in NP are known to admit zero-knowledge proofs [17]. Originally defined as interactive protocols involving multiple rounds of communication, zero-knowledge proofs have since evolved into non-interactive forms, which are more suitable for asynchronous environments like blockchains. These Non-Interactive Zero-Knowledge Proofs (NIZKs) [18], often constructed using the Fiat-Shamir heuristic [19], allow a prover to generate a single static proof $\pi$ that convinces a verifier without further interaction.

In general, a NIZK $\Pi$ is based on a function $R$ known as an NP-relation. The relation $R$ takes a pair $(x, w)$ as input, where $x \in \mathcal{L}$ is the public statement, and $w$ is the secret information known only to the prover, or witness. It returns $R(x, w) = 1$ (true) only if the relation holds. Based on the definitions by Bennaroch et al. [20] and Groth et al. [7], a NIZK is formally defined as a tuple $\Pi$ consisting of the following three algorithms:

- **KeyGen(R) $\rightarrow$ ($\mathbf{CRS_{prove}}$, $\mathbf{CRS_{verify}}$):** Takes a relation $R$ as input and generates a Common Reference String (CRS) consisting of a proving key $\text{CRS}_{\text{prove}}$ and a verification key $\text{CRS}_{\text{verify}}$.
- **Prove($\mathbf{CRS_{prove}}$, x, w) $\rightarrow \pi$:** Takes the proving key, a statement $x$, and a witness $w$ satisfying $R(x, w) = 1$ as input, and generates a proof $\pi$.
- **VerProof($\mathbf{CRS_{verify}}$, x, $\pi$) $\rightarrow$ {0, 1}:** Takes the verification key, a statement $x$, and a proof $\pi$ as input, and either accepts (1) or rejects (0) the proof.

This NIZK must satisfy three properties: Completeness, Soundness, and Zero-Knowledge.

Furthermore, if a NIZK possesses an additional property called **Succinctness**, meaning the proof size and the verification complexity are at most logarithmic to the witness length, which we specifically referred to as a **SNARK** (Succinct Non-Interactive Argument of Knowledge).

**Definition 1** (SNARK). *A NIZK $\Pi$ for language $\mathcal{L}$ is called a succinct non-interactive argument of knowledge (SNARK) if $\Pi$ has an additional property called* succintness*, such that,*

- **Succinctness** *Given a statement $x \in \mathcal{L}$, and a witness $w$, the running time of verifier in $\Pi$ is* $\text{poly}(\lambda + |x| + \log |w|)$ *and the argument size is* $\text{poly}(\lambda + \log |w|)$.

Various zkSNARK schemes have been proposed, such as Groth16 [7]. We will also use a zk-friendly cryptographic hash function called Poseidon [21], which offers significant efficiency improvement compared to common hash functions like SHA256 or Keccak.

### B. Zero-Knowledge Set Membership Proof

A zero-knowledge set membership proof is a cryptographic protocol that allows a prover to demonstrate that an element belongs to a public set without revealing which specific element it is. The method utilized in this work is based on a combination of a Merkle Tree [15] and a zkSNARK.

First, a Merkle tree is constructed with the public keys $pk_1, pk_2, \ldots, pk_n$ of all registered members of the system as its leaves. Only the root of this tree, the Merkle root *root*, is published and maintained by a trusted third party. As the individual public keys $pk_i$ are not published, privacy is preserved.

When a member wishes to prove their membership, they know their secret key $sk_i$, the corresponding public key $pk_i$, and the Merkle path $path_i$, which is the list of sibling node hash values required to reach the root of the tree. The user provides these as a secret input (witness) to generate a zkSNARK proof $\pi$. This proof $\pi$ convinces a verifier of the following statement:

- The prover knows a certain $pk_i$ and $path_i$ that can be used to correctly compute the public *root*.

The verifier is thus convinced that the person is a legitimate member of the set, without learning their specific identity $pk_i$.

However, this basic scheme is vulnerable to replay attacks, where a valid proof $\pi$, once generated, can be intercepted by a malicious third party and reused illicitly. The standard solution to this vulnerability in the cryptographic community is the introduction of a Nullifier [22], [23].

A nullifier is a unique, single-use identifier associated with each proof, typically computed from a user's secret information and information that is unique to each proof instance. During proof verification, the verifier checks this nullifier against a list of all previously submitted nullifiers to detect and prevent double-spending. This mechanism ensures that each proof is valid only once, thereby providing resistance to replay attacks. Our proposed method also adopts this established approach to ensure the security of the authentication system.

### C. Symmetric Quantization

Since the proof process in zkSNARKs uses operations over a finite field $\mathbb{F}_p$, real-valued embedding vectors must be converted to integers. Therefore, this work employs symmetric quantization [24]. It maps a real number $x \in \mathbb{R}$ within a representable range $[-x_{\max}, x_{\max}]$ to an integer $\hat{x} \in \mathbb{F}_p$ using a scaling factor $s$, which determines the fixed-point precision, as follows:

$$\hat{x} = \begin{cases} \text{round}(x \cdot s), & \text{if } 0 \leq x \leq x_{\max}, \\ P - \text{round}(|x| \cdot s), & \text{if } -x_{\max} \leq x < 0. \end{cases}$$

Here, $P$ is the modulus of the finite field. This quantization allows for arithmetic operations within the circuit. Addition is modular addition of the quantized integers. The result of a multiplication, however, has a scale of $s^2$, requiring a rescale to the original scale $s$. Division $c = a/b$ is efficiently performed

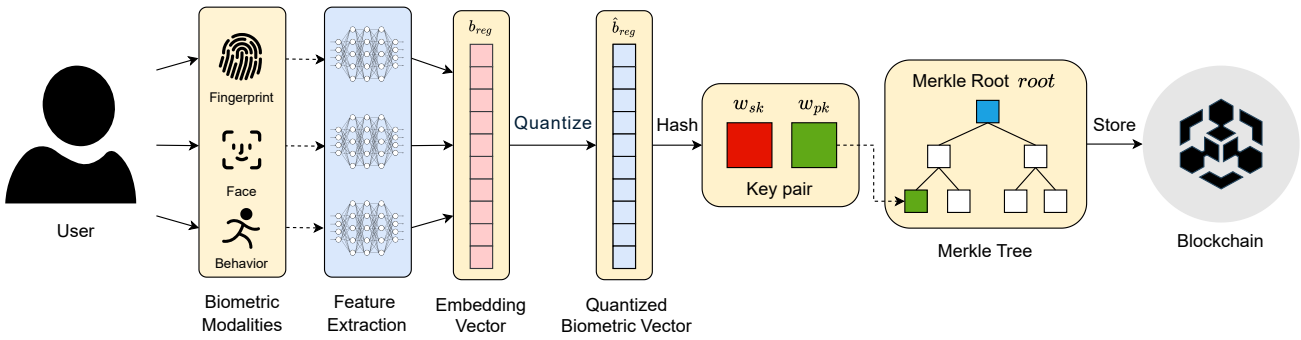


Fig. 1. zkSABER: Registration Protocol.

in-circuit by having the prover provide a quotient $q$ and a remainder $r$ (where $0 \leq r < |\hat{b}|$), while the verifier checks the identity $\hat{a} = q \cdot \hat{b} + r$.

## III. Proposed Method

In this section, we describe zkSABER, a succinct blockchain-based biometric authentication scheme, which allows constant transaction gas cost and proof size. This is made possible by our quantization pipeline, which efficiently converts a floating-point value from a DNN model into a finite field representation suitable for zkSNARKs.

In the registration protocol, an off-chain scanning device first captures the user's biometric data, which is then processed by a DNN to generate an embedding vector. This vector is then converted into an integer vector $\hat{\mathbf{b}}_{\text{reg}} \in \mathbb{F}_p^d$ suitable for zkSNARKs using the method described in Section II-C. From this vector, the user deterministically derives a secret key $\omega_{sk}$ and a corresponding public key $\omega_{pk}$. An administrator then aggregates the $\omega_{pk}$ of all users into a Merkle tree and records only its root on-chain.

During the authentication protocol, the user generates a proof $\pi$ off-chain, which simultaneously proves four claims in zero-knowledge: (1) membership in the Merkle tree (2) that the cosine similarity between the registered vector $\hat{\mathbf{b}}_{\text{reg}}$ and the authentication-time vector $\hat{\mathbf{b}}_{\text{auth}}$ exceeds a predefined threshold $\tau$ (3) the cryptographic binding between the biometric data and the identity and (4) resistance to replay attacks via a nullifier.

Since the smart contract only needs to verify the small, fixed-size proof $\pi$ without processing any variable-length data, an $O(1)$ verification cost is achieved. This enables the practical and large-scale deployment of privacy-preserving biometric authentication systems.

### A. System Model and Definitions

In this section, we define the components of the proposed authentication protocol, zkSABER, and the underlying security assumptions.

The hash function used in this paper is Poseidon, defined as $\mathrm{H} : \mathbb{F}_p \times \mathbb{F}_p \to \mathbb{F}_p$. Furthermore, for a vector $\hat{\mathbf{b}} \in \mathbb{F}_p^d$ of length $d > 2$, we recursively apply the hash function H in a manner similar to building a Merkle tree to compute a single root element. We denote this operation as $\mathrm{H}(\hat{\mathbf{b}})$.

*1) System Model and Security Assumptions:* This authentication system consists of the following three entities:

- **User (Prover):** Registers biometric information and generates a zero-knowledge proof for authentication.
- **Verifier (Smart Contract):** Verifies the submitted proof on-chain.
- **Administrator:** Handles the initial system setup and updates the Merkle tree upon user registration.

The trust model assumes that the Verifier and Administrator are honest entities that follow the protocol, while Users are potentially malicious. Based on this model, zkSABER aims to protect against the following threats. A detailed technical analysis for each threat is provided in the later Security Analysis section.

- **Spoofing:** Fraudulent authentication using other user information.
- **Replay Attack:** Unauthorized reuse of a previously valid authentication proof.
- **Privacy Violation:** Inferring sensitive data, such as biometric information, from public data on-chain.

### B. Registration Protocol

This section details the user registration protocol in zkSABER. As shown in Figure 1, the protocol consists of an off-chain process on the user's device and an on-chain registration performed by the Administrator. The registration process comprises the following three main steps.

*a) Secret Key Generation via Quantization:* The user extracts a feature vector $\mathbf{b}_{\text{reg}}$ using a DNN model and quantizes it into a finite field vector $\hat{\mathbf{b}}_{\text{reg}} \in \mathbb{F}_p^d$ suitable for zkSNARKs. Next, they deterministically generate a secret key $\omega_{sk} = \mathrm{H}(\hat{\mathbf{b}}_{\text{reg}})$ from this vector (Algorithm 1, lines 1-3).

*b) Identity Commitment Construction:* Using the generated secrets $\omega_{sk}$, the user constructs an identity commitment $\omega_{pk}$, where $\omega_{pk} = \mathrm{H}(\omega_{sk})$ serves as their public identifier (Algorithm 1, lines 4).

*c) On-chain Registration:* The Administrator receives $\omega_{pk}$, adds it as a new leaf to the Merkle tree, and then records the updated Merkle root to the smart contract on-chain(Algorithm 1, lines 5-8).

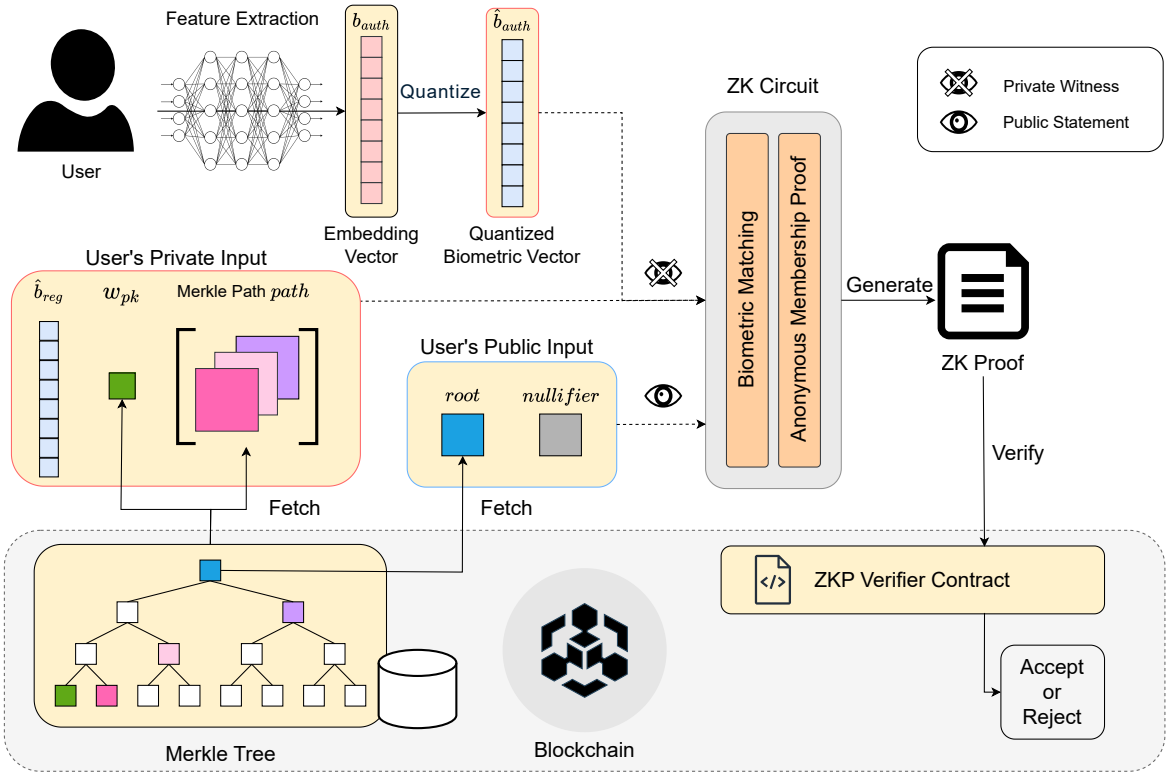


Fig. 2. zkSABER: Authentication Protocol.

**Algorithm 1** Registration Protocol.

**Input:** User's face image $I$.
**Output:** The blockchain state is updated with a new Merkle root.

*// Generate keys from biometric data*
$\mathbf{b}_{\text{reg}} \leftarrow \text{DNN\_Extractor}(I)$
$\hat{\mathbf{b}}_{\text{reg}} \leftarrow \text{Quantize}(\mathbf{b}_{\text{reg}})$ ▷ To finite field vector
$\omega_{sk} \leftarrow \text{H}(\hat{\mathbf{b}}_{\text{reg}})$ ▷ H is Poseidon hash
$\omega_{pk} \leftarrow \text{H}(\omega_{sk})$

*// Update the on-chain Merkle root with the new user*
Receive $\omega_{pk}$ from User
$current\_root \leftarrow \text{SmartContract.getMerkleRoot}()$
$root_{\text{new}} \leftarrow \text{GetRoot}(\text{InsertLeaf}(current\_root, \omega_{pk}))$
Call `updateMerkleRoot` on Smart Contract with $root_{\text{new}}$

### C. Authentication Protocol

This section details the authentication protocol in zkSABER, which allows a registered user to prove their identity. As illustrated in Figure 2, the authentication process consists of two main phases: an off-chain zero-knowledge proof generation executed on the user's device, and an on-chain verification conducted by a smart contract.

The core of our protocol lies in the construction of a unified zero-knowledge proof $\pi$ that simultaneously attests to multiple claims, thereby achieving security, privacy, and the efficiency of a constant time $O(1)$ verification cost. For this purpose, the user generates a proof that satisfies a single integrated authentication relation, denoted $R_{\text{zkSABER}}$. This relation is composed of the following two subrelations:

*1) Biometric Matching Relation:* This subrelation verifies that the cosine similarity between the registered biometric vectors and the ones captured during authentication exceeds a predefined threshold $\tau \in (0, 1)$. To address the high cost of direct square root and division operations in a zkSNARK circuit, our method achieves efficiency by approximating the squared $\ell_2$ norm $n$ with bounded error. Furthermore, division is replaced by an approach that relies on multiplication adapted from the Euclidean division identity (detailed in Section II-C). Accordingly, the threshold $\tau$ is also converted to a scaled integer $\tau' := \lfloor \tau \cdot s \rfloor$. This matching relation $R_{\text{match}}$, reflecting these efficiency improvements, is defined as follows:

$$
\begin{aligned}
&R_{\text{match}}(\tau', (\hat{\mathbf{b}}_{\text{reg}}, \hat{\mathbf{b}}_{\text{auth}}, n_{\text{reg}}, n_{\text{auth}}, \tilde{c}, \epsilon)) = 1 \\
&\iff |n^2 - \|\hat{\mathbf{b}}\|^2| < \theta \\
&\wedge s \cdot \langle \hat{\mathbf{b}}_{\text{reg}}, \hat{\mathbf{b}}_{\text{auth}} \rangle = \tilde{c} \cdot (n_{\text{reg}} \cdot n_{\text{auth}}) + \epsilon \\
&\wedge \tilde{c} > \tau'.
\end{aligned}
$$

Here, $\theta$ denotes the allowed error tolerance in the norm approximation, $\epsilon$ is the remainder term, and $s$ is a fixed scaling factor.

*2) Anonymous Membership Proof Relation:* This subrelation proves that the user is a valid member of a group based

on a Merkle tree and that the authentication is non-reusable. It checks the validity of a Merkle path and a nullifier linked to the user's secret key $\omega_{\text{sk}}$. The relation is defined as follows, where $R_{\text{pair}}$ verifies that the public-private key pair is consistent:

$$\begin{aligned} R_{\text{AnonMem}}(root, \text{nullifier}, (\omega_{\text{pk}}, \omega_{\text{sk}}, path, r)) = 1 \\ \iff \omega_{\text{pk}} \in root \land R_{\text{pair}}(\omega_{\text{pk}}, \omega_{\text{sk}}) = 1 \\ \land \text{nullifier} = \text{H}(\omega_{\text{sk}}, r). \end{aligned}$$

Finally, we unify the two subrelations, $R_{\text{match}}$ and $R_{\text{AnonMem}}$, into a single authentication relation, $R_{\text{zkSABER}}$. The core of this unification is a binding constraint that links the secret key $\omega_{\text{sk}}$ from the membership proof to the registered biometric vector $\hat{\mathbf{b}}_{\text{reg}}$. This constraint ensures that the secret key is deterministically derived from the biometric vector, thereby intrinsically connecting the user's on-chain anonymous membership to their physical biometric identity.

The complete relation $R_{\text{zkSABER}}$, a function of the public statement $x$ and private witness $\omega$, is thus defined by the following conjunction:

$$\begin{aligned} R_{\text{zkSABER}}(x, \omega) = 1 \\ \iff R_{\text{match}}(\tau', (\hat{\mathbf{b}}_{\text{reg}}, \hat{\mathbf{b}}_{\text{auth}}, n_{\text{reg}}, n_{\text{auth}}, \tilde{c}, \epsilon)) = 1 \\ \land R_{\text{AnonMem}}(root, \text{nullifier}, (\omega_{\text{pk}}, \omega_{\text{sk}}, path, r)) = 1 \\ \land H(\omega_{\text{sk}}) = H(\hat{\mathbf{b}}_{\text{reg}}), \end{aligned}$$

where the statement $x$ and witness $\omega$ consist of:

- $x \coloneqq (root, \text{nullifier}, \tau')$
- $\omega \coloneqq (\omega_{\text{pk}}, \omega_{\text{sk}}, path, r, \hat{\mathbf{b}}_{\text{reg}}, \hat{\mathbf{b}}_{\text{auth}}, n_{\text{reg}}, n_{\text{auth}}, \tilde{c}, \epsilon)$.

A key aspect of the proposed architecture's efficiency lies in the composition of its public statement. Since data of variable length such as biometric vectors and Merkle paths are treated entirely as private witnesses, the smart contract only verifies a small, fixed-size statement $x$. This design reduces the on-chain verification cost to constant time ($O(1)$), independent of the biometric vector length or the number of registered users. The concrete protocol flow for generating and verifying the zero-knowledge proof $\pi$ is presented in Algorithm 2.

**Algorithm 2** Authentication Protocol with Explanations.

**Input:** User's authentication face image $I_{\text{auth}}$; User's self-managed secret $\hat{\mathbf{b}}_{\text{reg}}$.
**Output:** Authentication result (Success / Fail).
*// — User Side (Off-chain) —*
*1. Generate proof components and chain state:*
$\hat{\mathbf{b}}_{\text{auth}} \leftarrow \text{Quantize}(\text{DNN\_Extractor}(I_{\text{auth}}))$
$(\omega_{sk}, \omega_{pk}) \leftarrow \text{GenerateKeys}(\hat{\mathbf{b}}_{\text{reg}})$
$\text{nullifier} \leftarrow \text{H}(\omega_{sk}, r)$
$(root, \text{path}) \leftarrow \text{GetMerkleProof}(\omega_{pk})$
$(n_{\text{reg}}, n_{\text{auth}}, \tilde{c}, \epsilon) \leftarrow \text{ComputeSimilarity}(\hat{\mathbf{b}}_{\text{reg}}, \hat{\mathbf{b}}_{\text{auth}})$

*2. Generate a zero-knowledge proof $\pi$:*
Define public statement $x$ and private witness $w$
$x \coloneqq (root, \text{nullifier}, \tau')$
$w \coloneqq (\omega_{pk}, \omega_{sk}, path, r, \hat{\mathbf{b}}_{\text{reg}}, \hat{\mathbf{b}}_{\text{auth}}, n_{\text{reg}}, n_{\text{auth}}, \tilde{c}, \epsilon)$
$\pi \leftarrow \text{ZK\_SNARK.Prove}(x, w)$
SendTransaction("authenticate", $\pi$, $x$)

*// — Verifier Contract (On-chain) —*
**function** authenticate($\pi$, $x$)
  *Check 1: Prevent replay attacks*
  **Require** `isSpent[x.nullifier] == false`

  *Check 2: Verify ZK proof*
  **Require** `Verifier.verify(`$\pi$, $x$`) == true`
  *The ZK proof internally verifies:*
  *(a) Relation $R_{\text{AnonMem}}$ holds: Proves valid membership & non-reusability.*
  *(b) Relation $R_{match}$ holds: Proves biometric similarity $> \tau'$.*
  *(c) Binding constraint holds: Links biometrics to the secret key.*

  `isSpent[x.nullifier] = true`
  **emit** AuthenticationSuccess
**end function**

## IV. Security Analysis

In this section, we will assert the claim about zkSABER security and privacy inside the blockchain. We assume the malicious adversary $\mathcal{A}$ is a probabilistic polynomial time algorithm (PPT), in which we treat their algorithm as a black box algorithm. We define two adversaries in our security proof, $\mathcal{A}_{\text{sec}}$ and $\mathcal{A}_{\text{priv}}$, each denoting the security adversary and privacy adversary. Our threat model disallowed the adversary from tampering with the algorithm inside the blockchain, assuming the immutability properties of the blockchain.

### A. Security Adversary Analysis

**Definition 2** (Authentication Forgery Game)**.** *The security of the authentication scheme $\Pi_{\text{auth}}$ is defined by the following game between a challenger and a PPT adversary $\mathcal{A}_{\text{sec}}$:*

1) ***Setup:*** *The challenger generates a set of user public/private key pairs, populates a Merkle tree with the public keys, and computes the root* root*. Let $pk_{\text{set}}$ be the set of all public keys and $\pi_{\text{set}}$ be the set of all valid proofs generated by honest users so far.*
2) ***Challenge:*** *The challenger provides the adversary $\mathcal{A}_{\text{sec}}$ with the public parameters: the Merkle root* root*, the set of public keys $pk_{\text{set}}$, and the set of existing proofs $\pi_{\text{set}}$.*
3) ***Forgery:*** *The adversary $\mathcal{A}_{\text{sec}}$ outputs a new proof and nullifier pair $(\pi', \text{nullifer}')$.*

*We say that the scheme is secure if for any PPT adversary $\mathcal{A}_{\text{sec}}$, the probability of winning the game is negligible. The adversary wins if $\Pi.\text{VerProof}(\text{root}, \text{nullifer}', \pi') = 1$ and the nullifier* nullifer$'$ *has not been used in any proof within $\pi_{\text{set}}$.*

$$\Pr\left[\begin{matrix}(\pi', \text{nullifer}') \leftarrow \mathcal{A}_{\text{sec}}(\text{root}, \pi_{\text{set}}, pk_{\text{set}}) \\ : \Pi.\text{VerProof}(\text{root}, \text{nullifer}', \pi') = 1\end{matrix}\right] \leq \text{negl}(\lambda).$$

**Theorem 1.** *Assuming the zkSNARK* $\Pi$ *is computationally knowledge sound and the hash function* $H$ *is a random oracle, our authentication scheme* $\Pi_{\text{auth}}$ *is secure under Definition 2.*

*Proof.* (Sketch) In the Authentication Forgery Game, any attempt by an adversary $\mathcal{A}_{\text{sec}}$ to generate a proof $(\pi', \text{nullifer}')$ that passes verification can be reduced to breaking either the computational knowledge soundness of the zkSNARK $\Pi$ or the random oracle assumption on the hash function $H$. To generate a valid proof $\pi'$, one must possess knowledge of the corresponding witness, which consists of the secret key and the Merkle path.

If an adversary could generate a valid proof $\pi'$ without knowledge of the witness, this would violate the computational knowledge soundness of the zkSNARK $\Pi$. Furthermore, any attempt to forge a Merkle path or to reverse-engineer a secret key from a public key would require finding a collision for the hash function $H$. Such an attack is computationally infeasible under the assumption that $H$ is a random oracle.

Additionally, the nullifer used in each authentication is unique and recorded by the system upon use. This prevents Replay Attacks at the protocol level, where an adversary eavesdrops on a legitimate user's transmission and resubmits it later.

Therefore, the success probability of $\mathcal{A}_{\text{sec}}$ is bounded by the probability of breaking either the knowledge soundness of $\Pi$ or the properties of $H$ as a random oracle. As these probabilities are negligible by definition, the adversary's success probability is bounded by a negligible function $\text{negl}(\lambda)$. □

### *B. Privacy Adversary*

**Definition 3** (User Guessing Game)**.** *The privacy of the scheme* $\Pi_{\text{auth}}$ *is defined by the following game between a challenger and a PPT adversary* $\mathcal{A}_{\text{priv}}$*:*

1) ***Setup:*** *The challenger sets up the system as in the security game, involving two honest users, user 0 and user 1, with public keys* $pk_0, pk_1 \in pk_{\text{set}}$.
2) ***Challenge:*** *The challenger flips a random bit* $b \in \{0, 1\}$. *It then generates a valid proof* $\pi_b$ *and nullifier* $\text{nullifer}_b$ *for user b. The challenger provides the adversary* $\mathcal{A}_{\text{priv}}$ *with the public parameters* $(\text{root}, \pi_{\text{set}}, pk_{\text{set}})$ *along with the challenge tuple* $(\pi_b, \text{nullifer}_b)$.
3) ***Guess:*** *The adversary* $\mathcal{A}_{\text{priv}}$ *outputs a guess* $b'$.

*We say that the scheme preserves privacy if for any PPT adversary* $\mathcal{A}_{\text{priv}}$*, their advantage in winning the game is negligible. The adversary wins if* $b' = b$.

$$\Pr\left[\begin{matrix} b' \leftarrow \mathcal{A}_{priv}(\text{root}, \text{nullifer}_b, \pi_b, \pi_{\text{set}}, pk_{\text{set}}) \\ : b' = b \end{matrix}\right] \leq \frac{1}{2} + \text{negl}(\lambda).$$

**Theorem 2.** *If the zkSNARK* $\Pi$ *is computationally zero-knowledge and the hash function* $H$ *is a random oracle, then our authentication scheme* $\Pi_{\text{auth}}$ *preserves privacy according to Definition 3.*

*Proof.* (Sketch) To win the User Guessing Game, the adversary $\mathcal{A}_{\text{priv}}$ must identify the user from the challenge information. Any such attempt will fail for the following reasons:

1) **The root**: It is shared among all users and contains no user distinguishing information.
2) **The nullifer**$_b$: It is generated from a secret and a fresh random value for each authentication. Due to the properties of the random oracle $H$, the nullifiers from different authentications by the same user have no computational correlation. This prevents an adversary from linking multiple actions to a specific user, ensuring unlinkability.
3) **The proof** $\pi_b$: The computational zero-knowledge property of the zkSNARK $\Pi$ ensures that the proof $\pi_b$ reveals nothing about the witness used to generate it. Thus, the proof is computationally indistinguishable from that of any other user.

Since all user specific information available to the adversary is computationally indistinguishable from random data, the adversary cannot guess the user's identity $b$ with a probability significantly better than random chance $(1/2)$. This advantage is bounded by the probability of breaking the zero-knowledge property of $\Pi$ or the random oracle property of $H$, which is a negligible function. □

## V. Implementation

We implement our work using the details shown in Table I. We choose Groth16 to implement our scheme, mainly due to its smallest proof size in zkSNARK, which is 2048 bits long, when implemented on BN254. We use Poseidon due to its low constraint, which improves the proving time significantly compared to using SHA256 in our Merkle Tree. We choose the embedding vector length from 2 to 1024, as currently there are many DNN model that provides embedding with higher than 512 embedding length. We choose the Merkle Tree of depth 10 (resp. 20), as it allows the system to possess 1024 users (resp. 1 million users) to register.

TABLE I
Implementation Details.

| **Item** | **Specification** |
|---|---|
| Hardware | Apple M1 CPU and 8GB RAM |
| Language | Python 3.10.14 |
| Toolkits | ZoKrates 0.8.8, Hardhat v2.25.0 |
| Key Libraries | py-solc-x v2.0.3, web3.py v7.11.1, numpy v1.26.4, poseidon-hash v0.1.4, deepface v0.0.93 |
| **Parameters** | |
| Elliptic Curve | BN254 [25] |
| Proof System | Groth16 |
| Hash Function | Poseidon |
| Embedding Vector Length | $\{2^1, 2^2, ..., 2^{10}\}$ |
| Merkle Tree Depth | $\{10, 20\}$ |
| Distance Metric | Cosine Distance |

## VI. Experiment

In this section, we will quantitatively measure our succinct blockchain-based biometric authentication scheme in some of its important metrics. As a baseline, we will also implement

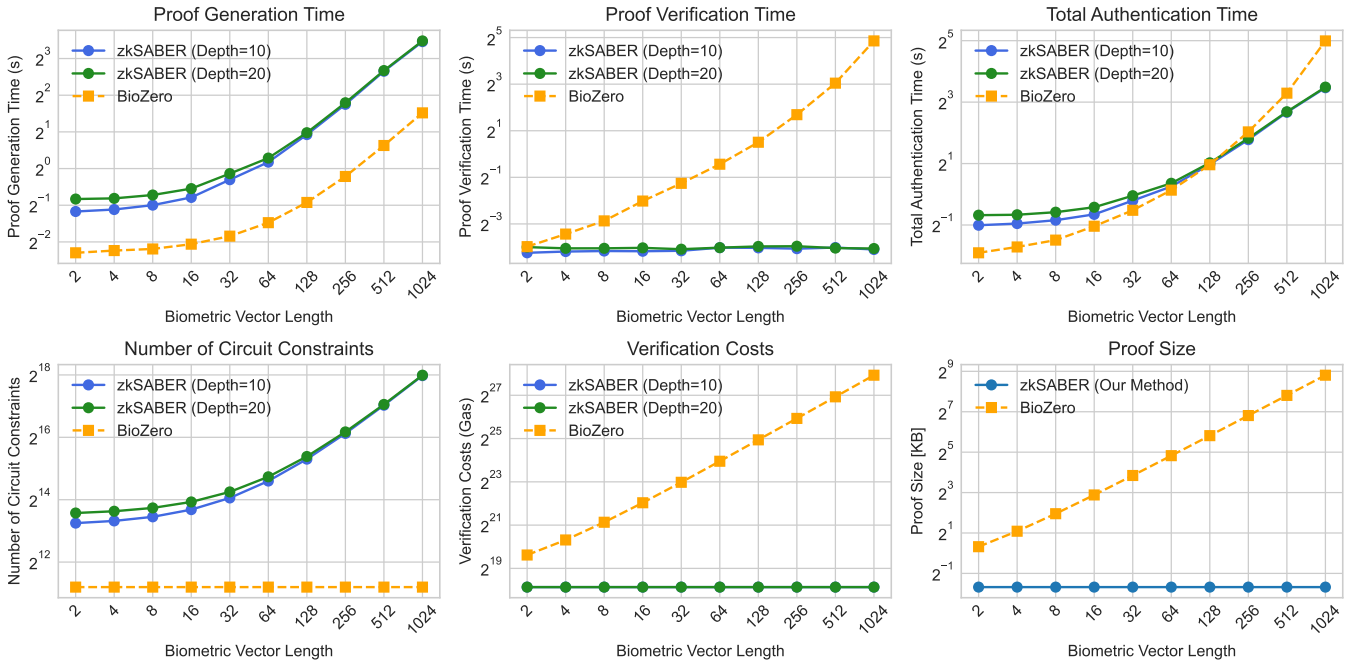


Fig. 3. Experimental evaluation results of zkSABER and BioZero.

BioZero [12] using a similar implementation, with some modifications included to allow comparison. For example, we implement their Pedersen Commitment using the multiplicative group of $\mathbb{F}_q$ where $q$ is the base prime in BN254. Thus, each element in this multiplicative group will be of size 256 bits.

We will focus our measurement on 6 different metrics, each capturing the scheme's performance in a detailed manner. The details of our evaluation metrics can be seen on Table II. We repeat the measurement of each scheme 100 times and record the value as written in the Figure. 3.

TABLE II
EVALUATION METRICS.

| Metric | Definition |
|---|---|
| *Proof Generation Time* | Time to generate a proof $\pi$. |
| *Proof Verification Time* | Time to verify a proof $\pi$. |
| *Total Authentication Time* | Time taken to generate and verify a proof in an off-chain environment. |
| *Proof Size* | Data size of the proof $\pi$ in bytes. |
| *Verification Costs* | Gas consumed for proof verification. |
| *Circuit Size* | Number of R1CS constraints. |

As shown in Figure 3, the Proof Size of zkSABER was observed to be constant, while its Verification Costs (Gas) and Verification Time remained nearly constant with respect to the increase in vector length. Specifically, the Proof Size was consistently 0.312 KB, and the Verification Costs were approximately 287,900 gas. In contrast, each metric for BioZero increased linearly with the vector length, reaching a Proof Size of 56.375 KB and Verification Costs of $3.22 \times 10^7$ gas at a dimension of 128.

This result clearly highlights the fundamental difference in the protocol design of the two methods. Our scheme is designed to achieve a constant verification cost by assigning all the embedding vectors inside the private witness of zkSNARK. On the other hand, BioZero prioritizes simplifying the zkSNARK circuit, in turn delegating the verification of vector-length-dependent Pedersen Commitments to the on-chain process, thereby causing its verification cost to increase linearly. As we observed, if the gas cost for a single transaction in Ethereum reaches 36M, it will be terminated due to the block gas limit. We found that BioZero reaches this limit when the embedding pass the 128 length, prohibiting longer embedding vector for their approach.

As shown in Figure 3, zkSABER's proof generation time, along with its number of circuit constraints, increases gradually with the vector length. This reflects an intentional design trade-off that accepts off-chain computational load in exchange for on-chain efficiency. However, in terms of the total authentication time, a performance improvement was observed where zkSABER becomes faster than BioZero when the vector length exceeds 128. Furthermore, increasing the Merkle tree depth from 10 to 20 had only a slight impact on off-chain performance and no impact on the verification costs, showing our scheme's scalability.

## VII. CONCLUSION

We proposed zkSABER, a succinct blockchain-based authentication scheme, enabling constant transaction gas cost and proof size by leveraging the Groth16 and a Merkle Tree with the Poseidon hash, supporting any length of embedding vector. To support compatibility with existing deep neural

network (DNN) models, we developed a quantization pipeline that converts floating-point vectors into finite field elements, producing inputs suitable for ZoKrates circuits. We provided our security analysis against two different adversaries, a security and privacy adversary, asserting our scheme's security and privacy guarantees in a public blockchain environment.

We implemented both zkSABER and BioZero, allowing a detailed quantitative comparison of their performance trade-offs. Our experimental results confirm that zkSABER achieves constant transaction gas cost, proof size, and verification time, regardless of the embedding vector length. None of our implementations exceeded the block gas limit, affirming the practicality and future-proof compatibility of our scheme.

Nevertheless, the proving time of zkSABER remains a primary performance bottleneck. This overhead, stemming from our implementation with Groth16, could potentially be reduced to logarithmic complexity while preserving succinct verification by adopting linear-time zkSNARKs such as Orion [26] or Brakedown [27]. Proving performance could be further optimized by replacing the Merkle tree with lookup arguments [28].

Further research directions include a rigorous security analysis of the quantization pipeline against side-channel and model inversion attacks, as well as a comprehensive performance evaluation on resource-constrained hardware, such as IoT devices, to validate its practical viability. Addressing these challenges is pivotal to the advancement of secure and scalable on-chain verification frameworks.

## ACKNOWLEDGEMENT

This research was supported in part by Japan Society for the Promotion of Science under Grant-in-Aid for Scientific Research (B) No. 24K02931 and NAIST J-PEAKS.

## REFERENCES

[1] A. Ouaddah, H. Mousannif, A. Abou Elkalam, and A. Ait Ouahman, "Access control in the internet of things: Big challenges and new opportunities," *Computer Networks*, vol. 112, pp. 237–262, 2017.

[2] J. C. L. Ho and C. Lin, "An anonymous on-street parking authentication scheme via zero-knowledge set membership proof," *CoRR*, vol. abs/2108.03629, 2021.

[3] D. Di Francesco Maesa, P. Mori, and L. Ricci, "Blockchain based access control," in *Proc. of the Distributed Applications and Interoperable Systems*, 2017, pp. 206–220.

[4] M. Yutaka, Y. Zhang, M. Sasabe, and S. Kasahara, "Using ethereum blockchain for distributed attribute-based access control in the internet of things," in *Proc. of the IEEE GLOBECOM*, 2019, pp. 1–6.

[5] C. Wiraatmaja, Y. Zhang, M. Sasabe, and S. Kasahara, "Cost-efficient blockchain-based access control for the internet of things," in *Proc. of the IEEE GLOBECOM*, 2021, pp. 1–6.

[6] C. Wiraatmaja and S. Kasahara, "Cost-efficient anonymous authentication scheme based on set-membership zero-knowledge proof," in *Proc. of the 2023 5th Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)*, 2023, pp. 1–8.

[7] J. Groth, "On the size of pairing-based non-interactive arguments," in *Proc. of the Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, ser. Lecture Notes in Computer Science, vol. 9666. Springer, 2016, pp. 305–326.

[8] A. Chiesa, Y. Hu, M. Maller, P. Mishra, N. Vesely, and N. Ward, "Marlin: Preprocessing zksnarks with universal and updatable srs," in *Proc. of the Advances in Cryptology – EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer-Verlag, 2020, p. 738–768.

[9] C. Guo, L. You, and G. Hu, "A novel biometric identification scheme based on zero-knowledge succinct noninteractive argument of knowledge," *Security and Communication Networks*, vol. 2022, no. 1, p. 2791058, 2022.

[10] P. Kothari, D. Chopra, M. Singh, S. Bhardwaj, and R. Dwivedi, "Incorporating zero-knowledge succinct non-interactive argument of knowledge for blockchain-based identity management with off-chain computations," *CoRR*, vol. abs/2310.19452, 2023.

[11] M. Blanton and D. Murphy, "Privacy preserving biometric authentication for fingerprints and beyond," in *Proceedings of the Fourteenth ACM Conference on Data and Application Security and Privacy*, ser. CODASPY '24. Association for Computing Machinery, 2024, p. 367–378.

[12] J. Lai, T. Wang, S. Zhang, Q. Yang, and S. C. Liew, "Biozero: An efficient and privacy-preserving decentralized biometric authentication protocol on open blockchain," *CoRR*, vol. abs/2409.17509, 2024.

[13] C. Wiraatmaja, https://github.com/christopherwira/anon-auth-zkp-set-membership.

[14] D. Benarroch, M. Campanelli, D. Fiore, K. Gurkan, and D. Kolonelos, "Zero-knowledge proofs for set membership: Efficient, succinct, modular," in *Proc. of the Financial Cryptography and Data Security - 25th International Conference*, ser. Lecture Notes in Computer Science, vol. 12674. Springer, 2021, pp. 393–414.

[15] S. Micali, M. Rabin, and J. Kilian, "Zero-knowledge sets," in *Proc. of the 44th Annual IEEE Symposium on Foundations of Computer Science, 2003.*, 2003, pp. 80–91.

[16] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof systems," *SIAM Journal on Computing*, vol. 18, no. 1, pp. 186–208, 1989.

[17] O. Goldreich, S. Micali, and A. Wigderson, "Proofs that yield nothing but their validity or all languages in np have zero-knowledge proof systems," *J. ACM*, vol. 38, no. 3, p. 690–728, jul 1991.

[18] M. Blum, P. Feldman, and S. Micali, "Non-interactive zero-knowledge and its applications," in *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, ser. STOC '88. Association for Computing Machinery, 1988, p. 103–112.

[19] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Proceedings on Advances in Cryptology—CRYPTO '86*. Springer-Verlag, 1987, p. 186–194.

[20] M. Campanelli, D. Fiore, and A. Querol, "Legosnark: Modular design and composition of succinct zero-knowledge proofs," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*. ACM, 2019, pp. 2075–2092.

[21] L. Grassi, D. Khovratovich, C. Rechberger, A. Roy, and M. Schofnegger, "Poseidon: A new hash function for zero-knowledge proof systems," in *Proc. of the 30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 519–535.

[22] E. Ben Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," in *Proc. of the 2014 IEEE Symposium on Security and Privacy*, 2014, pp. 459–474.

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

[24] K. Abbaszadeh, C. Pappas, J. Katz, and D. Papadopoulos, "Zero-knowledge proofs of training for deep neural networks," Cryptology ePrint Archive, Paper 2024/162, 2024. [Online]. Available: https://eprint.iacr.org/2024/162

[25] P. S. L. M. Barreto and M. Naehrig, "Pairing-friendly elliptic curves of prime order," ser. SAC'05. Berlin, Heidelberg: Springer-Verlag, 2005, p. 319–331.

[26] T. Xie, Y. Zhang, and D. Song, "Orion: Zero knowledge proof with linear prover time," in *Proc. of the Advances in Cryptology – CRYPTO 2022*, Y. Dodis and T. Shrimpton, Eds. Cham: Springer Nature Switzerland, 2022, pp. 299–328.

[27] A. Golovnev, J. Lee, S. Setty, J. Thaler, and R. S. Wahby, "Brakedown: Linear-time and field-agnostic snarks for r1cs," in *Proc. of the Advances in Cryptology – CRYPTO 2023*, H. Handschuh and A. Lysyanskaya, Eds. Cham: Springer Nature Switzerland, 2023, pp. 193–226.

[28] A. Zapico, V. Buterin, D. Khovratovich, M. Maller, A. Nitulescu, and M. Simkin, "Caulk: Lookup arguments in sublinear time," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '22. Association for Computing Machinery, 2022, p. 3121–3134.