

FLAME: Flexible and Lightweight Biometric Authentication Scheme in Malicious Environments

Fuyi Wang

*Singapore University of Technology
and Design*

wong_fuyi@outlook.com

Fangyuan Sun

*Qingdao
University*

sakgofish@gmail.com

Mingyuan Fan

*East China Normal
University*

fmy2660966@gmail.com

Jianying Zhou

*Singapore University of Technology
and Design*

jianying_zhou@sutd.edu.sg

Jin Ma

Clemson University
jin7@clemson.edu

Chao Chen

RMIT University
chao.chen@rmit.edu.au

Jiangang Shu

Guangzhou University
jiangangshu@gmail.com

Leo Yu Zhang

Griffith University
leo.zhang@griffith.edu.au

Abstract—Privacy-preserving biometric authentication (PPBA) enables client authentication without revealing sensitive biometric data, addressing privacy and security concerns. Many studies have proposed efficient cryptographic solutions to this problem based on secure multi-party computation, typically assuming a semi-honest adversary model, where all parties follow the protocol but may try to learn additional information. However, this assumption often falls short in real-world scenarios, where adversaries may behave maliciously and actively deviate from the protocol.

In this paper, we propose, implement, and evaluate **FLAME**, a **F**lexible and **L**ightweight biometric **A**uthentication scheme designed for a **M**alicious **E**nvironment. By hybridizing lightweight secret-sharing-family primitives within two-party computation, **FLAME** carefully designs a line of supporting protocols that incorporate integrity checks with rationally extra overhead. Additionally, **FLAME** enables server-side authentication with various similarity metrics through a cross-metric-compatible design, enhancing flexibility and robustness without requiring any changes to the server-side process. A rigorous theoretical analysis validates the correctness, security, and efficiency of **FLAME**. Extensive experiments highlight **FLAME**'s superior efficiency, with a communication reduction by $97.61\times \sim 110.13\times$ and a speedup of $2.72\times \sim 2.82\times$ (resp. $6.58\times \sim 8.51\times$) in a LAN (resp. WAN) environment, when compared to the state-of-the-art work.

Index Terms—Privacy-preserving protocols, malicious security, secret sharing, biometric authentication.

1. Introduction

Biometric authentication offers a convenient way to authenticate clients' identities based on their unique biological traits, eliminating the need to memorize passwords or manage other secret credentials [11]. To decouple authentication from a specific device, biometric templates are

often outsourced to cloud servers rather than stored locally. This enables clients to authenticate seamlessly across devices, enhancing scalability and flexibility. However, this raises significant privacy and security concerns, as biometric templates are highly private and sensitive. Additionally, unlike passwords, biometric templates—such as fingerprints, facial images, iris scans, and genomic data—cannot be superseded if compromised by cloud servers or other adversaries. Therefore, privacy-preserving biometric authentication (PPBA) schemes [1], [19], [24], [26], [33] have emerged to strengthen the protection of biometric templates. They leverage cryptographic primitives such as homomorphic encryption (HE) [25] and/or multi-party computation (MPC) [17], including garbled circuit (GC) and secret sharing (SS) techniques. PPBA schemes are classified according to their underlying security models, namely the semi-honest and malicious settings. A comprehensive comparison of representative schemes under both models is presented in Table 1.

Related semi-honest schemes. Most existing PPBA studies target the semi-honest setting. Studies [26], [27], [33] employ (fully/partially) HE to achieve privacy-preserving facial recognition. However, the significant computational and communication overhead associated with HE renders it impractical for real-world scenarios. To improve efficiency, many hybrid solutions that combine HE with GC [19], [21], [24], SS [31], or cancelable biometrics [28], have been proposed for PPBA. While these hybrid solutions do achieve performance gains, HE-involved schemes still require the client to retain a private decryption key, which nullifies the advantage of biometric authentication which promotes authentication without the need to remember passwords or manage secret keys. To address this, recent efforts have been shifted towards utilizing MPC primitives, which offer more efficient alternative authentication schemes for various biometrics, such as fingerprint [31], [34], voice [29], and iris [16]. However, these semi-honest-centric studies assume that all parties follow protocols without deviation, which is still challenging to meet in reality [7].

TABLE 1. COMPARISON OF PPBA SCHEMES.

Schemes	Model	Primitives	Security	Integrity	f^{lin}	f^{nonlin}	Metrics	Latency ²
Nieminen <i>et al.</i> [24]	Client-server	HE,GC	○	✗	✗	✗ ¹	Euclidean	High
Wei <i>et al.</i> [33]	Client-server	HE	○	✗	✗	✗ ¹	Cosine	High
Pradel <i>et al.</i> [26]	Client-server	HE	○	✗	✗	✗ ¹	Euclidean	High
Huang <i>et al.</i> [19]	Two servers	HE,GC	○	✗	✗	✗ ¹	Euclidean	High
Wang <i>et al.</i> [31]	Two servers	SS,GC	○	✗	✓	✗	Euclidean	Moderate
Im <i>et al.</i> [20]	Client-server	HE	◐	HE	✗	✗ ¹	Euclidean	Moderate
Bassit <i>et al.</i> [4]	Client-server	HE	◐	ZKP	✗	✗ ¹	Likelihood	Moderate
Al-Mannai <i>et al.</i> [1]	Client-server	HE	◐	ZKP	✗	✗ ¹	Euclidean	Low
Cheng <i>et al.</i> [11]	Two servers	OptSS, FuncSS	●	MAC	✗	✓	Cosine	Moderate
FLAME	Two servers	OptSS, FuncSS	●	MAC	✓	✓	Cosine, Euclidean	Low

○: semi-honest, ◐: malicious client-only, ●: malicious. HE: homomorphic encryption, SS: secret sharing, GC: garbled circuit, OptSS: optimized SS, FuncSS: function SS, f^{lin} and f^{nonlin} : the optimization of linear and non-linear functions, Cosine: Cosine similarity, Euclidean: squared Euclidean distance, Likelihood: log likelihood ratio classifier.

¹ These schemes optimize computation using precomputation and packing techniques, rather than optimizing the protocols.

² The latency is evaluated based on the results reported in their papers.

Related malicious schemes. PPBA schemes against malicious adversaries have recently attracted increasing attention. Broadly, two models have emerged: the client-server interactive model and the fully outsourced two-server model. In the client-server model, existing works typically assume a malicious client, and employ mechanisms such as commitments [20] and the zero-knowledge proof (ZKP) [1], [4] to ensure the integrity of intermediate results from the client. However, these approaches often require frequent client participation and fail to ensure the correctness of server-side computations, limiting their scalability and practicality for resource-constrained clients. To address these challenges, the literature [1], [5] introduced provably PPBA schemes that leverage ZKPs to detect and mitigate malicious behavior from servers under a fully outsourced setting. [3], [11] utilized the SPDZ MPC protocol to learn the biometric authentication result in a secure manner, employing the lightweight message authentication code (MAC) to ensure result integrity.

Maliciously secure protocols are significantly more complex, with computational overhead often exceeding that of semi-honest counterparts by an order of magnitude. Since an equal amount of linear computations are significantly faster than non-linear ones over the secret-sharing domain, existing malicious-against computation schemes focus on optimizing costly non-linear comparisons [11] or improving efficiency using packing techniques [4], [20], while largely overlooking improvements in linear computation efficiency. However, in PPBA, the linear similarity computation remains a major performance bottleneck due to the large volume and high dimensionality of reference biometric templates, leaving substantial room for optimization (**Challenge ①**). Moreover, these schemes rely on over-the-threshold Cosine similarity or squared Euclidean distance for biometric authentication, which introduces two extra challenges. **Challenge ②**: Determining the appropriate threshold is challenging due to its variability across different applications

and environments, making it difficult to establish a one-size-fits-all solution. A poorly chosen threshold can lead to either high false acceptance rates or high false rejection rates, both of which compromise the effectiveness of the authentication system. **Challenge ③**: Different types of biometric traits, each with distinct characteristics, require different similarity metrics. These schemes typically support only a single similarity metric, which limits their flexibility and robustness. Switching to a different metric often requires corresponding modifications to the authentication protocol. For example, Cosine similarity requires the computed value to exceed a threshold for successful authentication, whereas Euclidean distance requires it to fall below a threshold.

Our contributions. To address these challenges, we design, implement, and evaluate FLAME, a fast, flexible, and secure biometric authentication scheme built on a two-party computation (2PC) setup. In FLAME, even if both servers access intermediate results, they cannot reconstruct the original biometric templates, thereby preserving client privacy. It further ensures malicious security under a dishonest majority, guaranteeing the integrity of authentication outcomes (e.g., preventing forged “success” or “failure” results). Also, FLAME is seamlessly compatible with both Cosine similarity and Euclidean distance metrics by bridging the semantic gap, ensuring flexible applicability across different biometric authentication techniques. To boost efficiency, FLAME integrates lightweight cryptographic primitives with an offline-online paradigm. It significantly reduces online authentication latency while achieving malicious security with only $\approx 2\times$ overhead of semi-honest PPBA schemes. This enables real-time, large-scale biometric authentication with strong security guarantees. In summary, our contributions are threefold.

- We present FLAME, a flexible and crypto-friendly biometric authentication scheme that supports diverse similarity metrics in the malicious setting. Based on a 2PC setup, one of the servers is always assumed to be ma-

icious, ensuring strong security against internal adversaries.

- Based on secret-sharing-family primitives, we customize a suite of secure linear and non-linear protocols for PPBA by leveraging an offline-online paradigm. These protocols are carefully crafted to minimize online computational overhead and reduce communication to a single round.
- We formally prove the correctness, efficiency, and security of FLAME. We conduct extensive experiments and compare our design with state-of-the-art studies. The results highlight that FLAME achieves high efficiency and scalability, with face recognition serving as a representative application.

2. Secret Sharing

Optimized secret sharing (OptSS) [6] is an enhanced arithmetic secret-sharing approach, dividing operations in the secret-sharing domain into pre-processing (i.e., offline) and online phases. In the pre-processing phase, correlated random offset shares (λ -values) are generated for the input and output wires of each gate of the arithmetic circuit. These input-independent shares can be executed prior to the online phase, where the actual function f is evaluated via efficient secure computation. This paper adopts the 2-out-of-2 OptSS over the ring $\mathbb{Z}_{2^l}$ and we now give the formal definition. On the secret $\langle x \rangle \in \mathbb{Z}_{2^l}$ ($\langle x \rangle_0 + \langle x \rangle_1 = x$), parties call $\Pi_{\text{OptSS}}(\langle x \rangle)$ to learn OptSS-based shares $[\![x]\!]_\theta = (\Delta_x, \langle \lambda_x \rangle_\theta) \in \mathbb{Z}_{2^l}$, s.j. $\Delta_x = x + \lambda_x \pmod{2^l}$, where $\langle \lambda_x \rangle$ is the x -associated random offset and $\lambda_x = \sum_{\theta=0}^1 \langle \lambda_x \rangle_\theta$. With 2 shares $(\Delta_x, \langle \lambda_x \rangle_0)$, $(\Delta_x, \langle \lambda_x \rangle_1)$, the secret x can be opened over the plaintext space $\mathbb{Z}$, i.e., $x = \Delta_x - \langle \lambda_x \rangle_0 - \langle \lambda_x \rangle_1 \pmod{2^l}$. Given one public value p and two secrets $\langle x \rangle, \langle y \rangle$, where $\mathcal{P}_\theta \ (\forall \theta \in \{0, 1\})$ holds OptSS shares $(\Delta_x, \langle \lambda_x \rangle_\theta)$ and $(\Delta_y, \langle \lambda_y \rangle_\theta)$, four basic secure computations for addition (add) and multiplication (mult) are defined as follows.

- $\Pi_{\text{add}} \leftarrow [\![x]\!] + [\![y]\!]$: $\mathcal{P}_\theta$ computes $\Delta_z = \Delta_x + \Delta_y$ and $\langle \lambda_z \rangle_\theta = \langle \lambda_x \rangle_\theta + \langle \lambda_y \rangle_\theta$.
- $[\![x]\!] + p$: $\mathcal{P}_\theta$ computes $\Delta_z = \Delta_x + p$ and $\langle \lambda_z \rangle_\theta = \langle \lambda_x \rangle_\theta$.
- $\Pi_{\text{mult}} \leftarrow [\![x]\!] \cdot [\![y]\!]$: A shared multiplication triple $(\langle a \rangle, \langle b \rangle, \langle c \rangle)$, values $\delta_x, \delta_y, \langle \lambda_z \rangle$ are generated, where $c = ab$, $\delta_x = a - \lambda_x, \delta_y = b - \lambda_y$. Two servers first compute the shared $\langle \Delta_z \rangle_\theta = \theta \cdot (\Delta_x + \delta_x)(\Delta_y + \delta_y) - \langle a \rangle_\theta(\Delta_y + \delta_y) - (\Delta_x + \delta_x)\langle b \rangle_\theta + \langle c \rangle_\theta + \langle \lambda_z \rangle_\theta$. $\forall \theta \in \{0, 1\}$, $\mathcal{P}_\theta$ sends $\langle \Delta_z \rangle_\theta$ to $\mathcal{P}_{1-\theta}$ to open Δ_z . Finally, the OptSS shares $(\Delta_z, \langle \lambda_z \rangle_\theta)$ for the product of $\langle x \rangle$ and $\langle y \rangle$ is obtained.
- $[\![x]\!] \cdot p$: $\mathcal{P}_\theta$ computes $\Delta_z = \Delta_x \cdot p$ and $\langle \lambda_z \rangle_\theta = \langle \lambda_x \rangle_\theta$.

Authenticated secret sharing (AuthSS) from SPD $\mathbb{Z}_{2^l}$ [13] is a cryptographic technique with the information-theoretic message authentication code (MAC) that ensures both the confidentiality and integrity of secret-shared values. It is a foundational building block in secure multiparty computation (MPC) protocols against malicious adversaries. Due to the presence of zero divisors in $\mathbb{Z}_{2^l}$, the parties must perform computations over a

larger ring modulo 2^{l+s} , while the security and correctness is only guaranteed modulo 2^l . Specifically, assume each party holds a shared $\langle \phi \rangle \xleftarrow{\$} \mathbb{Z}_{2^s}$ of a secret global MAC key $\phi = \sum_{\theta=0}^1 \langle \phi \rangle_\theta \pmod{2^{l+s}}$. Given a shared value $\langle x \rangle \in \mathbb{Z}_{2^{l+s}}$, each party $\mathcal{P}_\theta$ holds the authenticated share $\{\langle x \rangle, \langle m \rangle \in \mathbb{Z}_{2^{l+s}}, \langle \phi \rangle \in \mathbb{Z}_{2^s}\}$. These components satisfy $\sum_{\theta=0}^1 \langle m \rangle_\theta \equiv_{l+s} \left(\sum_{\theta=0}^1 \langle x \rangle_\theta \right) \cdot \left(\sum_{\theta=0}^1 \langle \phi \rangle_\theta \right)$, where $\equiv_{l+s}$ denotes ring module 2^{l+s} . Similarly, for a function f , the output includes the shares $\langle f(x) \rangle$ and $\langle \phi f(x) \rangle \in \mathbb{Z}_{2^{l+s}}$. Then, the authenticator performs $\langle d \rangle \leftarrow \langle \phi f(x) \rangle - \langle \phi \rangle \cdot \langle f(x) \rangle$. If $d = 0$, the verification succeeds; otherwise, abort.

Function secret sharing (FuncSS) [8], [9] within the 2PC setting divides a function f into 2 succinct function shares $\{f_0, f_1\}$. Each party receives one of the function shares, which reveals no information about f . For any input x , there exists $f_0(x) + f_1(x) = f(x)$. FLAME leverages the comparison FuncSS $f_{a,b}^<$ [8], [18], realized through distributed point functions (DPFs). Here, $f_{a,b}^<(x)$ evaluates to b if $x < a$, and 0 otherwise. Specifically, the comparison FuncSS $f_{a,b}^<$ consists of a pair of algorithms $\{\text{Gen}^<(a, b), \text{Eval}^<(\theta, \kappa_\theta, x)\}$. The key generation algorithm $\text{Gen}^<(a, b)$ outputs a pair of keys $\{\kappa_0, \kappa_1\}$, where each key implicitly represents $f_\theta^<$. For the party $\theta \in \{0, 1\}$, with the key κ_θ and the public input x , the evaluation algorithm $\text{Eval}^<(\theta, \kappa_\theta, x)$ outputs y_θ , i.e., the value of $f_\theta^<(x)$, where $f_{a,b}^<(x) = \sum_{\theta=0}^1 y_\theta$.

Multiple ideal functionalities $\mathcal{F}_{\text{ABB}}$ are used in FLAME as arithmetic black-box operations, including $\mathcal{F}_{\text{rand}}$, $\mathcal{F}_{\text{coin}}$, $\mathcal{F}_{\text{triple}}$, and others. These functionalities can be securely realized using well-established protocols in the malicious setting. Due to space constraints, we briefly outline their roles and refer to [12], [13] for further details.

- $\mathcal{F}_{\text{share}}(\mathbb{Z}_{2^{l+s}})$: Sample a pair of additive SS shares $\langle x \rangle \xleftarrow{\$} \mathbb{Z}_{2^{l+s}}$ for the input x , such that $\langle x \rangle_0 + \langle x \rangle_1 \equiv_{l+s} x$.
- $\mathcal{F}_{\text{rand}}(\mathbb{Z}_{2^{l+s}})$: Sample a random r , and share $\langle r \rangle \xleftarrow{\$} \mathbb{Z}_{2^{l+s}}$ between two parties.
- $\mathcal{F}_{\text{coin}}(\mathbb{Z}_{2^{l+s}})$: Sample a random $r \xleftarrow{\$} \mathbb{Z}_{2^{l+s}}$, and output r to two parties.
- $\mathcal{F}_{\text{triple}}(\mathbb{Z}_{2^{l+s}})$: Sample randoms a, b , compute $c = a \cdot b$, and output shared $\langle a \rangle_\theta, \langle b \rangle_\theta, \langle c \rangle_\theta \xleftarrow{\$} \mathbb{Z}_{2^{l+s}}$ to party θ .
- $\mathcal{F}_{\text{mult}}(\mathbb{Z}_{2^{l+s}})$: Take two additive shares $\langle x \rangle_\theta$ and $\langle y \rangle_\theta$ from party θ , and output shared $\langle z \rangle_\theta \equiv_{l+s} \langle x \cdot y \rangle_\theta$ to party θ .

3. System Overview

System model. Let us examine a scenario where a client requests access to a service, such as online tax declaration, using a biometric-based authentication system FLAME. This scenario is illustrated in Figure 1, consisting of three entities: the client $\mathcal{C}$, the service provider $\mathcal{S}$, and two cloud servers $\mathcal{P}_0$ and $\mathcal{P}_1$. Biometric authentication generally involves two phases: enrollment, where the client registers their reference biometric templates, and authentication, where the client is

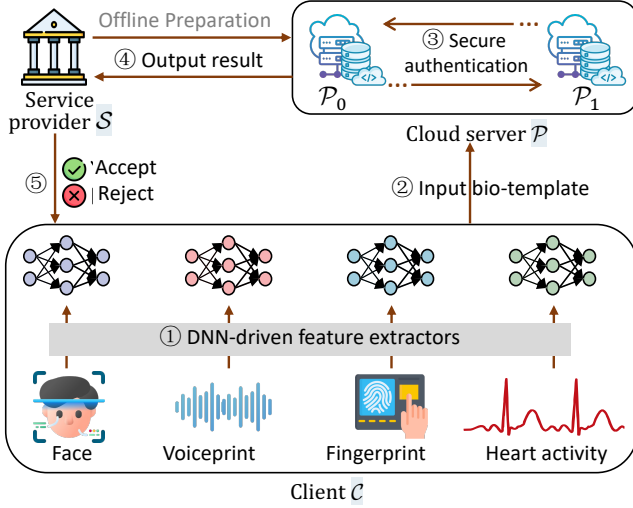


Figure 1. The system model.

verified for subsequent logins. In FLAME, we assume that the enrollment phase has been completed and focus on the authentication phase. Specifically, ① a client $\mathcal{C}$ generates a fresh biometric template via DNN-driven feature extractors and ② securely distributes it to $\mathcal{P}_0$ and $\mathcal{P}_1$. ③ $\mathcal{P}_0$ and $\mathcal{P}_1$ perform secure authentication by computing the similarity between the $\mathcal{C}$'s fresh biometric template and the stored references. They either identify the top- k closest matches or determine whether the similarity score exceeds a predefined threshold τ . ④ After authentication, $\mathcal{P}_0$ and $\mathcal{P}_1$ share the output with $\mathcal{S}$, cryptographically ensuring it is free from server-side tampering. ⑤ Upon receiving the authenticated decision result, $\mathcal{S}$ determines whether the client is granted access to the requested service. In the two-server setting, each server retains the shares of biometric templates and intermediate results and no single server can unilaterally access the private information, thereby safeguarding privacy. This setting has been widely adopted in various secure applications [11], [23], [30], [32].

Threat model and security. There are two standard security models for outsourced computation: semi-honest security and malicious security. A semi-honest adversary follows the specifications of the protocols, but may attempt to infer additional information from the shares it handles. A malicious adversary may deviate arbitrarily from the protocols in an attempt to manipulate the computations. In FLAME, the two computing servers are assumed to be malicious, while the client and the service provider are considered semi-honest. Focusing specifically on the malicious security of the two-server computation (i.e., 2PC), we formally define the security of FLAME within the real/ideal world. For simplicity, we assume that $\mathcal{P}_0$ is corrupted; the definition applies symmetrically to $\mathcal{P}_1$.

Definition 1. Let $\mathcal{F} = \{\mathcal{F}_0, \mathcal{F}_1\}$ be an ideal-world functionality and Π be a real-world protocol semantically-securely computing $\mathcal{F}$. For any malicious adversary $\mathcal{A}$, there exists a probabilistic polynomial-time (PPT) simulator Sim such

that

$$\begin{aligned} & \{\text{View}_{\mathcal{A}(z)}^{\Pi}(1^{l,s}, x, y), \text{Out}^{\Pi}(1^{l,s}, x, y)\}_{l,s,z,x,y} \\ & \stackrel{c}{=} \{\text{Sim}_{\mathcal{A}(z)}^{\mathcal{F}}(1^{l,s}, x, \mathcal{F}_0(x, y)), \mathcal{F}(x, y)\}_{l,s,z,x,y} \end{aligned}$$

where $l, s \in \mathbb{N}^+$ are the security parameters, $z \in \{0, 1\}^*$ is the auxiliary information, $\text{View}_{\mathcal{A}}^{\Pi}$ denotes the final view of the corrupted $\mathcal{P}_0$ in the real world, $\text{Sim}^{\mathcal{F}}$ denotes the view of the corrupted $\mathcal{P}_0$ generated by Sim , Out^{Π} denotes the output of the protocol Π , $\mathcal{F}(x, y)$ denotes the output of the ideal functionality $\mathcal{F}$, and $\stackrel{c}{=}$ denotes computational indistinguishability against PPT adversaries except for a negligible advantage.

Design goals. Based on the adversarial model and system requirements, we establish the following design goals for FLAME:

- **Privacy.** FLAME must ensure that the client's biometric data, both fresh and pre-stored templates, remain confidential and are never exposed in plaintext to the servers and the service provider.
- **Integrity.** FLAME must guarantee that the authentication result is computed correctly and has not been tampered with, even if some servers behave maliciously.
- **Efficiency.** Given the latency sensitivity of authentication tasks, FLAME must minimize computational and communication overhead while maintaining strong security guarantees.
- **Accuracy.** FLAME must yield an authentication accuracy comparable to plaintext biometric systems, achieving a success rate no lower than 95%.
- **Scalability.** FLAME should support large-scale deployments, efficiently handling many clients and high-dimensional biometric templates without significantly impacting performance.

4. Supporting Protocols

This section leverages secret-sharing-family (SS-family) primitives to design linear inner product and the non-linear comparison protocols for FLAME. The high-level implementation of the functionality $\mathcal{F}_f$ is formally described in Figure 2. At a high level, FLAME splits each protocol into offline and online phases, aiming to alleviate the computational burden and minimize the communication rounds in the online phase. In the data-independent offline phase, heavy cryptographic operations—such as generating randomness, Beaver triples, and FuncSS keys—are executed with the assistance of a trusted dealer. In FLAME, the service provider $\mathcal{S}$ acts as the trusted dealer, distributing reliable correlated randomness to the two servers. For clarity, the main notations in this paper are summarized in Table 2.

4.1. Secure Inner Product Protocol

For n -dimensional OptSS shared $(\Delta_X, \langle \lambda_X \rangle)$, ϕ -involved $(\Delta_{\phi X}, \langle \lambda_{\phi X} \rangle)$, and $(\Delta_Y, \langle \lambda_Y \rangle)$, $\langle \lambda \rangle$ -related shares are independent of the actual inputs and stored in $\mathcal{P}_0$ and

TABLE 2. THE SUMMARY OF NOTATIONS

θ	Party identity $\theta \in \{0, 1\}$.
$\mathcal{P}_0, \mathcal{P}_1$	Parties (i.e., Servers) running secure interactive computation.
$\mathbf{X}, X, x$	Tensor/Matrix, vector, element.
$[1, x]$	Set of positive integers $\{1, \dots, x\}$.
$\langle x \rangle_\theta$	Arithmetic share of x held by party θ , where $x = \langle x \rangle_0 + \langle x \rangle_1$.
$\llbracket x \rrbracket_\theta$	Optimized arithmetic share of x held by party θ , where $\llbracket x \rrbracket_\theta = (\Delta_x, \langle \lambda_x \rangle_\theta)$ and $\Delta_x = x + \lambda_x$.
$X[i]$	The i -th element in vector X .
τ	Authentication threshold.
ϕ	Secret MAC key.
$\otimes$	Inner product operation for arithmetic values.
$\cdot$	Multiplication operation for arithmetic values.

Linear function $f^{\text{lin}}: \mathbb{Z}_{2^l} \rightarrow \mathbb{Z}_{2^l}$.

Input: $\mathcal{P}_\theta$ inputs $\llbracket \phi x \rrbracket_\theta$, $\llbracket x \rrbracket_\theta$, and $\llbracket y \rrbracket_\theta$, $\forall \theta \in \{0, 1\}$.

Output: $\mathcal{P}_\theta$ learns $\llbracket f^{\text{lin}}(x, y) \rrbracket_\theta$ and $\llbracket \phi f^{\text{lin}}(x, y) \rrbracket_\theta$, $\forall \theta \in \{0, 1\}$.

Non-linear function $f^{\text{non-lin}}: \mathbb{Z}_{2^l} \rightarrow \mathbb{Z}_{2^l}$.

Input: $\mathcal{P}_\theta$ inputs $\llbracket x \rrbracket_\theta$ and $\llbracket \phi \rrbracket_\theta$, $\forall \theta \in \{0, 1\}$.

Output: $\mathcal{P}_\theta$ learns $\llbracket f^{\text{non-lin}}(x) \rrbracket_\theta$ and $\llbracket \phi f^{\text{non-lin}}(x) \rrbracket_\theta$, $\forall \theta \in \{0, 1\}$.

Figure 2. Functionality $\mathcal{F}_f$.

Protocol $\prod_{\text{SecIP}}$

Offline Phase:

Input: $\langle \lambda_X \rangle_\theta, \langle \lambda_{\phi X} \rangle_\theta, \langle \lambda_Y \rangle_\theta$.

- 1) Call $\mathcal{F}_{\text{rand}}$ to learn the shared $\langle \lambda \rangle$ -values: $\langle \lambda_z \rangle$ and $\langle \lambda_{\phi z} \rangle$.
 - 2) Call $\mathcal{F}_{\text{triple}}$ to learn the shared $(\langle A_1 \rangle, \langle B \rangle, \langle C_1 \rangle)$ and $(\langle A_2 \rangle, \langle B \rangle, \langle C_2 \rangle)$, where $A_1 \cdot B = C_1$ and $A_2 \cdot B = C_2$.
 - 3) $\forall \theta \in \{0, 1\}$, $\mathcal{P}_\theta$ locally computes $\langle \delta_X \rangle_\theta = \langle A_1 \rangle_\theta - \langle \lambda_X \rangle_\theta$, $\langle \delta_{\phi X} \rangle_\theta = \langle A_2 \rangle_\theta - \langle \lambda_{\phi X} \rangle_\theta$, and $\langle \delta_Y \rangle_\theta = \langle B \rangle_\theta - \langle \lambda_Y \rangle_\theta$.
 - 4) Open $\langle \delta_X \rangle, \langle \delta_{\phi X} \rangle, \langle \delta_Y \rangle$ to get the clear $\delta_X, \delta_{\phi X}, \delta_Y$.
- Output:** $\mathcal{P}_\theta$ learns $\langle A_1 \rangle_\theta, \langle A_2 \rangle_\theta, \langle B \rangle_\theta, \langle C_1 \rangle_\theta, \langle C_2 \rangle_\theta, \langle \lambda_z \rangle_\theta, \langle \lambda_{\phi z} \rangle_\theta, \delta_{\phi X}, \delta_X, \delta_Y$.

Online Phase:

Input: For inputs X and Y with length n , $\mathcal{P}_\theta$ holds $(\Delta_X, \langle \lambda_X \rangle_\theta), (\Delta_{\phi X}, \langle \lambda_{\phi X} \rangle_\theta)$, and $(\Delta_Y, \langle \lambda_Y \rangle_\theta)$.

- 1) $\mathcal{P}_\theta$ locally computes $\langle \Delta_z \rangle_\theta = \sum_{i=1}^n (\theta \cdot (\Delta_X[i] + \delta_X[i]) (\Delta_Y[i] + \delta_Y[i]) - \langle A_1[i] \rangle (\Delta_Y[i] + \delta_Y[i]) - (\Delta_X[i] + \delta_X[i]) \langle B[i] \rangle + \langle C_1[i] \rangle)) + \langle \lambda_z \rangle_\theta$.
 - 2) $\mathcal{P}_\theta$ locally computes $\langle \Delta_{\phi z} \rangle_\theta = \sum_{i=1}^n (\theta \cdot (\Delta_{\phi X}[i] + \delta_{\phi X}[i]) (\Delta_Y[i] + \delta_Y[i]) - \langle A_2[i] \rangle (\Delta_Y[i] + \delta_Y[i]) - (\Delta_{\phi X}[i] + \delta_{\phi X}[i]) \langle B[i] \rangle + \langle C_2[i] \rangle)) + \langle \lambda_{\phi z} \rangle_\theta$.
 - 3) Open $\langle \Delta_z \rangle, \langle \Delta_{\phi z} \rangle$ to get the clear $\Delta_z, \Delta_{\phi z}$.
 - 4) The servers run $\prod_{\text{MACCheck}}$, if the check fails, abort.
- Output:** $\mathcal{P}_\theta$ learns $(\Delta_z, \langle \lambda_z \rangle_\theta)$ and $(\Delta_{\phi z}, \langle \lambda_{\phi z} \rangle_\theta)$.

Figure 3. Construction of secure inner product protocol $\prod_{\text{SecIP}}$.

$\mathcal{P}_1$ in advance. For an n -dimensional vector, the existing inner product computation has a complexity of $\mathcal{O}(n)$ over the secret-sharing domain. Inspired by [35], we introduce an inner product protocol $\prod_{\text{SecIP}}$ with online communication independent of the vector size n , while also optimizing the computation and communication overhead in the of-

Protocol $\prod_{\text{MACCheck}}$

On inputting a set of shared values $\{\langle X[i] \rangle\}_{i=1}^n$, shared MAC values $\{\langle \phi X[i] \rangle\}_{i=1}^n$, $\prod_{\text{MACCheck}}$ executes with the inherent shared MAC key $\langle \phi \rangle$ and outputs True if the MAC check passes and abort otherwise.

- Servers call $\langle r \rangle \leftarrow \mathcal{F}_{\text{rand}}(\mathbb{Z}_{2^s})$ and $\langle \phi r \rangle \leftarrow \mathcal{F}_{\text{mult}}(\langle \phi \rangle, 2^l \langle r \rangle)$.
- Servers call $\mathcal{F}_{\text{coin}}(\mathbb{Z}_{2^s})$ to sample public randomness $\{P[i]\}_{i=1}^n$.
- Servers locally compute $\langle y_0 \rangle \leftarrow 2^l \langle r \rangle + \sum_{i=1}^n P[i] \cdot \langle X[i] \rangle$ and $\langle y_1 \rangle \leftarrow \langle \phi r \rangle + \sum_{i=1}^n P[i] \langle \phi X[i] \rangle \pmod{2^{l+s}}$.
- Each server broadcasts $\langle y_0 \rangle \in \mathbb{Z}^{l+s}$ and all servers compute $y_0 = \langle y_0 \rangle_0 + \langle y_0 \rangle_1$.
- Servers locally compute $\langle z \rangle \leftarrow \langle y_1 \rangle - y_0 \langle \phi \rangle$, then open their commitments.
- Servers verify $\langle z \rangle_0 + \langle z \rangle_1 \equiv_{l+s} 0$. If it outputs True then all servers proceed and abort otherwise.

Figure 4. The batch MAC check protocol.

fine phase. Specifically, existing OptSS-based inner product protocols employ an open-then-sum strategy. This involves invoking $\prod_{\text{mult}}$ (described in Sec. 2) n times, requiring the opening of $\langle \Delta_Z[i] \rangle = \langle Z[i] \rangle + \langle \lambda_Z[i] \rangle$ ($i \in \{1, n\}$) for each $\prod_{\text{mult}}$ to retrieve $\Delta_Z[i]$, followed by summing up these values to learn $\Delta_z = \sum_{i=1}^n \Delta_Z[i]$. The key insight is that the open-then-sum strategy is equivalent to a sum-then-open strategy. In this optimized strategy, the intermediate shares $\langle Z[i] \rangle$ are summed locally first with only a single random shared $\langle \lambda_z \rangle$, resulting in a single aggregated share $\langle \Delta_z \rangle = \sum_{i=1}^n \langle Z[i] \rangle + \langle \lambda_z \rangle$. $\langle \Delta_z \rangle$ is then opened to retain Δ_z . With this strategy, we observe that the MAC-related computations for $\Delta_{\phi z}$ do not need the extra full multiplication triples. Instead, these computations can share the part of multiplication triples (i.e., B) required for Δ_z 's computation (refer to steps 2) and 3) of the offline phase in Figure. 3). After completing the secure computation, two servers collaboratively verify the integrity of each revealed value with the MAC key ϕ . Figure 3 presents the detailed online and offline phases for $\prod_{\text{SecIP}}$.

To ensure data integrity, the MAC check protocol $\prod_{\text{MACCheck}}$ is used to detect discrepancies by verifying the consistency between shared data and its corresponding MAC-related values—any introduced errors will disrupt this consistency. To improve the efficiency of this verification, a batch MAC check protocol is developed (see Figure 4), which has been adopted in prior works [13], [35]. This protocol enables the simultaneous verification of multiple values in a constant number of rounds, significantly reducing communication overhead. Its compact and scalable design makes it particularly suitable for secure computation settings.

Theorem 1 (The correctness of $\prod_{\text{SecIP}}$). *For any OptSS-shared input vectors $(\Delta_X, \langle \lambda_X \rangle_\theta)$, $(\Delta_{\phi X}, \langle \lambda_{\phi X} \rangle_\theta)$, and $(\Delta_Y, \langle \lambda_Y \rangle_\theta)$, the $\prod_{\text{SecIP}}$ yields the correct results $(\Delta_z, \langle \lambda_z \rangle_\theta)$ and $(\Delta_{\phi z}, \langle \lambda_{\phi z} \rangle_\theta)$ if the conditions $\Delta_z - \lambda_z = z = X \otimes Y =$*

$\sum_{i=1}^n X[i] \cdot Y[i]$ and $\Delta_{\phi z} - \lambda_{\phi z} = \phi z = \phi X \otimes Y = \sum_{i=1}^n \phi X[i] \cdot Y[i]$ are satisfied, with $\otimes$ represents the inner product operation.

Proof. Π_{SecIP} consists of offline and online phases described in Figure 3. The correctness of the offline phase is guaranteed through two main components. First, the functionalities $\mathcal{F}_{\text{rand}}$ and $\mathcal{F}_{\text{triple}}$ reliably generate the required λ -values and multiplication triples, guaranteeing their correctness. Second, the δ -values, which prepare inputs for the online phase, are computed locally using the relations $\delta_X = A_1 - \lambda_X$, $\delta_{\phi X} = A_2 - \lambda_{\phi X}$, and $\delta_Y = B - \lambda_Y$, ensuring proper input adjustments for subsequent evaluation.

Next, we prove the correctness of the online phase of the Π_{SecIP} protocol. Before producing the shared value $\langle z \rangle = \langle X \otimes Y \rangle$, we first establish the correctness of $\langle X[i] \cdot Y[i] \rangle$ for each i , as outlined below:

$$\begin{aligned} & \langle Z[i] \rangle_0 + \langle Z[i] \rangle_1 \\ &= (\Delta_X[i] + \delta_X[i]) (\Delta_Y[i] + \delta_Y[i]) - \langle A[i] \rangle_1 (\Delta_Y[i] + \delta_Y[i]) \\ & \quad - (\Delta_X[i] + \delta_X[i]) \langle B[i] \rangle_1 + \langle C[i] \rangle_1 - \langle A[i] \rangle_0 (\Delta_Y[i] + \delta_Y[i]) \\ & \quad - (\Delta_X[i] + \delta_X[i]) \langle B[i] \rangle_0 + \langle C[i] \rangle_0 \\ & \quad \# \Delta_X[i] + \delta_X[i] = X[i] + A[i] \leftarrow X[i] + \lambda_X[i] + A[i] - \lambda_X[i] \\ & \quad \# \Delta_Y[i] + \delta_Y[i] = Y[i] + B[i] \leftarrow Y[i] + \lambda_Y[i] + B[i] - \lambda_Y[i] \\ &= (X[i] + A[i])(Y[i] + B[i]) - A[i](Y[i] + B[i]) \\ & \quad - (X[i] + A[i])B[i] + C[i] \\ &= X[i] \cdot Y[i]. \end{aligned}$$

By summing over all $i \in [1, n]$, the protocol ensures that $\langle z \rangle = \sum_{i=1}^n \langle X[i] \cdot Y[i] \rangle$ is correct. Consequently, $\Delta_z = z + \lambda_z$ is proven correct by opening $\langle \Delta_z \rangle$, where $\langle \Delta_z \rangle = \langle z \rangle + \langle \lambda_z \rangle \leftarrow \sum_{i=1}^n \langle X[i] \cdot Y[i] \rangle + \langle \lambda_z \rangle$. The correctness of $\Delta_{\phi z} = \phi X \otimes Y + \lambda_{\phi z}$ is established in the same manner. $\square$

4.2. Secure Comparison Protocol

The FuncSS-based comparison operation presented in Sec. 2 operates with $f_{a,b}^<(x)$, designed for less-than evaluations where $x < a \Rightarrow b$. However, there are two challenges in employing $f_{a,b}^<(x)$ for FLAME. *Challenge 1:* The generic FLAME requires functionality for greater-than comparisons, specifically to assess whether a similarity score s exceeds a predefined authentication threshold τ or retrieve the top- k most similar matches. This can be expressed as $(s \geq \tau) ? b : 0$, where b represents the output when the condition is satisfied. *Challenge 2:* Extending this non-verifiable $f_{a,b}^<(x)$ to a lightweight, verifiable 2PC-FuncSS construction poses another challenge. To address these issues, we make two key technical observations.

- *For challenge 1:* We employ the observation that $f_{a,b}^>(x) = b + f_{a,-b}^<(x)$ where $x \geq a \Rightarrow b$. This formulation reuses the less-than comparison functionality, thereby eliminating the need for introducing new crypto-primitives or altering the protocol's core structure. This transformation incurs no additional overhead in terms of computation or communication. The additive operation $+$ within the secret-sharing domain is inherently *free*, as it involves purely local computation without any inter-party interaction.

Protocol Π_{SecCMP}

Offline Phase:

Input: $\langle \phi \rangle_\theta$ and $\langle \lambda_x \rangle_\theta$.

- 1) Let $\mathbf{b} = (b_0, b_1) = (1, \phi)$ and $a = \lambda_x$.
 - 2) $(\kappa'_0, \kappa'_1) \leftarrow \text{Gen}_{a, -\mathbf{b}}^<$.
 - 3) Call $\mathcal{F}_{\text{share}}$ to share b_0 and b_1 to learn the shared $\langle b_0 \rangle, \langle b_1 \rangle \leftarrow \mathbb{Z}_{2^{l+s}}$, s.t., $\langle b_0 \rangle_0 + \langle b_0 \rangle_1 \equiv_{l+s} b_0$, and $\langle b_1 \rangle_0 + \langle b_1 \rangle_1 \equiv_{l+s} b_1$.
 - 4) Let $\kappa_\theta = \kappa'_\theta || \langle \mathbf{b} \rangle_\theta$ for $\theta \in \{0, 1\}$.
 - 5) Call $\mathcal{F}_{\text{rand}}$ to learn the shared $\langle \lambda_z \rangle, \langle \lambda_{\phi z} \rangle$.
- Output:** $\mathcal{P}_\theta$ learns $\kappa_\theta, \langle \lambda_z \rangle_\theta, \langle \lambda_{\phi z} \rangle_\theta$.

Online Phase:

Input: For any inputs x , $\mathcal{P}_\theta$ holds $(\Delta_x, \langle \lambda_x \rangle_\theta)$.

- 1) Parse $\kappa_\theta = \kappa'_\theta || \langle \mathbf{b} \rangle_\theta$.
 - 2) Set $(\langle \gamma_0 \rangle_\theta, \langle \gamma_1 \rangle_\theta) \leftarrow \text{Eval}_{a, -\mathbf{b}}^<(\theta, \kappa'_\theta, \Delta_x)$.
 - 3) Set $\langle \Delta_z \rangle_\theta \leftarrow \langle \gamma_0 \rangle_\theta + \langle b_0 \rangle_\theta + \langle \lambda_z \rangle_\theta$.
 - 4) Set $\langle \Delta_{\phi z} \rangle_\theta \leftarrow \langle \gamma_1 \rangle_\theta + \langle b_1 \rangle_\theta + \langle \lambda_{\phi z} \rangle_\theta$.
 - 5) Open $\langle \Delta_z \rangle, \langle \Delta_{\phi z} \rangle$ to get the clear $\Delta_z, \Delta_{\phi z}$.
 - 6) The servers run Π_{MACCheck} , if the check fails, abort.
- Output:** $\mathcal{P}_\theta$ learns $(\Delta_z, \langle \lambda_z \rangle_\theta), (\Delta_{\phi z}, \langle \lambda_{\phi z} \rangle_\theta)$.

Figure 5. Construction of secure comparison protocol Π_{SecCMP} .

- *For challenge 2:* The shares of ϕ -related results (i.e., $\phi f_{a,b}^>(x)$) are either 0 or ϕb , depending on whether $f_{a,b}^>(x)$ is 0 or b . Inspired by [8], [18], we employ $f_{a,b}^>(x)$ to output a pair of authenticated coefficients. More concretely, the construction outputs the coefficients $\mathbf{b} = (b_0, b_1) = (b, \phi b)$ for $f_{a,b}^>(x)$ when $x \geq a$, otherwise $(0, 0)$.

Notably, in FLAME, the value of b is fixed to 1, i.e., $\mathbf{b} = (1, \phi)$. Building on these two observations, we propose a verifiable greater-than comparison protocol Π_{SecCMP} . The detailed procedures for both the online and offline phases of Π_{SecCMP} are illustrated in Figure 5.

Theorem 2 (The correctness of Π_{SecCMP}). *Given any OptSS-shared input $(\Delta_x, \langle \lambda_x \rangle)$, Π_{SecCMP} yields the correct results $(\Delta_z, \langle \lambda_z \rangle)$ and $(\Delta_{\phi z}, \langle \lambda_{\phi z} \rangle)$ if the conditions $\Delta_z - \lambda_z = 1\{x \geq 0\}$ and $\Delta_{\phi z} - \lambda_{\phi z} = \phi\{x \geq 0\}$ are satisfied.*

Proof. Likewise, Π_{SecCMP} consists of offline and online phases described in Figure 5. The offline phase involves invoking the key generation algorithm $\text{Gen}^<$ to produce a pair of keys $\{\kappa'_0, \kappa'_1\}$, $\mathcal{F}_{\text{share}}$ to generate related additive shares, and $\mathcal{F}_{\text{rand}}$ to reliably generate the required λ -values. The correctness of $\text{Gen}^<$ and $\text{Eval}^<$ has been directly proven in [8]. Now, let's prove the correctness of the online phase of Π_{SecCMP} in producing $\Delta_z, \langle \lambda_z \rangle$, as outlined below:

$$\begin{aligned} & \langle \Delta_z \rangle_0 + \langle \Delta_z \rangle_1 \\ &= \text{Eval}_{\lambda_x, -1}^<(0, \kappa'_0, \Delta_x)_0 + \langle 1 \rangle_0 + \langle \lambda_z \rangle_0 \\ & \quad + \text{Eval}_{\lambda_x, -1}^<(1, \kappa'_1, \Delta_x)_1 + \langle 1 \rangle_1 + \langle \lambda_z \rangle_1 \\ &= -1\{x + \lambda_x < \lambda_x\} + 1 + \lambda_z = 1\{x \geq 0\} + \lambda_z \\ &\Rightarrow \Delta_z - \lambda_z = z = 1\{x \geq 0\}. \end{aligned}$$

$$\begin{aligned}
& \langle \Delta_{\phi z} \rangle_0 + \langle \Delta_{\phi z} \rangle_1 \\
&= \text{Eval}_{\lambda_x, -\phi}^{\leq}(0, \kappa'_0, \Delta_x)_0 + \langle \phi \rangle_0 + \langle \lambda_{\phi z} \rangle_0 \\
&\quad + \text{Eval}_{\lambda_x, -\phi}^{\leq}(1, \kappa'_1, \Delta_x)_1 + \langle \phi \rangle_1 + \langle \lambda_{\phi z} \rangle_1 \\
&= -\phi \{x + \lambda_x < \lambda_x\} + \phi + \lambda_{\phi z} = \phi \{x \geq 0\} + \lambda_{\phi z} \\
&\Rightarrow \Delta_{\phi z} - \lambda_{\phi z} = \phi z = \phi \{x \geq 0\}. \quad \square
\end{aligned}$$

5. The Design of FLAME

Building on the components introduced in Sec. 4, we construct the FLAME scheme. Conceptually, FLAME comprises two key phases: client-side biometric template pre-processing and outsourcing and server-side secure authentication. Consistency check $\prod_{\text{MACCheck}}$ is executed in each component of the secure authentication phase to ensure correctness against malicious adversaries.

5.1. Client-side Pre-processing and Outsourcing

Existing PPBA studies primarily utilize the Euclidean distance or the Cosine distance to measure similarity. Below, we formally define these two metrics for two vectors $X = [X[i]]_{i=1}^n$ and $Y = [Y[i]]_{i=1}^n$. The Cosine distance between X and Y is formulated as

$$\text{CosD}(X, Y) = \frac{\sum_{i=1}^n X[i] \cdot Y[i]}{\sqrt{\sum_{i=1}^n X[i]^2} \cdot \sqrt{\sum_{i=1}^n Y[i]^2}}. \quad (1)$$

The Euclidean distance is defined as $\text{EucD}(X, Y) = \sum_{i=1}^n (X[i] - Y[i])^2$. From Eq. 1, it is evident that computing Cosine distance inherently involves expensive non-linear operations, namely division and square root. To mitigate this issue, Cheng *et al.* [11] proposed an MPC-friendly reformulation by integrating similarity score computation with authentication matching. Specifically, the authentication condition $1\{\text{CosD}(X, Y) \geq \tau\}$ (where τ is the public authentication threshold in plaintext) is transformed into $1\{\text{sign}(X \otimes Y) \wedge \text{sign}(\frac{1}{\tau^2}(X \otimes Y)^2 - (X \otimes X)(Y \otimes Y))\}$, where $\wedge$ denotes the logical AND operation, and $\text{sign}(x)$ is equivalent to the comparison operation that outputs 1 if $x \geq 0$ and 0 otherwise. This transformation effectively eliminates division and square root operations but introduces *two comparison and three inner product operations* per authentication. In contrast, Euclidean distance-based authentication merely involves *one comparison and one inner product operations*.

FLAME aims to enable service-selectable similarity metrics while maintaining server-side compatibility. That means a unified authentication procedure on the server side applies regardless of whether Euclidean or Cosine distance is used. However, this presents two key challenges:

- *Challenge 1:* Eliminating division and square root operations from Cosine distance to align with Euclidean distance computation.
- *Challenge 2:* Bridging the semantic gap between the two metrics—Cosine distance increases with similarity, whereas Euclidean distance decreases with similarity.

To address these challenges and achieve an efficient yet flexible authentication procedure, the client pre-processes the DNN-extracted biometric template $T = [T[1], T[2], \dots, T[n]]$ according to the similarity measure $m \in \{0, 1\}$. Here, $m = 0$ corresponds to Cosine distance, and Euclidean distance otherwise. For $m = 1$, the biometric template T is normalized to a unit-length vector, yielding $\hat{T} = [\frac{T[1]}{\|T\|}, \frac{T[2]}{\|T\|}, \dots, \frac{T[n]}{\|T\|}, 0]$, where the L^2 norm $\|T\| = \sqrt{\sum_{i=1}^n T[i]^2}$. Since the transformation guarantees $\|\hat{T}\| = \sqrt{\sum_{i=1}^n \hat{T}[i]^2} = 1$, explicit division and square root computations can be omitted during similarity computations (resolving **Challenge 1**). To reduce the impact of pre-processing on authentication accuracy while achieving storage compression, we adopt the method proposed in [15], which compresses each floating-point vector element into an 8-bit integer using min-max normalization. Conversely, for $m = 1$, the biometric template T is expanded by introducing an extra dimension, resulting in $\hat{T} = [T[1], T[2], \dots, T[n], -\frac{1}{2}(\sum_{i=1}^n T[i]^2)]$ for registration phase and $\hat{T} = [T[1], T[2], \dots, T[n], 1]$ for authentication phase. This expansion bridges the semantic gap between the two metrics, making the equivalent Euclidean distance (i.e., computed by the inner product directly) increase with similarity (resolving **Challenge 2**). The correctness is demonstrated below.

Given the processed fresh biometric template $\hat{T}$, and two stored reference biometric templates $\hat{D}_1$ and $\hat{D}_2$, their respective inner products with $\hat{T}$ are computed as:

$$\begin{aligned}
\hat{T} \otimes \hat{D}_j &= \sum_{i=1}^{n+1} \hat{T}[i] \cdot \hat{D}_j[i] \\
&= \sum_{i=1}^n \hat{T}[i] \cdot \hat{D}_j[i] - \frac{1}{2} \sum_{i=1}^n \hat{D}_j^2[i], \forall j \in \{1, 2\}.
\end{aligned}$$

The squared Euclidean distance difference between $\hat{T}$ and the two stored templates simplifies to:

$$\begin{aligned}
& \text{EucD}(\hat{T}, \hat{D}_1) - \text{EucD}(\hat{T}, \hat{D}_2) \\
&= \sum_{i=1}^n \hat{D}_1[i]^2 - 2 \sum_{i=1}^n \hat{T}[i] \cdot \hat{D}_1[i] - \sum_{i=1}^n \hat{D}_2[i]^2 + 2 \sum_{i=1}^n \hat{T}[i] \cdot \hat{D}_2[i] \\
&= 2(\sum_{i=1}^n \hat{T}[i] \cdot \hat{D}_2[i] - \frac{1}{2} \hat{D}_2[i]^2) - 2(\sum_{i=1}^n \hat{T}[i] \cdot \hat{D}_1[i] - \frac{1}{2} \hat{D}_1[i]^2) \\
&= 2(\hat{T} \otimes \hat{D}_2 - \hat{T} \otimes \hat{D}_1).
\end{aligned}$$

Thus, $\text{EucD}(\hat{T}, \hat{D}_1) \leq \text{EucD}(\hat{T}, \hat{D}_2) \iff \hat{T} \otimes \hat{D}_1 \geq \hat{T} \otimes \hat{D}_2$ is learned. This confirms that the inner product serves as an inverse surrogate for the Euclidean distance. This pre-processing guarantees a unified authentication procedure on the server side for different similarity metrics.

$\mathcal{C}$ then distributes the processed template $\hat{T} = [T[1], T[2], \dots, T[n], T[n+1]]$ between the two servers $\mathcal{P}_0$ and $\mathcal{P}_1$ with arithmetic secret sharing. Specifically, $\mathcal{C}$ selects a random vector $R \in \mathbb{Z}_{2^t+s}^{n+1}$, sending $\langle \hat{T} \rangle_0 = R$ to $\mathcal{P}_0$ and $\langle \hat{T} \rangle_1 = \hat{T} - R$ to $\mathcal{P}_1$. This entire dimensionality extension process is completed within a few microseconds and its cost

Protocol Π_{Initial}

Input: For any inputs $\hat{\mathbf{D}}$ with identity index I and ϕ , $\mathcal{P}_\theta$ hold $\langle \hat{\mathbf{D}} \rangle_\theta$ and $\langle \phi \rangle_\theta$, here $\langle \hat{\mathbf{D}} \rangle_\theta = [\langle I \rangle_\theta, \langle \hat{\mathbf{D}} \rangle_\theta]$.

- 1) Call $\mathcal{F}_{\text{rand}}$ to learn the shared $\langle \lambda_{\hat{\mathbf{D}}} \rangle, \langle \lambda_\phi \rangle, \langle \lambda_{\phi \hat{\mathbf{D}}} \rangle, \langle \lambda_{\hat{T}} \rangle$.
 - 2) Call $\mathcal{F}_{\text{triple}}$ to learn the shared multiplication triples $(\langle \mathbf{A} \rangle, \langle \mathbf{B} \rangle, \langle \mathbf{C} \rangle)$.
 - 3) Call $\Pi_{\text{OptSS}}(\langle \hat{\mathbf{D}} \rangle)$ and $\Pi_{\text{OptSS}}(\langle \phi \rangle)$ to learn $(\Delta_{\hat{\mathbf{D}}}, \langle \lambda_{\hat{\mathbf{D}}} \rangle_\theta)$ and $(\Delta_\phi, \langle \lambda_\phi \rangle_\theta)$.
 - 4) $\forall \theta \in \{0, 1\}$, $\mathcal{P}_\theta$ locally computes $\langle \delta_\phi \rangle_\theta = \langle \mathbf{A} \rangle_\theta - \langle \lambda_\phi \rangle_\theta$ and $\langle \delta_{\hat{\mathbf{D}}} \rangle_\theta = \langle \mathbf{B} \rangle_\theta - \langle \lambda_{\hat{\mathbf{D}}} \rangle_\theta$.
 - 5) Open $\langle \delta_\phi \rangle, \langle \delta_{\hat{\mathbf{D}}} \rangle$ to get the clear $\delta_\phi, \delta_{\hat{\mathbf{D}}}$.
 - 6) $\mathcal{P}_\theta$ locally computes $\langle \Delta_{\phi \hat{\mathbf{D}}} \rangle = (\Delta_\phi + \delta_\phi)(\Delta_{\hat{\mathbf{D}}} + \delta_{\hat{\mathbf{D}}}) - \langle \mathbf{A} \rangle(\Delta_{\hat{\mathbf{D}}} + \delta_{\hat{\mathbf{D}}}) - (\Delta_\phi + \delta_\phi)\langle \mathbf{B} \rangle + \langle \mathbf{C} \rangle + \langle \lambda_{\phi \hat{\mathbf{D}}} \rangle$.
 - 7) Open $\langle \Delta_{\phi \hat{\mathbf{D}}} \rangle$ to get the clear $\Delta_{\phi \hat{\mathbf{D}}} = \phi \hat{\mathbf{D}} + \lambda_{\phi \hat{\mathbf{D}}}$.
- Output:** $\mathcal{P}_\theta$ splits $(\Delta_{\hat{\mathbf{D}}}, \langle \lambda_{\hat{\mathbf{D}}} \rangle_\theta)$ and $(\Delta_{\phi \hat{\mathbf{D}}}, \langle \lambda_{\phi \hat{\mathbf{D}}} \rangle_\theta)$ to learn $(\Delta_{\hat{\mathbf{D}}}, \langle \lambda_{\hat{\mathbf{D}}} \rangle_\theta)$ and $(\Delta_I, \langle \lambda_I \rangle_\theta)$ and $(\Delta_{\phi \hat{\mathbf{D}}}, \langle \lambda_{\phi \hat{\mathbf{D}}} \rangle_\theta)$ and $(\Delta_{\phi I}, \langle \lambda_{\phi I} \rangle_\theta)$, as well as $(\Delta_\phi, \langle \lambda_\phi \rangle_\theta)$, and $(\lambda_{\hat{T}})_\theta$.

Figure 6. The initialization of our FLAME.

Algorithm 1 Secure Authentication Phase in FLAME.

- Input:** $\mathcal{P}_0$ and $\mathcal{P}_1$ receive shared $[\langle \hat{T} \rangle, \langle I_C \rangle]$.
- Output:** $\mathcal{P}_0$ and $\mathcal{P}_1$ output $(\Delta_\eta, \langle \lambda_\eta \rangle_\theta)$ to $\mathcal{S}$.
- 1: Call $\Pi_{\text{OptSS}}([\langle \hat{T} \rangle, \langle I_C \rangle])$ to learn $[\hat{T}] = (\Delta_{\hat{T}}, \langle \lambda_{\hat{T}} \rangle)$ and $[I_C] = (\Delta_{I_C}, \langle \lambda_{I_C} \rangle)$.
 - 2: # **Similarity score computation:**
 - 3: **for** $i = 1 \rightarrow m$ **do**
 - 4: $[\mathcal{S}[i]], [\phi \mathcal{S}[i]] = \text{SecIP}([\hat{\mathbf{D}}[i]], [\phi \hat{\mathbf{D}}[i]], [\hat{T}])$.
 - 5: **end for**
 - 6: # **Top-1 similarity retrieval:**
 - 7: $[\tilde{\mathcal{S}}] = [\mathcal{S}[1]], [\tilde{I}] = [I[1]]$.
 - 8: $[\phi \tilde{\mathcal{S}}] = [\phi \mathcal{S}[1]], [\phi \tilde{I}] = [\phi I[1]]$.
 - 9: **for** $i = 2 \rightarrow m$ **do**
 - 10: $[\mathbf{b}], [\phi \mathbf{b}] = \text{SecCMP}([\tilde{\mathcal{S}}] - [\mathcal{S}[i]])$.
 - 11: $[\tilde{\mathcal{S}}] = [\mathcal{S}[i]] + \Pi_{\text{mult}}([\mathbf{b}], [\tilde{\mathcal{S}}] - [\mathcal{S}[i]])$ and $[\phi \tilde{\mathcal{S}}] = [\phi \mathcal{S}[i]] + \Pi_{\text{mult}}([\mathbf{b}], [\phi \tilde{\mathcal{S}}] - [\phi \mathcal{S}[i]])$.
 - 12: $[\tilde{I}] = [I[i]] + \Pi_{\text{mult}}([\mathbf{b}], [\tilde{I}] - [I[i]])$ and $[\phi \tilde{I}] = [\phi I[i]] + \Pi_{\text{mult}}([\mathbf{b}], [\phi \tilde{I}] - [\phi I[i]])$.
 - 13: **end for**
 - 14: # **Identity Match:**
 - 15: $[\eta] = [\tilde{I}] - [I_C]$ and $[\phi \eta] = [\phi \tilde{I}] - [\phi I_C]$.
 - 16: The servers run Π_{MACCheck} to check the each secure computation result. If the check fails, abort.
 - 17: Return $[\eta]$ to the service provider for further decision.

is effectively amortized over subsequent authentication.

5.2. Server-side Secure Authentication

In FLAME, we assume m clients have already completed registration, and the servers have pre-stored a large number of shared reference biometric templates $\langle \hat{\mathbf{D}} \rangle$ with identity index $\langle I \rangle$ ($\forall I \in [1, m]$), that is $\langle \hat{\mathbf{D}} \rangle_\theta = [\langle I \rangle_\theta, \langle \hat{\mathbf{D}} \rangle_\theta]$. Given the MAC key $\langle \phi \rangle$ from the trusted $\mathcal{S}$, the two

servers $\mathcal{P}_0$ and $\mathcal{P}_1$ initialize to compute $(\Delta_{\hat{\mathbf{D}}}, \langle \lambda_{\hat{\mathbf{D}}} \rangle)$ and $(\Delta_{\phi \hat{\mathbf{D}}}, \langle \lambda_{\phi \hat{\mathbf{D}}} \rangle)$, as shown in Figure 6. Upon receiving the shared fresh biometric template $\langle \hat{T} \rangle$ from the client $\mathcal{C}$ (i.e., the identity index is I_C), $\mathcal{P}_0$ and $\mathcal{P}_1$ jointly execute the authentication phase, shown in Algorithm 1. In this phase, the servers securely compute the similarity scores between $\langle \hat{T} \rangle$ and the stored biometric database $\langle \hat{\mathbf{D}} \rangle$ without revealing any sensitive information. The computation leverages Π_{SecIP} to obtain a similarity score vector $[\mathcal{S}[i]], [\phi \mathcal{S}[i]]$, where each entry corresponds to the similarity between $\langle \hat{T} \rangle$ and the i -th stored biometric template. Finally, the servers collaboratively determine the most similar entry (i.e., top-1 nearest neighbor) using a secure maximum-finding protocol, yielding the shared authentication result $[\eta], [\phi \eta]$. At the end of the online secure authentication phase, the servers run Π_{MACCheck} to detect malicious behaviors (line 16 in Algorithm 1). If the consistency check fails, the servers immediately abort and output $\perp$, thereby ensuring the correctness and integrity of the authentication result. The service provider reconstructs $\eta = \Delta_\eta - \langle \lambda_\eta \rangle_0 - \langle \lambda_\eta \rangle_1$ and grants the client access to the requested service if $\eta = 0$; otherwise, the client's access request is rejected.

6. Theoretical Analysis

Security analysis. FLAME's pipeline integrates a variety of maliciously-secure protocols for different computations, and the input and output of each computation in the secret-sharing domain. Using the sequential composition theorem [17], we deduce the overall security of FLAME as stated in Theorem 3. Due to space constraints, we present a sketch of the security proof of the FLAME scheme.

Theorem 3. *FLAME's biometric authentication scheme Π^{FLAME} correctly and securely realizes the ideal functionality $\mathcal{F}^{\text{FLAME}}$ in the presence of one malicious adversary $\mathcal{A}$ in the $(\Pi_{\text{add}}, \Pi_{\text{mult}}, \Pi_{\text{SecIP}}, \Pi_{\text{SecCMP}})$ -hybrid model.*

Proof Sketch. The malicious security of FLAME is proven according to the cryptographic standard outlined in **Definition 1**. We begin by proving the security of supporting protocols $(\Pi_{\text{add}}, \Pi_{\text{mult}}, \Pi_{\text{SecIP}}, \Pi_{\text{SecCMP}})$ against malicious adversaries. Concretely, based on SS-variants, the view of the adversary $\mathcal{A}$ in the real-world execution, denoted as $\text{View}_{\mathcal{A}}^{\Pi}$, is computationally indistinguishable from the simulated view $\text{Sim}_{\mathcal{A}}^{\mathcal{F}}$ generated by the ideal-world simulator Sim . To further strengthen malicious security, we integrate authentication mechanisms into these supporting protocols. Rather than relying on computationally expensive asymmetric commitment schemes, we employ a symmetric-key-based MAC to ensure integrity. The MAC authenticates the results for each secure computation, allowing any unauthorized deviations from the secure computation to be detected. If any deviations from the expected execution are detected, Π^{FLAME} aborts immediately to prevent adversarial influence. Therefore, each supporting protocol is secure and correct against a PPT malicious adversary $\mathcal{A}$. Finally, according to the *universal composability* theory [10], we claim that our

TABLE 3. ONLINE THEORETICAL PERFORMANCE COMPARISON OF FLAME AND NOMADIC [11]. FOR A FAIR COMPARISON OF THE OVERALL SCHEMES, WE ALIGN FLAME WITH THE OVER-THE-THRESHOLD NOMADIC SCHEME FOR ONE-TIME PPBA.

		Computation	Communication
SecIP	Semi-honest	$n \prod_{\text{mult}}$	nl
	Nomadic	$2n \prod_{\text{mult}}$	$2n(l + s)$
	Ours	$2 \prod_{\text{mult}}$	$2(l + s)$
SecCMP	Semi-honest	$\text{Eval}^<$	l
	Nomadic	$4\text{Eval}^<$	$4(l + s)$
	Ours	$\text{Eval}^<$	$2(l + s)$
FLAME	Semi-honest	$n \prod_{\text{mult}} + \text{Eval}^<$	$(n + 1)l$
	Nomadic	$2n \prod_{\text{mult}} + 4\text{Eval}^<$	$(2n + 4)(l + s)$
	Ours	$4 \prod_{\text{mult}} + \text{Eval}^<$	$6(l + s)$

protocol $\prod^{\text{FLAME}}$ correctly and securely realizes the ideal functionality $\mathcal{F}^{\text{FLAME}}$ in the malicious environments. This completes the proof of Theorem 3. $\square$

Efficiency analysis. We analyze the online efficiency of FLAME and its key protocols (SecIP and SecCMP) against state-of-the-art (SotA) schemes in Table 3. There are three parameters involved in this analysis: (1) l, s is the concrete parameters for SPDZ_{2^l} , (2) n is the biometric template length. Note that addition and subtraction operations are considered free in this context, as it requires no interaction and can be performed locally. SecIP, SecCMP, and FLAME show powerful improvements in online computation, with the majority of costs incurred during the offline phase for generating cryptographic primitives and randomness, such as multiplication triples and keys. Additionally, they achieve lower online communication overhead compared to SotA.

7. Experimental Evaluation

Testbed and parameters. We implement a prototype of FLAME in C++. All experiments are conducted on two simulated servers, each equipped with an Apple M2 processor, 16 GB of RAM, running macOS Sonoma 14.4. We use the `tc` command to simulate LAN (RTT: 0.1 ms, 1 Gbps), MAN (RTT: 6 ms, 100 Mbps), and WAN (RTT: 80 ms, 40 Mbps) networks. For fairness, we implement the evaluations for SotA scheme Nomadic [11] and our FLAME in the MP-SPDZ framework¹ [22]. Since Nomadic does not release its source code, we re-implement it based on the techniques of SPDZ_{2^l} [14]. Also, the FuncSS-based comparison functions are implemented based on the library², which we enhance by incorporating the generation and evaluation algorithms from [8]. We set the parameters $l = s = 64$ for the SPDZ_{2^l} secret-sharing scheme.

Datasets and models. We evaluate FLAME on two real-world facial datasets: LFW³ and VidTIMIT⁴. Each biometric template is represented as a 512-dimensional feature vector ($n = 512$), extracted via FaceNet and ArcFace models,

because 512 dimensions strike a practical balance between recognition accuracy and computational efficiency, and are the standard output size for these widely adopted models. Our analysis identified 62 individuals from LFW and 43 from VidTIMIT who meet the registration criteria, each with 20 valid samples. All results are averaged over 10 trials, and the reported communication is amortized per party.

7.1. Micro-benchmarks on Supporting Protocols

We benchmark the offline and online costs of the building blocks for Nomadic and FLAME, including:

- **Benchmark 1.** Secure inner product of vector length 2^{10} with varying input size.
- **Benchmark 2.** Secure inner product of input size 2^{10} with varying vector lengths.
- **Benchmark 3.** Secure comparison with varying batch size.

Benchmark 1. Tables 4 and 5 report the runtime and communication costs of our SecIP and Nomadic under varying input sizes, for the offline and online phases, respectively. In the offline phase (Table 4), we observe the runtime and communication costs remain nearly identical, with factor differences mostly around $1.00\times$ across all network settings. In the online phase (Table 5), SecIP consistently outperforms Nomadic across all metrics. We achieve substantial speedups in online runtime, with improvements ranging from $3.35\times$ to $6.12\times$ across different settings. The communication cost in SecIP is reduced by more than $1000\times$ (approximately independent of the vector length 2^{10}) compared to Nomadic, which aligns with our theoretical analysis that the online communication is independent of the vector length.

Benchmark 2. Figs. 4 and 5 report the runtime and communication costs of our SecIP and Nomadic under varying vector lengths, for the offline and online phases, respectively. The results are consistent with **Benchmark 1**: Nomadic performs similarly to SecIP in the offline phase, with runtime and communication costs being identical for both protocols. In the online phase, as vector size increases, Nomadic exhibits a corresponding increase in communication overhead, while SecIP’s online communication cost remains constant. This further supports the finding that the communication cost in SecIP is independent of the vector length, saving about $2^{10}\times$ to $2^{13}\times$ in communication cost compared to Nomadic.

Benchmark 3. Tables 6 (for the offline phase) and 7 (for the online phase) present the performance of the non-linear comparison protocol SecCMP. In both the offline and online phases, SecCMP significantly outperforms Nomadic in terms of runtime and communication costs. In the offline phase, runtime improves by up to $132.66\times$ and communication costs are reduced by up to $460.22\times$. In the online phase, runtime improvements reach $33.81\times$ in the LAN setting and communication costs are reduced by up to $31.10\times$. These improvements remain consistent across different vector sizes, demonstrating SecCMP’s efficiency.

1. <https://github.com/data61/mp-spdz>

2. <https://github.com/frankw2/libfss>

3. <https://www.kaggle.com/datasets/jessicali9530/lfw-dataset>

4. <https://conradsanderson.id.au/VidTIMIT/>

TABLE 4. OFFLINE TIME (IN MINUTES) AND COMMUNICATION (COMM., IN GIGA-BYTES) OF SecIP EVALUATED ON VECTORS OF LENGTH 2^{10} .

Size	LAN Time			MAN Time			WAN Time			Communication		
	Nomadix	Ours	Factor	Nomadix	Ours	Factor	Nomadix	Ours	Factor	Nomadix	Ours	Factor
2^8	0.66	0.64	$1.03\times$	7.64	7.56	$1.01\times$	12.99	13.07	$0.99\times$	2.34	2.34	$1.00\times$
2^{10}	2.75	2.79	$0.99\times$	26.55	26.45	$1.00\times$	54.72	53.33	$1.03\times$	9.27	9.27	$1.00\times$
2^{12}	10.77	10.80	$1.00\times$	106.35	103.56	$1.03\times$	201.41	200.16	$1.01\times$	37.02	37.02	$1.00\times$
2^{14}	41.68	41.29	$1.01\times$	437.80	446.39	$0.98\times$	804.37	799.81	$1.01\times$	148.00	148.00	$1.00\times$

TABLE 5. ONLINE TIME (IN SECONDS) AND COMMUNICATION (COMM., IN MEGA-BYTES) OF SecIP EVALUATED ON VECTORS OF LENGTH 2^{10} .

Size	LAN Time			MAN Time			WAN Time			Communication		
	Nomadix	Ours	Factor	Nomadix	Ours	Factor	Nomadix	Ours	Factor	Nomadix	Ours	Factor
2^8	0.39	0.11	$3.64\times$	0.96	0.23	$4.17\times$	2.22	0.58	$3.83\times$	4.20	0.004	$1050.00\times$
2^{10}	1.47	0.43	$3.45\times$	3.25	0.68	$4.78\times$	4.95	1.04	$4.76\times$	16.79	0.016	$1049.37\times$
2^{12}	6.09	1.73	$3.52\times$	15.17	2.48	$6.12\times$	19.25	3.97	$4.85\times$	67.12	0.063	$1065.39\times$
2^{14}	23.60	7.04	$3.35\times$	65.93	15.65	$4.21\times$	74.73	20.23	$3.69\times$	268.45	0.25	$1073.80\times$

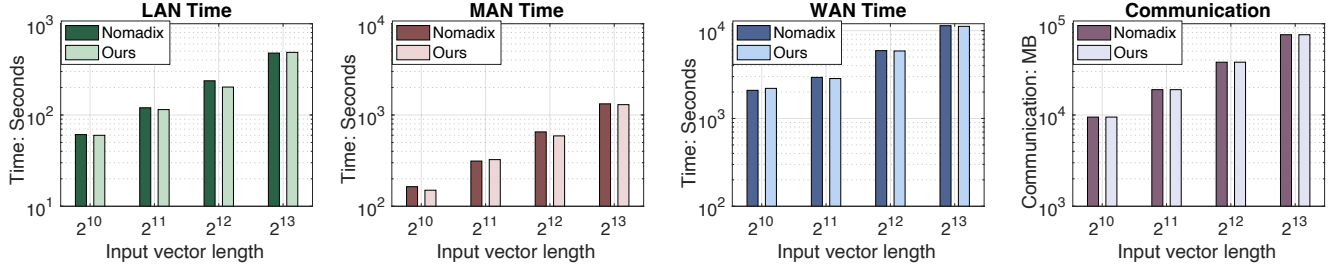
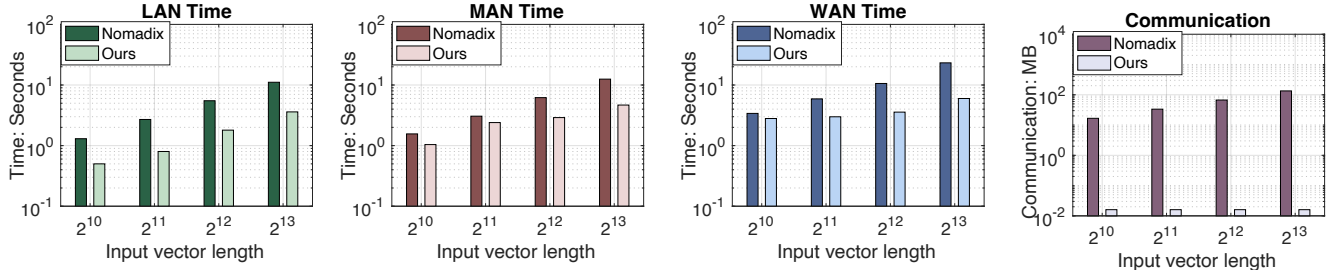
Figure 7. Offline performance comparison of SecIP protocol when the size of the dataset is 2^{10} .Figure 8. Online performance comparison of SecIP protocol when the size of the dataset is 2^{10} .

TABLE 6. OFFLINE TIME (IN SECONDS) AND COMMUNICATION (COMM., IN MEGA-BYTES) OF SecCMP.

Size	LAN Time			MAN Time			WAN Time			Communication		
	Nomadix	Ours	Factor	Nomadix	Ours	Factor	Nomadix	Ours	Factor	Nomadix	Ours	Factor
2^8	1.43	0.03	$47.66\times$	3.56	0.12	$29.66\times$	107.31	4.36	$24.61\times$	221.88	0.69	$321.56\times$
2^{10}	5.54	0.06	$92.33\times$	17.28	0.33	$52.36\times$	339.56	6.68	$50.83\times$	861.12	2.63	$327.42\times$
2^{12}	21.60	0.19	$113.68\times$	128.89	1.48	$87.08\times$	577.13	8.24	$70.04\times$	3424.10	7.44	$460.22\times$
2^{14}	86.23	0.65	$132.66\times$	278.53	3.12	$89.27\times$	2068.50	25.47	$81.21\times$	13668.95	38.41	$355.86\times$

7.2. Evaluations on FLAME Scheme

The results presented in Table 8 showcase the online performance of FLAME on two different facial biometric datasets: LFW and VidTIMIT. For LFW, FLAME achieves a remarkable factor of $2.82\times$ faster runtime in LAN, $4.36\times$ faster in MAN, and $6.58\times$ faster in WAN. The communication cost is drastically reduced by a factor

of $97.61\times$, highlighting FLAME's efficiency. Similarly, for VidTIMIT, FLAME shows substantial improvements, with runtime speedup factors of $2.72\times \sim 8.51\times$ in LAN, MAN, and WAN. The communication cost is reduced by $110.13\times$. These enhancements stem from our online-offline-paradigm optimized protocols with a one-round-communication design, providing strong evidence of FLAME's potential for real-world biometric authentication applications.

TABLE 7. ONLINE TIME (IN SECONDS) AND COMMUNICATION (COMM., IN MEGA-BYTES) OF **SecCMP**.

Size	LAN Time			MAN Time			WAN Time			Communication (MB)		
	Nomadix	Ours	Factor	Nomadix	Ours	Factor	Nomadix	Ours	Factor	Nomadix	Ours	Factor
2^8	0.16	0.023	$6.96\times$	0.36	0.05	$7.20\times$	4.30	0.11	$39.09\times$	0.52	0.02	$28.88\times$
2^{10}	0.74	0.053	$13.96\times$	1.45	0.11	$13.37\times$	12.22	0.26	$47.00\times$	2.09	0.07	$29.85\times$
2^{12}	2.93	0.18	$16.28\times$	5.77	0.32	$18.03\times$	15.16	0.54	$28.07\times$	8.71	0.28	$31.10\times$
2^{14}	7.44	0.22	$33.81\times$	13.86	0.34	$40.76\times$	37.51	0.67	$55.98\times$	33.84	1.13	$29.94\times$

TABLE 8. ONLINE TIME (IN SECONDS) AND COMMUNICATION (COMM., IN MEGA-BYTES) OF **FLAME** ON DIFFERENT BIOMETRIC DATASETS.

Dataset	LAN Time			MAN Time			WAN Time			Communication		
	Nomadix	Ours	Factor	Nomadix	Ours	Factor	Nomadix	Ours	Factor	Nomadix	Ours	Factor
LFW	1.78	0.63	$2.82\times$	3.71	0.85	$4.36\times$	12.78	1.94	$6.58\times$	12.69	0.13	$97.61\times$
VidTIMIT	1.28	0.47	$2.72\times$	2.83	0.68	$4.16\times$	9.71	1.14	$8.51\times$	8.81	0.08	$110.13\times$

TABLE 9. ACCURACY OF **FLAME** ON DIFFERENT DATASETS AND MODELS WITH VARIOUS CHOICES OF $k = 1/5/10$.

Datasets	Metrics	FaceNet			ArcFace		
LFW ($m = 1240$)	Cosine	99.27%	99.18%	98.99%	99.19%	98.66%	98.17%
	Euclidean	99.27%	99.11%	98.93%	98.87%	97.27%	96.36%
VidTIMIT ($m = 860$)	Cosine	97.91%	96.14%	95.76%	97.09%	89.91%	85.53%
	Euclidean	97.91%	95.98%	95.55%	96.98%	88.05%	81.12%

7.3. Accuracy

The accuracy results of **FLAME** with different similarity metrics are summarized in Table 9. Top- k accuracy is calculated as the number of samples whose true label appears in the top- k predicted labels, divided by the total number of samples evaluated. We can see that FaceNet, which is used for biometric feature extraction, consistently outperforms ArcFace in all cases ($k = 1, 5, 10$), achieving accuracy rates exceeding 95%. Considering both biometric authentication accuracy, we use FaceNet as the source of all real biometric features in the above experiments. Also, Cosine similarity performs better than Euclidean distance in facial authentication, providing valuable insights for future research on selecting appropriate biometric similarity metrics.

8. Conclusion and Future Work

In this paper, we present **FLAME**, a 2PC-based biometric authentication scheme that supports flexible similarity metrics, including cosine similarity and Euclidean distance. Through tailored protocols with an offline-online paradigm, **FLAME** ensures low-latency authentication while providing strong privacy and integrity guarantees against malicious adversaries. Finally, experimental results on real-world biometric datasets demonstrate the practicality and effectiveness of **FLAME**, highlighting its suitability for real-time, large-scale biometric authentication scenarios.

In future work, we aim to extend **FLAME** to support multi-modal biometric authentication, integrating multiple biometric modalities such as fingerprint, face, and audio recognition to further enhance accuracy, robustness, and

security. We also plan to explore the design of an end-to-end PPBA schemes, including privacy-preserving de-duplication during enrollment, to further improve practicality and address real-world deployment challenges.

Acknowledgments

This research is supported by the National Research Foundation, Singapore and Infocomm Media Development Authority under its Trust Tech Funding Initiative (DTC-T2FI-CFP-0002). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore and Infocomm Media Development Authority.

References

- [1] Al-Mannai, K., Bentafat, E., et al.: Secure biometric verification in the presence of malicious adversaries. *IEEE Access* (2024)
- [2] Bai, J., Song, X., Zhang, X., Wang, Q., Cui, S., Chang, E.C., Russell, G.: Mostree: malicious secure private decision tree evaluation with sublinear communication. In: *Proceedings of the 39th Annual Computer Security Applications Conference*. pp. 799–813 (2023)
- [3] Barni, M., Droandi, G., Lazzaretto, R., Pignata, T.: Semba: secure multi-biometric authentication. *IET Biometrics* **8**(6), 411–421 (2019)
- [4] Bassit, A., Hahn, F., Peeters, J., Kevenaar, T., Veldhuis, R.N., Peter, A.: Biometric verification secure against malicious adversaries. *arXiv preprint arXiv:2101.10631* (2021)
- [5] Bassit, A., Hahn, F., Peeters, J., et al.: Fast and accurate likelihood ratio-based biometric verification secure against malicious adversaries. *IEEE TIFS* **16**, 5045–5060 (2021)

- [6] Ben-Efraim, A., Nielsen, M., et al.: Turbospeedz: Double your online SPDZ! improving SPDZ using function dependent preprocessing. In: ACNS. pp. 530–549. Springer (2019)
- [7] Blanton, M., Murphy, D.: Privacy preserving biometric authentication for fingerprints and beyond. In: Proceedings of the Fourteenth ACM Conference on Data and Application Security and Privacy. pp. 367–378 (2024)
- [8] Boyle, E., Chandran, N., et al.: Function secret sharing for mixed-mode and fixed-point secure computation. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 871–900. Springer (2021)
- [9] Boyle, E., Gilboa, N., Ishai, Y.: Function secret sharing. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 337–367. Springer (2015)
- [10] Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: Proceedings 42nd IEEE Symposium on Foundations of Computer Science. pp. 136–145. IEEE (2001)
- [11] Cheng, N., Önen, M., Mitrokovska, A., et al.: Nomadic: Normalising maliciously-secure distance with cosine similarity for two-party biometric authentication. In: AsiaCCS. pp. 257–273 (2024)
- [12] Chida, K., Hamada, K., Ikarashi, D., et al.: Fast large-scale honest-majority MPC for malicious adversaries. *Journal of Cryptology* **36**(3), 15 (2023)
- [13] Cramer, R., Damgård, I., Escudero, D., et al.: $\text{Spd}\mathbb{Z}_{2^k}$: efficient MPC mod 2^k for dishonest majority. In: Annual International Cryptology Conference. pp. 769–798. Springer (2018)
- [14] Damgård, I., Escudero, D., Frederiksen, T., et al.: New primitives for actively-secure MPC over rings with applications to private machine learning. In: S&P. pp. 1102–1120. IEEE (2019)
- [15] Engelsma, J.J., Cao, K., Jain, A.K.: Learning a fixed-length fingerprint representation. *TPAMI* **43**(6), 1981–1997 (2021)
- [16] Fălămaș, D.E., Marton, K., Suci, A.: Assessment of two privacy preserving authentication methods using secure multiparty computation based on secret sharing. *Symmetry* **13**(5), 894 (2021)
- [17] Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game, or a completeness theorem for protocols with honest majority. In: Proceedings of the nineteenth annual ACM symposium on Theory of computing. pp. 218–229 (1987)
- [18] Hao, M., Li, H., Chen, H., et al.: FastSecNet: An efficient cryptographic framework for private neural network inference. *IEEE TIFS* **18**, 2569–2582 (2023)
- [19] Huang, H., Wang, L.: Efficient privacy-preserving face identification protocol. *IEEE TSC* **16**(4), 2632–2641 (2023)
- [20] Im, J.H., Jeon, S.Y., Lee, M.K.: Practical privacy-preserving face authentication for smartphones secure against malicious clients. *IEEE Transactions on Information Forensics and Security* **15**, 2386–2401 (2020)
- [21] Järvinen, K., Leppäkoski, H., et al.: PILOT: Practical privacy-preserving indoor localization using outsourcing. In: EuroS&P. pp. 448–463. IEEE (2019)
- [22] Keller, M.: Mp-spdz: A versatile framework for multi-party computation. In: CCS. pp. 1575–1590 (2020)
- [23] Makri, E., Rotaru, D., et al.: Rabbit: Efficient comparison for secure multi-party computation. In: FC. pp. 249–270. Springer (2021)
- [24] Nieminen, R., Järvinen, K.: Practical privacy-preserving indoor localization based on secure two-party computation. *IEEE TMC* **20**(9), 2877–2890 (2021)
- [25] Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 223–238. Springer (1999)
- [26] Pradel, G., Mitchell, C.: Privacy-preserving biometric matching using homomorphic encryption. In: TrustCom. pp. 494–505. IEEE (2021)
- [27] Saraswat, D., Ladhiya, K., et al.: PHBio: A pallier homomorphic biometric encryption scheme in healthcare 4.0 ecosystems. In: 2022 3rd International Conference on Intelligent Engineering and Management. pp. 306–312. IEEE (2022)
- [28] Shahreza, H.O., et al.: Hybrid protection of biometric templates by combining homomorphic encryption and cancelable biometrics. In: 2022 IEEE International Joint Conference on Biometrics. pp. 1–10. IEEE (2022)
- [29] Treiber, A., Nautsch, A., et al.: Privacy-preserving plda speaker verification using outsourced secure computation. *Speech Communication* **114**, 60–71 (2019)
- [30] Wang, F., Chen, Z., Pan, L., et al.: Cryptgraph: An efficient privacy-enhancing solution for accurate shortest path retrieval in cloud environments. In: AsiaCCS. pp. 1660–1674 (2024)
- [31] Wang, F., Ouyang, J., et al.: SEBioID: Secure and efficient biometric identification with two-party computation. In: International Conference on Applied Cryptography and Network Security. pp. 83–109. Springer (2025)
- [32] Wang, F., Zhang, L.Y., Pan, L., Hu, S., Doss, R.: Towards privacy-preserving neural architecture search. In: 2022 IEEE Symposium on Computers and Communications (ISCC). pp. 1–6. IEEE (2022)
- [33] Wei, F., Vijayakumar, P., Kumar, N., et al.: Privacy-preserving implicit authentication protocol using cosine similarity for internet of things. *IEEE Internet of Things Journal* **8**(7), 5599–5606 (2020)
- [34] Yang, X., Luo, Y., Xu, M., et al.: Privacy-preserving wifi fingerprint localization based on spatial linear correlation. In: International Conference on Wireless Algorithms, Systems, and Applications. pp. 401–412. Springer (2022)
- [35] Yuan, B., Yang, S., Zhang, Y., et al.: MD-ML: Super fast Privacy-Preserving machine learning for malicious security with a dishonest majority. In: USENIX Security. pp. 2227–2244 (2024)

Appendix

Security Proof. In this section, we first provide simulation-based security proofs for the supporting protocols. Then, we present the ideal-world definition of our proposed FLAME (see Figure 9) and give a hybrid argument proof for the security of the FLAME scheme.

Theorem 4. *In the $(\mathcal{F}_{\text{ABB}}, \prod_{\text{add}}, \prod_{\text{mult}})$ -hybrid model, the protocol $\prod_{\text{SecIP}}$ implements $\mathcal{F}_{\text{SecIP}}$ correctly and securely against malicious adversary.*

Proof. The proof of the correctness is established in Theorem 1. We construct an ideal-world simulator $\text{Sim}_{\text{SecIP}}$ as follows to show that the real $\prod_{\text{SecIP}}$ securely realizes the functionality $\mathcal{F}_{\text{SecIP}}$ in the presence of a malicious adversary $\mathcal{A}$.

- 1) $\text{Sim}_{\text{SecIP}}$ receives the public parameters l, s and the shares $\langle \lambda_X \rangle, \langle \lambda_{\phi_X} \rangle, \langle \lambda_Y \rangle \leftarrow \mathbb{Z}_{l+s}^n$ from the ideal functionality $\mathcal{F}_{\text{SecIP}}$.
- 2) $\text{Sim}_{\text{SecIP}}$ simulates the ideal functionalities $\mathcal{F}_{\text{rand}}$ and $\mathcal{F}_{\text{triple}}$. It randomly samples:

$$A'_1, A'_2, B', C'_1, C'_2 \leftarrow \mathbb{Z}_{l+s}^n, \quad \lambda'_z, \lambda'_{\phi_z} \leftarrow \mathbb{Z}_{l+s}^1,$$

and generates corresponding additive shares $\langle A'_1 \rangle, \langle A'_2 \rangle, \langle B' \rangle, \langle C'_1 \rangle, \langle C'_2 \rangle, \langle \lambda'_z \rangle, \langle \lambda'_{\phi_z} \rangle$, and sends these simulated shares to the malicious adversary.

- 3) When the malicious adversary computes the differences: $\langle \delta'_X \rangle = \langle A'_1 \rangle - \langle \lambda'_X \rangle, \langle \delta'_{\phi_X} \rangle = \langle A'_2 \rangle - \langle \lambda'_{\phi_X} \rangle, \langle \delta'_Y \rangle =$

$\langle B' \rangle - \langle \lambda'_Y \rangle$, and sends the shares of $\langle \delta'_X \rangle$, $\langle \delta'_{\phi X} \rangle$, and $\langle \delta'_Y \rangle$ to open. $\text{Sim}_{\text{SecIP}}$ sends the clear values of δ'_X , $\delta'_{\phi X}$, and δ'_Y to the adversary.

- 4) Upon receiving the adversary's optimized shares: $(\Delta_X, \langle \lambda_X \rangle)$, $(\Delta_{\phi X}, \langle \lambda_{\phi X} \rangle)$, $(\Delta_Y, \langle \lambda_Y \rangle)$, $\text{Sim}_{\text{SecIP}}$ computes: $\Delta'_z = \sum_{i=1}^n ((\Delta_X[i] + \delta'_X[i]) (\Delta_Y[i] + \delta'_Y[i]) - A'_1[i] (\Delta_Y[i] + \delta'_Y[i]) - (\Delta_X[i] + \delta'_X[i]) B'[i] + C'_1[i]) + \lambda'_z$ and $\Delta'_{\phi z} = \sum_{i=1}^n ((\Delta_{\phi X}[i] + \delta'_{\phi X}[i]) (\Delta_Y[i] + \delta'_Y[i]) - A'_2[i] (\Delta_Y[i] + \delta'_Y[i]) - (\Delta_{\phi X}[i] + \delta'_{\phi X}[i]) B'[i] + C'_2[i]) + \lambda'_{\phi z}$.
- 5) $\text{Sim}_{\text{SecIP}}$ opens Δ'_z and $\Delta'_{\phi z}$ and returns them to the adversary.
- 6) S_{SecIP} outputs all values the adversary would have seen in the real protocol, $\text{Sim}_{\text{SecIP}}$ verifies the consistency of the shares via MAC checks. If the MAC checks fail, $\text{Sim}_{\text{SecIP}}$ aborts and notifies $\mathcal{F}_{\text{SecIP}}$ to abort as well.

Since the simulator $\text{Sim}_{\text{SecIP}}$ only uses simulated values and the distributions of these shares are indistinguishable from those in the real protocol due to the information-theoretic security of secret sharing primitives, the view $\text{View}_{\text{SecIP}}$ of the adversary $\mathcal{A}$ in the simulation is indistinguishable from its view in the real execution. Moreover, any deviation from the protocol Π_{SecIP} is detected through Π_{MACCheck} and leads to an abort, just as in the real world. Therefore, the real protocol Π_{SecIP} securely realizes the ideal functionality $\mathcal{F}_{\text{SecIP}}$. $\square$

Theorem 5. *In the $(\mathcal{F}_{\text{ABB}}, \Pi_{\text{Gen}}, \Pi_{\text{Eval}})$ -hybrid model, the protocol Π_{SecCMP} implements $\mathcal{F}_{\text{SecCMP}}$ correctly and securely against malicious adversary.*

Proof. The proof of the correctness is established in Theorem 2. We construct an ideal-world simulator S_{SecCMP} to simulate the view $\text{View}_{\text{SecCMP}}$ of a malicious adversary $\mathcal{A}$ in Π_{SecCMP} and show that it securely realizes the ideal functionality $\mathcal{F}_{\text{SecCMP}}$.

- 1) $\text{Sim}_{\text{SecCMP}}$ receives the the public parameters l, s , input shares $\langle \lambda_x \rangle, \langle \phi \rangle \leftarrow \mathbb{Z}_{l+s}$ from the ideal functionality $\mathcal{F}_{\text{SecCMP}}$.
- 2) $\text{Sim}_{\text{SecCMP}}$ opens $a = \lambda_x$ and $\mathbf{b} = (1, \phi)$.
- 3) $\text{Sim}_{\text{SecCMP}}$ runs the (honest) $\text{Gen}_{a, -\mathbf{b}}^{\leq}$ algorithm to get (κ'_0, κ'_1) , sets $\kappa''_{\theta} = \kappa''_{\theta} \| \langle \mathbf{b} \rangle_{\theta}$, then sends κ''_{θ} to the adversary.
- 4) $\text{Sim}_{\text{SecCMP}}$ receives the invocations to $\mathcal{F}_{\text{share}}$ from the adversary $\mathcal{A}$, returns shares $\langle b'_0 \rangle, \langle b'_1 \rangle$ to $\mathcal{A}$.
- 5) $\text{Sim}_{\text{SecCMP}}$ receives the $\mathcal{F}_{\text{rand}}$ calls from $\mathcal{A}$, randomly generates corresponding additive shares $\langle \lambda'_z \rangle, \langle \lambda'_{\phi z} \rangle$, then sends these simulated shares to the adversary.
- 6) Upon receiving the adversary's optimized share $(\Delta_x, \langle \lambda_x \rangle)$, $\text{Sim}_{\text{SecCMP}}$ runs the honest evaluation algorithm: $(\langle \gamma'_0 \rangle, \langle \gamma'_1 \rangle) \leftarrow \text{Eval}_{a, -\mathbf{b}}^{\leq}(\theta, \kappa'_{\theta}, \Delta_x)$.
- 7) $\text{Sim}_{\text{SecCMP}}$ learns $\langle \Delta'_z \rangle = \langle \gamma'_0 \rangle + \langle b'_0 \rangle + \langle \lambda'_z \rangle$, $\langle \Delta'_{\phi z} \rangle = \langle \gamma'_1 \rangle + \langle b'_1 \rangle + \langle \lambda'_{\phi z} \rangle$.
- 8) $\text{Sim}_{\text{SecCMP}}$ opens Δ'_z and $\Delta'_{\phi z}$ to the adversary.
- 9) $\text{Sim}_{\text{SecCMP}}$ simulates Π_{MACCheck} using all the adversary's values. If the check fails, abort the execution with the adversary $\mathcal{A}$; otherwise, continue.

Since all values are either correctly computed using values from the ideal functionality or uniformly random from the adversary's perspective, and any deviation is caught by the MAC check, the simulator's $\text{Sim}_{\text{SecCMP}} = \{\kappa''_{\theta}, \langle \lambda'_z \rangle, \langle \lambda'_{\phi z} \rangle, \langle b'_0 \rangle, \langle b'_1 \rangle, \langle \gamma'_0 \rangle, \langle \gamma'_1 \rangle, \Delta'_z, \Delta'_{\phi z}\}$ is indistinguishable from that in the real view $\text{View}_{\text{SecCMP}} = \{\kappa_{\theta}, \langle \lambda_z \rangle, \langle \lambda_{\phi z} \rangle, \langle b_0 \rangle, \langle b_1 \rangle, \langle \gamma_0 \rangle, \langle \gamma_1 \rangle, \Delta_z, \Delta_{\phi z}\}$. Thus, Π_{SecCMP} securely realizes $\mathcal{F}_{\text{SecCMP}}$. $\square$

- **Parameters.** The servers $\mathcal{P}_0$ and $\mathcal{P}_1$ has, security settings $\{l, s\}$, pre-stored the authenticated MAC key $\llbracket \phi \rrbracket$ and the shared biometric database (i.e., identities and biometric templates) $\llbracket \mathbf{D} \rrbracket = \llbracket [I], [\mathbf{D}] \rrbracket$.
- **Input.** The client $\mathcal{C}$ sends secret shares of the fresh pre-processed biometric template $\llbracket \hat{T} \rrbracket$ with identity index $\llbracket I_C \rrbracket$ to $\mathcal{F}^{\text{FLAME}}$.
- **Computation.** $\mathcal{F}^{\text{FLAME}}$ internally computes the similarity scores between $\llbracket \hat{T} \rrbracket$ and all stored templates $\llbracket \mathbf{D}[i] \rrbracket$ using Π_{SecIP} , producing shared similarity results $\llbracket S[i] \rrbracket, \llbracket \phi S[i] \rrbracket$ for all $i \in [1, m]$. Then, it identifies the top-1 match using Π_{SecCMP} , and obtains the result $\llbracket \eta \rrbracket, \llbracket \phi \eta \rrbracket$.
- **Integrity Check.** $\mathcal{F}^{\text{FLAME}}$ runs $\Pi_{\text{MACCheck}}(\llbracket \eta \rrbracket, \llbracket \phi \eta \rrbracket)$. If the MAC check fails, $\mathcal{F}^{\text{FLAME}}$ aborts and outputs $\perp$.
- **Output.** If the MAC check passes, $\mathcal{F}^{\text{FLAME}}$ returns the authenticated result $\llbracket \eta \rrbracket$ to the service provider $\mathcal{S}$ and nothing to the client $\mathcal{C}$. $\mathcal{S}$ grants access if $\eta = 0$.

Figure 9. Ideal Functionality $\mathcal{F}^{\text{FLAME}}$.

Next, we present the security proof for Theorem 3, where $\mathcal{P}_0$ and $\mathcal{P}_1$ access to the simulator for ideal functionality of the supporting protocols Π_{OptSS} , Π_{SecIP} , Π_{SecCMP} , Π_{mult} , and Π_{MACCheck} of the FLAME scheme in the following specified order.

Hyb₀: $\mathcal{P}_0$ and $\mathcal{P}_1$ jointly execute the real Π_{OptSS} protocol to obtain the initial OptSS shares of the input data. All subsequent protocols, including Π_{SecIP} , Π_{SecCMP} , Π_{mult} , and Π_{MACCheck} , are replaced by their respective ideal functionalities accessed via the simulator. Since the ideal functionalities leak nothing beyond what is revealed by the output, the malicious adversary's view in this hybrid is determined solely by the execution of Π_{OptSS} , which have been proven in [6]. Therefore, the view of any PPT adversary in **Hyb₀** is simulatable, and the adversary learns nothing beyond the prescribed output.

Hyb₁: Building upon **Hyb₀**, $\mathcal{P}_0$ and $\mathcal{P}_1$ execute both the real Π_{OptSS} and Π_{SecIP} protocols, while the functionalities of Π_{SecCMP} , Π_{mult} , and Π_{MACCheck} remain ideal and are realized via the simulator. The difference between the views of $\mathcal{P}_0$ and $\mathcal{P}_1$ in **Hyb₁** and **Hyb₀** lies only in the substitution of the ideal functionality of Π_{SecIP} with its real execution. Since Π_{SecIP} is proven against the malicious adversary in Theorem 4, its view is simulatable, and thus no PPT adversary can distinguish **Hyb₁** from **Hyb₀**.

Hyb₂: Further to **Hyb₁**, $\mathcal{P}_0$ and $\mathcal{P}_1$ execute the real Π_{SecCMP} protocol in addition to the real executions of Π_{OptSS} and

Π_{SecIP} , while the remaining protocols Π_{mult} and Π_{MACCheck} are still implemented via ideal functionalities. As the only difference between **Hyb**₂ and **Hyb**₁ is the real execution of Π_{SecCMP} instead of its ideal functionality, and since Π_{SecCMP} is also simulatable (Theorem 5), it follows that no PPT adversary can distinguish **Hyb**₂ from **Hyb**₁.

Hyb₃: On the basis of **Hyb**₂, $\mathcal{P}_0$ and $\mathcal{P}_1$ additionally execute the real Π_{mult} protocol, while only Π_{MACCheck} is still realized by its ideal functionality via the simulator. Again, the only difference between the adversarial view in **Hyb**₃ and **Hyb**₂ is due to replacing the ideal version of Π_{mult} with its real protocol. As shown above, Π_{mult} is malicious secure [6], [35], so the adversary's view remains simulatable. Hence, **Hyb**₃ and **Hyb**₂ are indistinguishable.

Hyb₄: Finally, $\mathcal{P}_0$ and $\mathcal{P}_1$ execute the full FLAME scheme, including the real execution of Π_{OptSS} , Π_{SecIP} , Π_{SecCMP} , Π_{mult} , and Π_{MACCheck} protocols. The only difference between **Hyb**₄ and **Hyb**₃ lies in the replacement of the ideal version of Π_{MACCheck} with its real implementation. As the Π_{MACCheck} protocol is proven to be malicious secure [2] and its view simulatable, **Hyb**₄ is indistinguishable from **Hyb**₃.

In the above real protocol executions, any additive error introduced by a malicious adversary will cause the MAC check to fail, except with probability at most $\frac{1}{2^{l+s}-1}$. Therefore, the simulation remains statistically indistinguishable from the real execution, up to a negligible statistical error. This establishes the security of the FLAME scheme against malicious adversaries.