

CryptoMask: Privacy-preserving Face Recognition

Jianli Bai¹, Xiaowu Zhang², Xiangfu Song^{3, (✉)}, Hang Shao⁴, Qifan Wang¹, Shujie Cui⁵, and Giovanni Russello¹

¹ University of Auckland, Auckland, New Zealand
{jbai795, qwan301}@aucklanduni.ac.nz g.russello@auckland.ac.nz

² CloudWalk Technology, Beijing, China
zhangxiaowu@cloudwalk.com

³ National University of Singapore, Singapore, Singapore
songxf@comp.nus.edu.sg

⁴ Beijing Institute of Graphic Communication, Beijing, China
mir_soh@163.com

⁵ Monash University, Melbourne, Australia
shujie.cui@monash.edu

Abstract. Face recognition is a widely-used technique for identification or verification, where a verifier checks whether a face image matches anyone stored in a database. However, in scenarios where the database is held by a third party, such as a cloud server, both parties are concerned about data privacy. To address this concern, we propose CryptoMask, a privacy-preserving face recognition system that employs homomorphic encryption (HE) and secure multi-party computation (MPC). We design a new encoding strategy that leverages HE properties to reduce communication costs and enable efficient similarity checks between face images, without expensive homomorphic rotation. Additionally, CryptoMask leaks less information than existing state-of-the-art approaches. CryptoMask only reveals whether there is an image matching the query or not, whereas existing approaches additionally leak sensitive intermediate distance information. We conduct extensive experiments that demonstrate CryptoMask’s superior performance in terms of computation and communication. For a database with 100 million 512-dimensional face vectors, CryptoMask offers $\sim 5\times$ and $\sim 144\times$ speed-ups in terms of computation and communication, respectively.

Keywords: Face recognition · Privacy-preserving · Homomorphic Encryption · Secure Multiparty Computation.

1 Introduction

Biometric authentication has become increasingly vital in various applications in recent years. This work focuses on face recognition, which identifies or verifies a person’s identity based on their facial features. Due to its ease of use and convenience, face recognition has gained significant traction in real-world applications such as public place surveillance (*e.g.*, streets, airports, etc.) [22], social media [6], and corporate punch card supervision [13].

As face recognition systems become more widespread, concerns about privacy have grown. In a typical system, a server stores face images belonging to users who are registered. When a verifier, who possesses a user’s face image, queries the server to check if the user is verified, the system measures the similarity or distance between the queried image and the images in the database. However, in many cases, it may not be permissible to disclose users’ face images to the server due to privacy concerns or the possibility of human rights abuses [4]. *Therefore, it is essential to develop privacy-preserving face recognition protocols that protect data privacy while maintaining efficient recognition.*

Encrypting pre-processed images (e.g., extracted face vectors) and performing face recognition over encrypted data is a straightforward approach to ensure data privacy. Homomorphic Encryption (HE) is a promising encryption scheme for this purpose, which was first proposed in [27] and realized in [12]. HE allows computation in the encrypted domain without decryption. However, HE-based privacy-preserving face recognition protocols, such as the one proposed in [3], are several orders of magnitudes slower than the original method, even when utilizing the Single-Instruction-Multiple-Data (SIMD) technique [33] to amortize the cost of homomorphic operations. To overcome this, the approach proposed in [9] explores encoding methods on the image database, reducing the number of homomorphic multiplications and rotations required and improving computation efficiency. Moreover, previous works [3,9] in this field fail to protect the private information of the database, as they allow the verifier to learn sensitive distance or similarity information and the number of face images close to the queried one.

In this paper, we propose Cryptomask, an efficient privacy-preserving face recognition protocol that only reveals a single bit of information to the verifier, indicating whether the queried face image is present in the database. We propose a novel encoding method to encrypt the database in a compact manner, resulting in improved performance. For distance computation, we use efficient matrix multiplication techniques that avoid expensive homomorphic rotations. Additionally, we ensure the privacy of distance calculations by designing a secure result-revealing protocol and optimizing its efficiency. CryptoMask outperforms existing distance-based privacy-preserving biometric schemes constructed via HE in terms of computation and storage overhead, and information leakage. Table 1 provides a comparison of different schemes, showing that our approach requires the least number of HE multiplications and additions and has minimal information leakage. We implement CryptoMask and compare its performance with existing works [3] and [9]. In the case of a database with 100 million face images, CryptoMask outperforms others up to $\sim 5\times$ and $\sim 144\times$ in computation and communication, respectively.

1.1 Related Work

The early work given in [28] relies on secret sharing to authenticate face recognition. However, it cannot ensure the privacy of face images. There are some similar works [25,35,21] working for biometric authentication. Another line is employing pattern recognition to protect the queried database [23,19]. However, this method also fails to ensure the security of the database and the queried face image. Some works [31,37] employ secure multi-party computation (MPC) [38] to achieve the privacy-preserving

Table 1: Summary of existing privacy-preserving face recognition protocols.

| Protocol | Multiplication | Addition | Rotation | Memory | Leakage |
|---------------------------|---------------------------------|-----------------------------------|-------------|-------------------------------------|---|
| Naïve | md | $m(d-1)$ | 0 | $O(md\ell)$ | $\mathbf{A}, \mathbf{b}, \mathbf{d}, r$ |
| Hu <i>et al.</i> [14] | md^3 | $md^2(d-1)$ | 0 | $O(md^2)$ | \mathbf{d}, m |
| Pradel <i>et al.</i> [24] | md | $m(d-1)$ | 0 | $O(mdN)$ | \mathbf{d}, m |
| Boddeti <i>et al.</i> [3] | m | $m\log_2 d$ | $m\log_2 d$ | $O(mN)$ | \mathbf{d}, m |
| HERS [9] | $\lceil \frac{m}{N} \rceil d$ | $\lceil \frac{m}{N} \rceil (d-1)$ | 0 | $O(dN \lceil \frac{m}{N} \rceil)$ | \mathbf{d}, m |
| Erkin <i>et al.</i> [10] | $m(d+2)$ | $2m(d-1)$ | 0 | $O(mdN)$ | m |
| CryptoMask | $\lceil \frac{m}{N-d} \rceil d$ | $\lceil \frac{m}{N-d} \rceil d$ | 0 | $O(dN \lceil \frac{m}{N-d} \rceil)$ | m |

\mathbf{A} : database containing face vectors; \mathbf{b} : queried face vector; m : database size; d : dimension of each face vector; N : HE plaintext polynomial degree; ℓ : length of each element in face vector; \mathbf{d} : distance vector; r : face recognition result. The notation $\lceil x \rceil$ denotes rounding up to the nearest integer of x . naïve represents the face recognition performed in plaintext.

goals, yet they are communication costly due to multiple interactions between the participants. Homomorphic encryption [27] allows computations to be performed over encrypted data without first decrypting it. Many face recognition protocols [36,10,34,3,9] based on HE have been proposed. Unfortunately, they either result in heavy computation [10,34,3] or cannot provide full secrecy (*e.g.* leakage of distance similarity) [3,9]. We fill this gap by employing HE to perform distance computations and utilizing MPC to do a secure result-revealing process. Compared with the state-of-the-art [9], our work reduces both the computation and communication while maintaining the privacy of not only inputs and outputs but also intermediate data.

2 Background

In this section, we describe the face recognition algorithm and introduce the encoding method for a given matrix. Then we present some cryptographic primitives we use.

2.1 Face Recognition

In a face recognition system, each face image is represented by a feature vector, we say a face vector. The extraction algorithm usually consists of face detection, alignment, normalization, and feature extraction, which is out of the scope of this work. We assume the face vector of each image is ready to use. In fact, the face vector extracted from the facial images of the same person could be slightly different. Thus, for face recognition, we should compare the similarity between two face vectors rather than check the equality. A simple method is to use either the Euclidean distance [7] or the cosine similarity [32] to measure the similarity between two face vectors. In this paper, we employ cosine similarity. Specifically, given two vectors $\tilde{\mathbf{a}} = (\tilde{a}^0, \dots, \tilde{a}^{d-1}) \in \mathbb{Z}^d$ and $\tilde{\mathbf{b}} = (\tilde{b}^0, \dots, \tilde{b}^{d-1}) \in \mathbb{Z}^d$, their cosine similarity is $d(\tilde{\mathbf{a}}, \tilde{\mathbf{b}}) = \frac{\sum_{i=0}^{d-1} \tilde{a}^i \tilde{b}^i}{\sqrt{\sum_{i=0}^{d-1} (\tilde{a}^i)^2} \sqrt{\sum_{i=0}^{d-1} (\tilde{b}^i)^2}}$. By setting $a^i = \frac{\tilde{a}^i}{\|\tilde{\mathbf{a}}\|}$ and $b^i = \frac{\tilde{b}^i}{\|\tilde{\mathbf{b}}\|}$, which are the normalization representations, we can

convert it to $d(\tilde{\mathbf{a}}, \tilde{\mathbf{b}}) = \sum_{i=0}^{d-1} a^i b^i$. By doing so, $d(\tilde{\mathbf{a}}, \tilde{\mathbf{b}})$ can be considered as the inner product of vector $\mathbf{a} = (a^0, \dots, a^{d-1})$ and $\mathbf{b} = (b^0, \dots, b^{d-1})$. Note that a^i and b^i can be pre-computed offline. A larger value of $d(\tilde{\mathbf{a}}, \tilde{\mathbf{b}})$ means higher similarity between $\tilde{\mathbf{a}}$ and $\tilde{\mathbf{b}}$, and if it is greater than a threshold value, we say $\tilde{\mathbf{a}}$ and $\tilde{\mathbf{b}}$ matches with each other, *i.e.*, they represent the same person. In the following of this paper, all the face vectors are normalization representations.

2.2 Encoding Method

Given a set of encrypted face vectors, computing the cosine similarity one by one is time-consuming. A promising method is computing that in parallel. The encoding method from Cheetah [17] achieves the best paralleling performance. In the following, we briefly describe the encoding method in Cheetah [17].

Given a matrix $\mathbf{A} = \{\mathbf{a}_0, \mathbf{a}_1, \dots, \mathbf{a}_{\tilde{m}-1}\} \in \mathbb{Z}^{\tilde{m} \times d}$ with \tilde{m} rows and d columns, where $\mathbf{a}_i = (a_i^0, \dots, a_i^{d-1})$ and $0 \leq i \leq \tilde{m} - 1$, it can be represented into a polynomial as

$$\begin{aligned} \pi(\mathbf{A}) &= a_0^{d-1} X^0 + a_0^{d-2} X^1 + \dots + a_0^0 X^{d-1} \\ &\quad + a_1^{d-1} X^d + a_1^{d-2} X^{d+1} + \dots + a_1^0 X^{2d-1} + \\ &\quad \dots \\ &\quad + a_{\tilde{m}-1}^{d-1} X^{(\tilde{m}-1)d} + a_{\tilde{m}-1}^{d-2} X^{(\tilde{m}-1)d+1} + \dots + a_{\tilde{m}-1}^0 X^{\tilde{m}d-1}. \end{aligned}$$

Given another polynomial $\pi(\mathbf{b}) = b^0 X^0 + b^1 X^1 + \dots + b^{d-1} X^{d-1}$, we can get polynomial $\pi(\mathbf{d})$ by computing $\pi(\mathbf{d}) \leftarrow \pi(\mathbf{A}) * \pi(\mathbf{b})$, where $*$ denotes polynomial multiplication. It is notable that the coefficient of degree $X^{(i+1)d-1}$, where $i \in [0, \tilde{m} - 1]$ in polynomial $\pi(\mathbf{d})$ forms the dot product result of the i -th row vector from \mathbf{A} and the vector \mathbf{b} . The correctness comes from the fact that the elements order of each vector in matrix \mathbf{A} is revised when it is encoded into a polynomial. We refer readers to Cheetah [17] to see the detailed proof of correctness.

2.3 Homomorphic Encryption

HE [1] allows us to compute over encrypted data where the result is indeed the encrypted version of the operations on the plaintext. In this work, we use a lattice-based HE: ring learning with errors (RLWE)-based HE called BFV [11]. We briefly describe the construction of BFV scheme. See [11] for a detailed formal description and security definition.

BFV Scheme. The plaintext space of BFV scheme is taken from $R_t = \mathbb{Z}_t/(x^N + 1)$ which represents polynomials with degree less than N where N is a power of 2, with the coefficients modulo t . Similarly, the ciphertext is defined in a ring R_q with the coefficients modulo q . We use symbols \boxplus and \boxtimes to represent homomorphic addition and homomorphic multiplication, respectively. The BFV scheme consists of the following algorithms:

- $(pk, sk) \leftarrow \text{KeyGen}(1^\lambda)$: On input the security parameter λ , it generates a pair of keys (pk, sk) .

- $ct \leftarrow \text{Encrypt}(pk, m)$: On input the public key pk and the plaintext m , it outputs the ciphertext ct .
- $m \leftarrow \text{Decrypt}(sk, ct)$: On input the secret key sk and the ciphertext ct , it outputs a plaintext m .
- $\text{Eval}(ct_i, ct_j)$: Given two ciphertexts ct_i and ct_j , output a ciphertext corresponding to the following operation.
 - $\text{Eval.Add}(ct_i, ct_j)$: Output $ct \leftarrow ct_i \boxplus ct_j$.
 - $\text{Eval.Mul}(ct_i, ct_j)$: Output $ct \leftarrow ct_i \boxtimes ct_j$.

2.4 Key-switching

Key-switching enables the data encrypted by one set of encryption keys to be re-encrypted by another without decrypting the data. BFV scheme [11] naturally supports the key-switching operation. The key-switching process consists of two algorithms:

- $k_{A \rightarrow B} \leftarrow \text{SwKeyGen}(sk_A, sk_B)$: On input two BFV secret keys sk_A, sk_B , it outputs a key-switching key $k_{A \rightarrow B}$.
- $ct_B \leftarrow \text{Switching}(ct_A, k_{A \rightarrow B})$: On input a key-switching key $k_{A \rightarrow B}$ and a ciphertext ct_A encrypted by a public key pk_A associated with sk_A , it outputs a ciphertext ct_B encrypted by a public key pk_B associated with sk_B .

More details about the key-switching technique can be found in [20].

2.5 Secret Sharing

For an l -bit value $x \in \mathbb{Z}_{2^l}$, we use $\langle x \rangle^A$ to denote x is arithmetically shared between parties P_0 and P_1 where P_0 holds x_0^A and P_1 holds x_1^A such that $x = x_0^A + x_1^A$ with $x_0^A, x_1^A \in \mathbb{Z}_{2^l}$. Similarly, $\langle x \rangle^B$ denotes a boolean share of x where $x = x_0^B \oplus x_1^B$ with $x_0^B, x_1^B \in \mathbb{Z}_{2^l}$. Note that each share itself does not reveal any information about x . In some cases, we need the conversion between different sharing formats. We use the **B2A** technique to convert x from its boolean sharing $\langle x \rangle^B$ to its arithmetic sharing $\langle x \rangle^A$, which we represent as $(x_0^A, x_1^A) \leftarrow \text{B2A}(x_0^B, x_1^B)$. The detailed **B2A** conversion can be referred to [8]. If \mathbf{x} is a vector, then $\mathbf{x} = \mathbf{x}_0^A + \mathbf{x}_1^A$ means each element in the vector is additionally shared between two parties. In our design, the cloud server (CS) plays the role of P_0 , and the verifier plays the role of P_1 .

2.6 Secure Comparison

Secure comparison, also known as Millionaire’s problem [38], compares two integers held by two parties. The inputs contain x from one party and y from another party, and the output bit 1 or 0 is shared between the two parties. Cryptflow2 [26] proposes an efficient comparison protocol based on the observation: assume $x = x_1 || x_0$ and $y = y_1 || y_0$, we must have $x < y$ either when $x_1 = y_1$ and $x_0 < y_0$ or when $x_1 < y_1$, i.e., $\mathbf{1}\{x < y\} = (\mathbf{1}\{x_1 = y_1\} \wedge \mathbf{1}\{x_0 < y_0\}) \oplus \mathbf{1}\{x_1 < y_1\}$ ¹. By separating the binary represented values into small parts, the queried Oblivious Transfer (OT) [18] is

¹ $\mathbf{1}\{condition\}$ and $\mathbf{0}\{condition\}$ mean the condition is true and false, respectively.

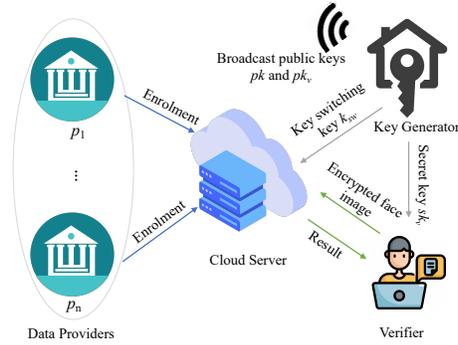


Fig. 1: System model.

also small, optimizing the communication cost. Recently, Cheetah [17] provides further optimization by replacing the underlying secure AND implementation with Random OT (ROT) [18] generated Beaver Triples [2]. For simplicity, we represent secure comparison as $(b_0, b_1) \leftarrow SC_{lt}(x, y)$ which means one party inputs x and another party inputs y and outputs $b = 1$ if $x < y$ and $b = 0$ otherwise, where $b = b_0 \oplus b_1$. For more details about the state-of-the-art secure comparison, please refer to [17,26].

3 Overview of Our Approach

This section describes the system model and threat model and overviews CryptoMask.

3.1 System Model

In CryptoMask, we consider the scenario where the database is stored on a cloud server, and the corresponding face vectors are received from a group of data providers. A verifier wants to check if a given face image matches an image in the database. Our system consists of four types of entities: a trusted **Key Generator (KG)** who generates keys for other entities for privacy-preserving purposes. A group of **Data Providers (DPs)** who upload extracted face vectors to a cloud server, a **Cloud Server (CS)** who stores the database of face vectors, and a **Verifier** who checks if a given face vector is in the database, as shown in Fig. 1.

KG. KG generates a pair of HE public/private keys (pk, sk) and distributes pk to other entities. KG also generates another pair of public/private keys (pk_v, sk_v) and sends them to the verifier. When KG receives a “setup” request from the verifier, it computes a key-switching key k_{sw} based on sk and sk_v and sends it to CS.

DPs. In our system, DPs can upload images (represented by face vectors) to CS. To keep their data private, DPs encrypt the face vectors using the public key pk before uploading them to CS. We call this process *enrolment*.

CS. CS stores the encrypted face vectors. It performs face recognition protocol with the verifier without learning anything about the queried face information or the result.

Verifier. The verifier has a face image and intends to check if the image is in the database by performing a privacy-preserving face recognition protocol with CS. We call this process *evaluation*. It learns the image exists in the database if the check result is one. For example, a verifier can be a service provider who receives or collects a face image from a user after the user’s consent. The verifier then wants to check whether the user is a verified user in order to provide subsequent service.

3.2 Threat Model

Similar to previous work, such as [3] and [9], we assume the CS and the verifier are honest-but-curious (semi-honest). That is, they will follow the protocol honestly but may try to infer as much information as possible. We also assume CS and the verifier will never collude with each other. It is reasonable in practice because CS (*e.g.*, education management organization) is motivated to maintain its reputation and is not likely to take the risk of colluding with the verifier. The KG is a fully trusted party.

3.3 Overview of CryptoMask

Encrypting each face vector with HE and computing the cosine similarity between the query and each vector in the database is a straightforward but expensive way to perform face recognition securely. With m face vectors and d features per vector, this method requires md homomorphic multiplications, which can be significantly time-consuming. Additionally, this approach poses a privacy risk by leaking sensitive information, such as the computed distance vectors \mathbf{d} . Previous works, such as those proposed in [3,9,10,24], also suffer from the same issue. To tackle all the issues above, we introduce CryptoMask. In particular, we design a novel encoding method to enhance performance and a secure result-revealing protocol to minimize information leakage.

To reduce both the communication and computation overhead, our main idea is to encrypt face vectors in batches and compute the cosine similarity between the query and a batch of face vectors, rather than one by one. Specifically, during the enrollment process, given a batch of face vectors, DP encodes them into one BFV ciphertext ct_i and sends it to CS. When the verifier queries for an image, CS performs only one homomorphic multiplication between each BFV ciphertext ct_i and the encrypted query. The resulting ciphertext contains the cosine similarity between batched face vectors and the queried face vector. To determine if the queried image matches any image stored in the CS, the next step is to compare the cosine similarity with the threshold. Directly revealing the cosine similarity results to the verifier or the CS exposes sensitive information. For example, they can learn how many face images in the database are similar to the given one. To avoid such leakage, CryptoMask runs a secure result-revealing protocol between CS and the verifier, which only reveals whether the queried face image exists in the database to the verifier.

To further enhance the performance of CryptoMask, we can adopt a paralleling technique to compute the cosine similarity between the query and batched face vectors. As done in work [3] [9], the homomorphic multiplication performed during the evaluation can be processed in parallel with the SIMD technique. However, this technique requires a prime plaintext modulus[17], implying that the homomorphic encryption must be

performed in \mathbb{Z}_p with p as prime. In our secure result-revealing protocol, the secure comparison is a non-linear function, and [26] has shown that OT-based protocols on the ring \mathbb{Z}_{2^t} perform 40%-60% better than on the prime field \mathbb{Z}_p in bandwidth consumption, with almost no cost for modulo reduction. Hence, in this work, rather than employing SIMD, we opt for the parallelization technique from [17] to compute homomorphic multiplication in parallel. This technique enables us to work exclusively in the ring domain \mathbb{Z}_{2^t} and brings another efficiency improvement by avoiding expensive rotation, the key operation for SIMD-based work. Furthermore, while [17] necessitates an extraction algorithm (RLWE-based ciphertext to LWE-based ciphertext) for useful information extraction from the resulting ciphertext, we avoid it by masking the resulting ciphertext and sending it back to the verifier, which is more efficient.

3.4 Data Representation

The coefficients of the BFV plaintext polynomial must be integers. To achieve this, we need to encode our real-valued representation $\mathbf{A} \in \mathbb{R}^{m \times d}$ as an integer-valued representation, which we denote by $\mathbf{A} \in \mathbb{Z}^{m \times d}$. For the remainder of the paper, we use \mathbf{A} to refer to the matrix where all elements are integers. We scale the real-valued features into integers using a specified precision. This scaling method results in a loss of precision during computation. In our experiments, we evaluate the level of precision loss by setting different precision scales, and report the results in Table 2 in Appendix B.

4 CryptoMask Details

This section describes the enrollment and evaluation processes of CryptoMask in detail.

4.1 Our Encoding Method

BFV scheme [11] is designed to work on a polynomial ring $R_t = \mathbb{Z}_t/(x^N + 1)$ with degree N . The observation is that the number of slots in a polynomial (*e.g.*, 4096) is far more than the dimension of a face vector (*e.g.*, $d = 128$). Thus, we can employ one polynomial to represent multiple face vectors as done in Cheetah [17]. In our design, each row in the matrix \mathbf{A} represents a face vector. That is, before encrypting and uploading the face vectors to CS, DP encodes them into a matrix \mathbf{A} and then transforms it into the polynomial $\pi(\mathbf{A})$. Then DP encrypts this polynomial using BFV as ct_i and sends it to CS. The verifier encrypts the queried face vector \mathbf{b} as ct and sends it to CS. The cosine similarity is computed by multiplying these two ciphertexts ct_i and ct , whose underlying plaintext polynomial is exactly $\pi(\mathbf{d})$. As mentioned, the plaintext space of BFV scheme is taken from $R_t = \mathbb{Z}_t/(x^N + 1)$, which means the maximum degree of a plaintext polynomial is N . The direct method is we fill all the coefficients slots in the plaintext polynomial when considering encoding our face vectors database. However, this might result in a loss of valid similarity. The reason is that the valid value in the product will be dropped (modulo reduced to a position with a degree less than N) if its associated degree is greater than N , which means we will get the wrong distance between the last face vector in the matrix and the queried image. Our idea is to leave

Algorithm 1 Secure enrolment

Input: An indicator ind and the last ciphertext ct_{la} from CS; n_u d -dimensional face vectors $\mathbf{V} = \{\mathbf{a}_0, \dots, \mathbf{a}_{n_u-1}\} \in \mathbb{Z}^{n_u \times d}$ and public key pk from DP.

Parameter: $\delta = \lceil \frac{N-d}{d} \rceil$ where N is the plaintext polynomial degree.

Output: CS adds the encrypted face vectors to the database.

- 1: DP informs CS to add new face vectors. CS sends ind to DP.
- 2: DP takes $\delta - ind$ face vectors and organizes them into a matrix $\mathbf{A}_0 \in \mathbb{Z}^{\delta \times d}$ by padding ind zero vectors before these real samples. Then DP represents \mathbf{A}_0 as $\pi(\mathbf{A}_0)$ and gets $ct_0 \leftarrow \text{Encrypt}(pk, \pi(\mathbf{A}_0))$.
- 3: DP separates the remaining vectors into $e\delta$ vectors and remains f vectors where $f < \delta$ and $n_u = \delta - ind + e\delta + f$.
- 4: DP constructs e polynomials $\pi(\mathbf{A}_1), \dots, \pi(\mathbf{A}_e)$ using $e\delta$ face vectors and performs $ct_i \leftarrow \text{Encrypt}(pk, \pi(\mathbf{A}_i))$ for each $i \in [1, e]$.
- 5: DP pads $\delta - f$ zero vectors to the remaining f vectors and gets $\pi(\mathbf{A}_{e+1})$. Then DP encrypts it as $ct_{e+1} \leftarrow \text{Encrypt}(pk, \pi(\mathbf{A}_{e+1}))$ and sets $ind \leftarrow \delta - f$.
- 6: DP uploads $\{ct_0, \dots, ct_{e+1}\}$ and ind to CS.
- 7: After receiving the ciphertexts, CS first updates ind and saves $\{ct_1, \dots, ct_{e+1}\}$. Then CS performs $ct_{la} \leftarrow \text{Eval.Add}(ct_{la}, ct_0)$.

the last d positions in the polynomial $\pi(\mathbf{A})$ for “buffer” use and set their coefficients as 0. Thus, all valid values will be presented as coefficients with degrees less than N . That is, if the degree of a plaintext polynomial is N , we only encode its lower $N - d$ coefficients and leave the higher d coefficients as zeros. A similar strategy applies to the queried face vector. Using this encoding method, the concrete number of ciphertext for m face vectors with dimension d will be $\lceil \frac{md^2}{N-d} \rceil$.

4.2 Enrolment Process

Based on our encoding method, we improve enrollment efficiency by reducing the number of ciphertexts uploaded by DPs. Specifically, we use one plaintext polynomial with degree N to represent $\lceil \frac{N-d}{d} \rceil$ face vectors, which results in only one homomorphic ciphertext. Thus, DP only needs to upload a single homomorphic ciphertext for $\lceil \frac{N-d}{d} \rceil$ images to CS while the state-of-the-arts [3] and [9] require $\lceil \frac{N-d}{d} \rceil$ and d ciphertext, respectively. This encoding strategy is also beneficial to CS for saving storage overhead compared with work [3], [9]. The reason is that our designed encoding method allows CS to merge its last stored ciphertext with a new one that comes from another DP.

The details of the enrollment process are given in Algorithm 1. We suppose CS already stored some encrypted face vectors under the public key pk and a DP then wants to add n_u d -dimensional face vectors $\mathbf{V} = \{\mathbf{a}_0, \dots, \mathbf{a}_{n_u-1}\}$ to CS. CS maintains an indicator ind , which tells DP the start vacant position in the last stored ciphertext. Rather than directly encrypting these vectors and sending them to CS, DP first encodes the data based on our proposed encoding method and then performs BFV encryption over the encoded data. When receiving the indicator ind from CS, DP divides its vectors into three parts. The first part contains $\delta - ind$ vectors where $\delta = \lceil \frac{N-d}{d} \rceil$ represents the maximum number of face vectors that can be encoded into a polynomial. Since CS is allowed to merge the last ciphertext with a newly come one, DP organizes the first $\delta - ind$

Algorithm 2 Secure distance computation

Input: An encrypted database $\{ct_0, \dots, ct_{s-1}\}$, where each ct_i is the ciphertext of a $\delta \times d$ matrix $\mathbf{A} = \{\mathbf{a}_0, \mathbf{a}_1, \dots, \mathbf{a}_{s-1}\} \in \mathbb{Z}^{\delta \times d}$ with $m = s\delta$; A queried face vector $\mathbf{b} \in \mathbb{Z}^d$ from verifier.

Parameter: $\delta = \lceil \frac{N-d}{d} \rceil$ where N is the plaintext polynomial degree.

Output: CS gets the secret share \mathbf{d}_0^A and the verifier gets the secret share \mathbf{d}_1^A where $\mathbf{d} = \mathbf{d}_0^A + \mathbf{d}_1^A$ is an m -length vector of computed distances.

- 1: The verifier sends a “setup” signal to KG. Then KG generates a key-switching key $k_{sw} \leftarrow \text{SwKeyGen}(sk, sk_v)$ and sends it to CS.
- 2: The verifier encodes and encrypts \mathbf{b} as $ct \leftarrow \text{Encrypt}(pk, \pi(\mathbf{b}))$ and sends ct to CS.
- 3: CS and the verifier generate two empty vectors \mathbf{d}_0^A and \mathbf{d}_1^A , respectively.
- 4: **for** $i \in [0, s-1]$ **do**
- 5: CS computes $ct'_i \leftarrow \text{Eval.Mul}(ct, ct_i)$.
- 6: CS randomly generates a plaintext polynomial $\mathbf{r}_i = r_0X^0 + \dots + r_{N-1}X^{N-1}$.
- 7: CS extracts its the $(kd-1)$ -th coefficients from \mathbf{r}_i and sets $\mathbf{d}_0^A[i\delta + k - 1] \leftarrow -r_{kd-1}$ where $k \in [1, \delta]$.
- 8: CS computes $ct''_i \leftarrow \text{Eval.Add}(\mathbf{r}_i, ct'_i)$.
- 9: CS performs $c'_i \leftarrow \text{Switching}(k_{sw}, ct''_i)$.
- 10: **end for**
- 11: CS sends $\{c'_0, \dots, c'_{s-1}\}$ to the verifier and keeps $\mathbf{d}_0^A[i]$ where $i \in [0, m-1]$.
- 12: **for** $i \in [0, s-1]$ **do**
- 13: The verifier performs $\mathbf{p}_i \leftarrow \text{Decrypt}(sk_v, ct'_i)$ for each $i \in [0, s-1]$, where $\mathbf{p}_i = a_0X^0 + \dots + a_{N-1}X^{N-1}$.
- 14: The verifier extracts the $(kd-1)$ -th coefficients $a_{(kd-1)}$ from polynomial \mathbf{p}_i and sets $\mathbf{d}_1^A[i\delta + k - 1] \leftarrow a_{kd-1}$ where $k \in [1, \delta]$.
- 15: **end for**

vectors into a matrix $\mathbf{A}_0 \in \mathbb{Z}^{\delta \times d}$ by padding ind zero vectors before these real samples. This matrix is encrypted as ct_0 . When CS receives ct_0 , it can merge it to its last stored ciphertext ct_{la} by simply performing a homomorphic addition $\text{Eval.Add}(ct_{la}, ct_0)$. The second part contains $e\delta$ vectors, and the last part contains f vectors where $f < \delta$ and $n_u = \delta - ind + e\delta + f$. DP encrypts matrices $\mathbf{A}_1, \dots, \mathbf{A}_e$ in the second part separately using BFV and sends them to CS. Unlike the first part, for the last part, DP first pads $\delta - f$ zero vectors to the remaining vectors, then encrypts it as ct_{e+1} and sends it to CS. In the last, DP updates the indicator $ind = \delta - f$ and sends it to CS for further use.

4.3 Evaluation Process

The evaluation process happens between a verifier and a CS. Specifically, as shown in Algorithm 2, given a face vector, the verifier first encodes it into a polynomial. Then the verifier encrypts the polynomial using the public key pk and sends it to CS. CS gets a key-switching key k_{sw} from KG after KG receives a “setup” signal from the verifier. After receiving the encrypted query ct from the verifier, CS first runs local homomorphic multiplication between each ciphertext ct_i stored in CS and ct , where $i \in [0, s-1]$. Rather than directly sending the computed results to the verifier, CS masks each of them using a randomly selected plaintext polynomial \mathbf{r}_i . CS can easily extract the $(kd-1)$ -th coefficients r_{kd-1} from \mathbf{r}_i where $k \in [1, \delta]$ and keeps its additive inverse into $\mathbf{d}_0^A[i\delta + k - 1]$, which is one of the secret parts of computed distances.

Algorithm 3 Secure result-revealing

Input: The secret share of distance vector \mathbf{d}_0^A from CS; The secret share of distance vector \mathbf{d}_1^A and a threshold ts from verifier.

Output: The verifier learns whether its face image exists in the database.

- 1: **for** $i \in [0, m - 1]$ **do**
- 2: The verifier updates $\mathbf{d}_1^A[i] \leftarrow ts - \mathbf{d}_1^A[i]$.
- 3: CS and verifier jointly run $(b_0^B[i], b_1^B[i]) \leftarrow \text{SC}_{It}(\mathbf{d}_1^A[i], \mathbf{d}_0^A[i])$.
- 4: CS and verifier jointly perform $(b_0^A[i], b_1^A[i]) \leftarrow \text{B2A}(b_0^B[i], b_1^B[i])$.
- 5: **end for**
- 6: CS computes $b_0 = \sum_{i=0}^{m-1} b_0^A[i]$ and verifier computes $b_1 = \sum_{i=0}^{m-1} b_1^A[i]$.
- 7: CS and verifier jointly perform $(\mu_0, \mu_1) \leftarrow \text{SC}_{It}(-b_0, b_1)$.
- 8: CS sends μ_0 to the verifier. The verifier computes $\mu \leftarrow \mu_0 \oplus \mu_1$ and learns its face image is in the database by $\mu = 1$. Otherwise, it learns its face image is not in the database by $\mu = 0$.

To enable the verifier to perform decryption by itself, CS transfers each ciphertext encrypted by pk to pk_v by a key-switching technique before sending them to the verifier. With the masked distances, CS does not require performing RLWE to LWE extraction function, a key design in [17]. The extraction function is considered time-consuming as it is performed over homomorphic ciphertext [5]. In our design, the verifier can extract the coefficients by itself after decrypting the RLWE ciphertext. Doing this saves the homomorphic extraction overhead on the CS side. Besides, we also reduce the required communication for $\lceil \frac{N-d}{d} \rceil$ face vectors from $\lceil \frac{N-d}{d} \rceil (N+1)q$ to $2Nq$ where q denotes the ciphertext coefficients modulo. After decrypting all the received ciphertext, the verifier similarly extracts coefficients from obtained polynomial and saves them into \mathbf{d}_1^A , which is another part of secret-shared computed distances.

Then CS runs a secure result-revealing protocol with the verifier as shown in Algorithm 3. For each shared distance, CS and the verifier jointly run a secure comparison to compute $\mathbf{d}_1^A[i] < \mathbf{d}_0^A[i]$, where $\mathbf{d}_1^A[i] \leftarrow ts - \mathbf{d}_1^A[i]$ is from the verifier and $\mathbf{d}_0^A[i]$ is from CS. Clearly, the result represents the less than comparison between the given threshold ts and the distance $\mathbf{d}[i]$. However, the comparison result is in binary format, so we cannot directly aggregate all results. Thus, we need a **B2A** conversion $(b_0^A[i], b_1^A[i]) \leftarrow \text{B2A}(b_0^B[i], b_1^B[i])$. After that, CS can compute $b_0 = \sum_{i=0}^{m-1} b_0^A[i]$ and the verifier computes $b_1 = \sum_{i=0}^{m-1} b_1^A[i]$. To obtain the queried result, CS and verifier jointly perform $(\mu_0, \mu_1) \leftarrow \text{SC}_{It}(-b_0, b_1)$ and CS sends μ_0 to the verifier. In the end, the verifier learns whether the queried face image exists in the database by computing $\mu \leftarrow \mu_0 \oplus \mu_1$.

4.4 Security Analysis

The security of CryptoMask follows from the semantic security of HE and the security of MPC. The complexity and security analysis can be found in Appendix A.

4.5 Optimizations

We present some optimizations to improve the efficiency of CryptoMask.

Reducing Computation Overhead. In Algorithm 2, CS should run a key-switching before sending the masked distance ciphertext to the verifier, which is time-consuming. We can put this key-switching when the verifier first sets up. Rather than sending the face vector encrypted by pk , the verifier encrypts it using its public key pk_v . Then all computations in CS are over the encrypted data over pk_v . However, this is a trade-off since it will save computation overhead but increase CS’s storage.

Reducing Communication Overhead. We employ the ciphertext compression technique from SEAL library [30], compressing the original ciphertext into around two-thirds of the original size. Notably, this ciphertext compression can only be used for data to be decrypted because it will cause a decryption error if the data is computed over compressed ciphertext. Clearly, CryptoMask can benefit from the compression technique. Another ciphertext size reduction of CryptoMask is gained from Cheetah [17]. The observation is that CS only needs to send high-end bits of two parts of ciphertext to the verifier. In this way, we save around 16% – 25% communication with a negligible decryption failing chance (*i.e.*, $< 2^{-38}$). For a more detailed analysis, see [17].

5 Performance Evaluation

We implemented a prototype of CryptoMask on top of Cheetah [17] and evaluated its performance with different datasets. In this section, we present our experimental results.

Experimental Setup. The experiment runs on a laptop running Centos 7.9 equipped with Xeon(R) Gold 6240 2.6GHZ CPU with 32 GB RAM. The network setting is LAN with RTT 0.1 ms and bandwidth 1 Gbps. We run all the experiments in a single-threaded environment. We set the BFV parameter N as 4096, t as 20 bits, and q as $60 + 49$ bits. The security level λ is set as 128 bits. We also evaluated the performance of the existing works [3] and [9] in the same environment with the same values for parameters. We compared their results with CryptoMask. The time we report is averaged over ten trials.

Datasets. Similar to [3] and [9], we evaluate the performance of CryptoMask with datasets that have different numbers of face images and dimensions. To show how the accuracy is influenced by precision scaling, as done in [3], we use a real dataset LFW [15] for the evaluation, which can be obtained from [16]. Specifically, LFW consists of 13,233 face images of 5,749 subjects. As done in [3] and [9], We utilize the state-of-the-art face representation FaceNet [29] to extract face vectors.

5.1 Efficiency

Following the same dataset construction from [9], we evaluate CryptoMask on four representations at different dimensions (32-D, 64-D, 128-D, and 512-D). Fig. 2 and Fig. 3 separately report the concrete computation and communication overhead with dataset sizes varying from 1 to 100 million. In the following, for simplicity, we use SFM to name the work in [3] and use HERS to name the work in [9].

Computation overhead. We report two computation overhead lines of CryptoMask in Fig. 2 where CryptoMask-W denotes we fully implement CryptoMask while CryptoMask-WO represents the version without the secure result-revealing protocol. In particular,

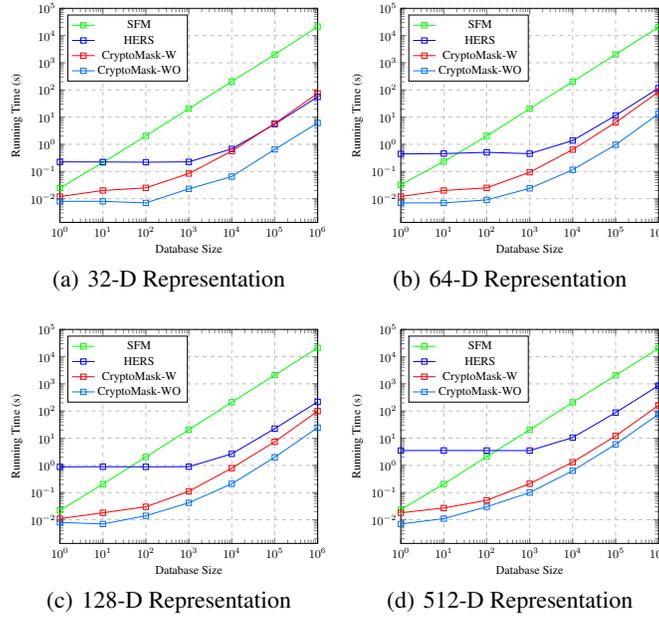


Fig. 2: Performance of evaluation process.

CryptoMask-WO, SFM, and HERS have comparable information leakage, where they all leak the computed similarity to the verifier.

From Fig. 2 we can see both CryptoMask-W and CryptoMask-WO outperform SFM in the four dimensions settings. The reason is the primary computation overhead in secure face recognition is caused by the homomorphic multiplication, which is m times in [3] while it is $\lceil \frac{m}{N-d} \rceil d$ times in CryptoMask. Compared with HERS, CryptoMask-WO shows the same tendency but enjoys less computation overhead. The main reason is we provide optimizations for computation. As for CryptoMask-W, the required computation overhead is near to HERS but achieves better security by concealing the similarity between face vectors from the verifier. CryptoMask is sensitive to the feature dimension, and the running time gap between SFM and CryptoMask-W drops with the increase of the dimension. For example, when working on 32-D, CryptoMask-W outperforms SFM by $283\times$ against a gallery of 100 million. When working on 512-D, CryptoMask-W only saves around $132\times$ computation than SFM, yet CryptoMask still shows its high efficiency for the large-scale dataset. Even when compared with similar work HERS, CryptoMask-W lies between CryptoMask-WO and HERS, indicating that it enjoys a better computation overhead while ensuring database security.

Communication. Fig. 3 details the communication consumption of CryptoMask-W, SFM and HERS. It shows that CryptoMask-W requires the least communication resource than the other two. The main reason comes from the given communication optimizations mentioned in Section 4.5.

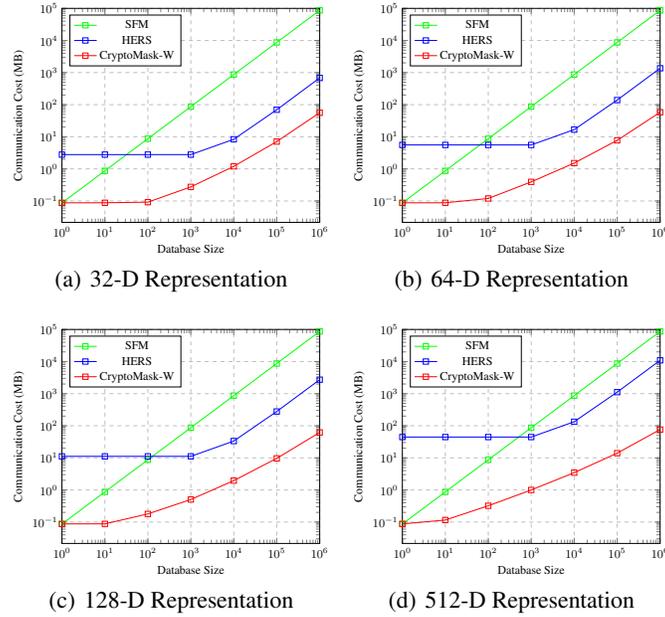


Fig. 3: Communication overhead comparison of our protocol with SFM and HERS.

6 Conclusion

We introduce CryptoMask, a practical privacy-preserving face recognition protocol that leverages homomorphic encryption and secure multi-party computation techniques. Our encoding strategy facilitates an efficient enrollment process, enabling DP to add more face vectors to CS. We construct an efficient matrix computation for distance calculation, based on our encoding method. Unlike existing state-of-the-art techniques that reveal the computed distance to the verifier, we protect intermediate results using a secure result-revealing protocol. Our experiments show that CryptoMask outperforms existing approaches in both computation and communication.

Acknowledgment

We thank the anonymous reviewers for their insightful comments and suggestions. Bai and Russello would like to acknowledge the MBIE-funded programme STRATUS (UOWX1503) for its support and inspiration for this research.

References

1. Acar, A., Aksu, H., Uluagac, A.S., Conti, M.: A survey on homomorphic encryption schemes: Theory and implementation. *ACM Computing Surveys (Csur)* **51**(4), 1–35 (2018)

2. Beaver, D.: Efficient multiparty protocols using circuit randomization. In: Annual International Cryptology Conference. pp. 420–432. Springer (1992)
3. Boddeti, V.N.: Secure face matching using fully homomorphic encryption. In: 2018 IEEE 9th International Conference on Biometrics Theory, Applications and Systems (BTAS). pp. 1–10. IEEE (2018)
4. Bowyer, K.W.: Face recognition technology: security versus privacy. *IEEE Technology and society magazine* **23**(1), 9–19 (2004)
5. Chen, H., Dai, W., Kim, M., Song, Y.: Efficient homomorphic conversion between (ring) lwe ciphertexts. In: International Conference on Applied Cryptography and Network Security. pp. 460–479. Springer (2021)
6. Cherepanova, V., Goldblum, M., Foley, H., Duan, S., Dickerson, J., Taylor, G., Goldstein, T.: Lowkey: Leveraging adversarial attacks to protect social media users from facial recognition. arXiv preprint arXiv:2101.07922 (2021)
7. Danielsson, P.E.: Euclidean distance mapping. *Computer Graphics and image processing* **14**(3), 227–248 (1980)
8. Demmler, D., Schneider, T., Zohner, M.: Aby-a framework for efficient mixed-protocol secure two-party computation. In: NDSS (2015)
9. Engelsma, J.J., Jain, A.K., Boddeti, V.N.: Hers: Homomorphically encrypted representation search. *IEEE Transactions on Biometrics, Behavior, and Identity Science* (2022), <https://github.com/human-analysis/secure-face-matching>
10. Erkin, Z., Franz, M., Guajardo, J., Katzenbeisser, S., Lagendijk, I., Toft, T.: Privacy-preserving face recognition. In: International symposium on privacy enhancing technologies symposium. pp. 235–253. Springer (2009)
11. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. *Cryptology ePrint Archive* (2012)
12. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the forty-first annual ACM symposium on Theory of computing. pp. 169–178 (2009)
13. Haigh, T.: The chromium-plated tabulator: Institutionalizing an electronic revolution, 1954–1958. *IEEE Annals of the History of Computing* **23**(4), 75–104 (2001)
14. Hu, S., Li, M., Wang, Q., Chow, S.S., Du, M.: Outsourced biometric identification with privacy. *IEEE Transactions on information forensics and security* **13**(10), 2448–2463 (2018)
15. Huang, G.B., Mattar, M., Berg, T., Learned-Miller, E.: Labeled faces in the wild: A database for studying face recognition in unconstrained environments. In: Workshop on faces in ‘Real-Life’ Images: detection, alignment, and recognition (2008)
16. Huang, G.B., Ramesh, M., Berg, T., Learned-Miller, E.: Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Tech. Rep. 07-49, University of Massachusetts, Amherst (October 2007)
17. Huang, Z., Lu, W.j., Hong, C., Ding, J.: Cheetah: Lean and fast secure two-party deep neural network inference. *IACR Cryptol. ePrint Arch.* **2022**, 207 (2022), <https://github.com/Alibaba-Gemini-Lab/OpenCheetah>
18. Ishai, Y., Kilian, J., Nissim, K., Petrank, E.: Extending oblivious transfers efficiently. In: Annual International Cryptology Conference. pp. 145–161. Springer (2003)
19. Jin, Z., Hwang, J.Y., Lai, Y.L., Kim, S., Teoh, A.B.J.: Ranking-based locality sensitive hashing-enabled cancelable biometrics: Index-of-max hashing. *IEEE Transactions on Information Forensics and Security* **13**(2), 393–407 (2017)
20. Kim, A., Polyakov, Y., Zucca, V.: Revisiting homomorphic encryption schemes for finite fields. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 608–639. Springer (2021)
21. Lee, Y.J., Park, K.R., Lee, S.J., Bae, K., Kim, J.: A new method for generating an invariant iris private key based on the fuzzy vault system. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* **38**(5), 1302–1313 (2008)

22. Parmar, D.N., Mehta, B.B.: Face recognition methods & applications. arXiv preprint arXiv:1403.0485 (2014)
23. Patel, V.M., Ratha, N.K., Chellappa, R.: Cancelable biometrics: A review. *IEEE signal processing magazine* **32**(5), 54–65 (2015)
24. Pradel, G., Mitchell, C.: Privacy-preserving biometric matching using homomorphic encryption. In: 2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom). pp. 494–505. IEEE (2021)
25. Rao, Y.S., Sukonkina, Y., Bhagwati, C., Singh, U.K.: Fingerprint based authentication application using visual cryptography methods (improved id card). In: TENCON 2008-2008 IEEE Region 10 Conference. pp. 1–5. IEEE (2008)
26. Rathee, D., Rathee, M., Kumar, N., Chandran, N., Gupta, D., Rastogi, A., Sharma, R.: Cryptflow2: Practical 2-party secure inference. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. pp. 325–342 (2020)
27. Rivest, R.L., Adleman, L., Dertouzos, M.L., et al.: On data banks and privacy homomorphisms. *Foundations of secure computation* **4**(11), 169–180 (1978)
28. Ross, A., Othman, A.: Visual cryptography for biometric privacy. *IEEE transactions on information forensics and security* **6**(1), 70–81 (2010)
29. Schroff, F., Kalenichenko, D., Philbin, J.: Facenet: A unified embedding for face recognition and clustering. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 815–823 (2015)
30. Microsoft SEAL (release 3.7). <https://github.com/Microsoft/SEAL> (Sep 2021), microsoft Research, Redmond, WA.
31. Shashank, J., Kowshik, P., Srinathan, K., Jawahar, C.: Private content based image retrieval. In: 2008 IEEE Conference on Computer Vision and Pattern Recognition. pp. 1–8. IEEE (2008)
32. Singhal, A., et al.: Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.* **24**(4), 35–43 (2001)
33. Smart, N.P., Vercauteren, F.: Fully homomorphic simd operations. *Designs, codes and cryptography* **71**(1), 57–81 (2014)
34. Troncoso-Pastoriza, J.R., González-Jiménez, D., Pérez-González, F.: Fully private noninteractive face verification. *IEEE Transactions on Information Forensics and Security* **8**(7), 1101–1114 (2013)
35. Uludag, U., Pankanti, S., Jain, A.K.: Fuzzy vault for fingerprints. In: International Conference on Audio-and Video-Based Biometric Person Authentication. pp. 310–319. Springer (2005)
36. Upmanyu, M., Namboodiri, A.M., Srinathan, K., Jawahar, C.: Efficient biometric verification in encrypted domain. In: International Conference on Biometrics. pp. 899–908. Springer (2009)
37. Upmanyu, M., Namboodiri, A.M., Srinathan, K., Jawahar, C.: Efficient privacy preserving video surveillance. In: 2009 IEEE 12th international conference on computer vision. pp. 1639–1646. IEEE (2009)
38. Yao, A.C.C.: How to generate and exchange secrets. In: 27th Annual Symposium on Foundations of Computer Science (sfcs 1986). pp. 162–167. IEEE (1986)

A Complexity and Security Analysis

We first provide a theoretical complexity analysis to show the efficiency of CryptoMask. Then we show that CryptoMask is secure against a semi-honest adversary while assuming KG is fully trusted.

A.1 Complexity Analysis

In CryptoMask, communication overhead mainly comes from two parts. One is from CS, who sends all the encrypted distances to the verifier, which contains $O(Nm/d)$ communication cost. Another one is the result of the secure revealing process, which requires $O(ml)$ communication. We can obtain the overall communication complexity as $O(Nm/d + ml)$. The computation overhead is more complex. We set the computation for data encryption using HE as C_{en} , for homomorphic multiplication as C_{mul} , for homomorphic addition as C_{add} , for key switching as C_{sw} , for secure comparison as C_{com} and for secure **B2A** as C_{cov} . The overall computation overhead for the CS side is $O((Nm/d)(C_{com} + C_{add} + C_{sw}) + m(C_{com} + C_{cov}))$ and for the verifier side is $O(C_{en} + m(C_{com} + C_{cov}))$.

A.2 Security Analysis

Privacy of Face Vector Matrix. In CryptoMask, all face vectors are encrypted by HE, and only the KG knows the secret key. Due to the semantic security of HE, neither CS nor the verifier learns sensitive information about the underlying encrypted face vector; thus, the privacy of the face vector is always maintained.

Now we show CryptoMask only reveals a face recognition result to the verifier and nothing else to either party. This is argued as regards to a corrupted CS and a corrupted verifier, respectively. Note we only provide the security of the HE-based part as the simulation of the comparison/**B2A** protocols can be implemented in the existing ways.

Corrupted CS. We first demonstrate the security against a semi-honest CS. Intuitively, the security against a semi-honest CS comes from the fact that the CS's view of the execution includes only ciphertext, thus reducing the argument to the semantic security of HE. We now give the formal argument.

Let \mathcal{A} be the semi-honest CS in the real protocol. We construct a simulator \mathcal{S} in the ideal world as follows:

1. At the beginning of the protocol execution, \mathcal{S} receives the input \mathbf{A} from the environment \mathcal{E} and also receives the public key pk and the vector length d . The simulator sends \mathbf{A} to the trusted party.
2. Start running \mathcal{A} on input \mathbf{A} . Next, \mathcal{S} computes and sends a ciphertext ct , which encrypts a d dimensional vector $\mathbf{0}$ to the CS under the public key pk .
3. Output whatever \mathcal{A} outputs.

We argue the above simulated view is indistinguishable from real protocol execution. Using the fact that \mathcal{A} is semi-honest, at the end of the protocol in the real world, the verifier obtains the encryption of $\mathbf{A} \cdot \mathbf{b}$ where \mathbf{b} is the verifier's queried face image. Since \mathcal{S} is semi-honest, this also holds in the ideal world. Since $\mathbf{A} \cdot \mathbf{b}$ is a deterministic function, the joint distribution of the verifier's output and the adversary's output decomposes. Thus, it is sufficient to show that the simulated view from \mathcal{S} is computationally indistinguishable from the real view from \mathcal{A} .

The view of \mathcal{A} in the real world contains one part: the encrypted face image ct from the verifier. When interacting with the simulator \mathcal{S} , adversary \mathcal{A} sees an encryption of $\mathbf{0}$. Security follows immediately by the semantic security of the BFV scheme.

Corrupted Verifier. We now prove the security against a semi-honest verifier. We construct a simulator \mathcal{S} in the ideal world as follows:

1. At the beginning of the execution, \mathcal{S} receives the input \mathbf{b} from the environment \mathcal{E} and also receives the BFV key pairs (pk, sk) and the matrix size m, d . The simulator sends \mathbf{b} to the trusted party.
2. Start running \mathcal{A} on input \mathbf{b} . Next, \mathcal{S} computes and sends ciphertexts c_i which is the encryption of an $m \times d$ matrix filled by some random values to the verifier under the public key pk_v .
3. Output whatever \mathcal{A} outputs.

At the end of face recognition, CS has no output. Thus, to show the security against a semi-honest verifier, it suffices to show that the output of \mathcal{S} is computationally indistinguishable from the output of the adversary \mathcal{A} . Now we show the view of simulator \mathcal{S} in the ideal world is computationally indistinguishable from the view of the adversary \mathcal{A} in the real world.

The view of \mathcal{A} in the real world contains one part: the encrypted face database $\{c_1, \dots, c_n\}$ from CS. When interacting with the simulator \mathcal{S} , adversary \mathcal{A} sees the encryption of random values. Security follows immediately by the semantic security of the BFV scheme.

B Accuracy

We report the results of face recognition on dataset LFW for state-of-the-art face representation FaceNet in Table 2. We only test face templates of 128-D. For more results on different representations, we refer to [3], which is also constructed on BFV. Same as [3], we report true acceptance rate (TAR) at three different operating points of 0.01%, 0.1% and 1.0% false accept rates (FARs). We first report the performance of the unencrypted face images. We treat these outputs as a baseline to compare. To evaluate encrypted face images, we consider four different quantization for each element in facial features. Specifically, we employ precision of 0.1, 0.01, 0.0025 and 0.0001. It shows that the performance of most given precision is competitive with the performance conducted from the raw data. We conclude that CryptoMask working over HE and MPC can perform as well as the one working over raw data.

Table 2: Face recognition accuracy for LWF dataset (TAR @ FAR in %)

| Method | 128-D FaceNet (Accuracy) | | |
|-----------------------------|--------------------------|-------|-------|
| | 0.01% | 0.1% | 1% |
| No FHE | 98.70 | 98.70 | 98.70 |
| FHE(1.0×10^{-4}) | 98.70 | 98.70 | 98.70 |
| FHE(2.5×10^{-3}) | 98.70 | 98.70 | 98.70 |
| FHE(1.0×10^{-2}) | 98.76 | 98.76 | 98.76 |
| FHE(1.0×10^{-1}) | 98.50 | 98.50 | 98.50 |