POP-HIT: Partially Order-Preserving Hash-Induced Transformation for Privacy Protection in Face Recognition Access Control

Yatish Dubasi(✉), Qinghua Li, and Khoa Luu

University of Arkansas, Fayetteville, AR 72701, USA
{yrdubasi,qinghual,khoaluu}@uark.edu

Abstract. Server-based face recognition access control (SFRAC) is a widely used biometric authentication method where a camera captures a user's face image and sends the face information to a backend server to determine an authentication result. In existing systems, user face images or raw facial features are sent to the backend server, which induces privacy concerns. This paper proposes a method to protect user face privacy in SFRAC systems, such that user face images or raw facial features are never sent to the server, but the system can still do face-based access control. The method uses a novel partially order-preserving hash-induced transformation (POP-HIT) to perform feature transformation on facial embeddings. POP-HIT maps a user's facial features to a transformed space in order to preserve privacy. Authentication is performed by checking the transformed facial embedding of the user's face image captured at the time of authentication against the user's pre-stored transformed facial embeddings at the server, based on similarity metrics or machine learning methods. Since the server only sees transformed facial embeddings, which cannot be used to recover the facial image, facial privacy is protected. Analysis and experimental evaluations show that our method has a high authentication accuracy, and it achieves privacy protection without being vulnerable to impersonation attacks.

Keywords: Facial recognition · Access control · Privacy

1 Introduction

Facial recognition technology is a growing field. There are more and more cloud services on facial recognition. Amazon's Rekognition can be used as a cloud-based system to recognize people [1,2], along with other similar commercial services [3,4]. Server-based face recognition access control (SFRAC) is a method of biometric authentication that relies on a server or the cloud to compute and/or determine the authentication result based on user face recognition. A common use of SFRAC is building access control, which is already widely used in commercial buildings and also increasingly deployed in residential buildings. There are commercially available door locks that utilize SFRAC [5,6].

H. Duan et al. (Eds.): SecureComm 2023, LNICST 568, pp. 80–98, 2025.
https://doi.org/10.1007/978-3-031-64954-7_5

However, there are rising privacy concerns with server-based facial recognition: users might not want pictures of their faces to be sent to backend servers or the cloud [2]. A common approach to combat this privacy concern is to only send the facial embedding of the image to the server. A facial embedding is a numerical representation of features extracted from an image of a face. Some random noise could even be added to the facial embedding vectors to further protect privacy. However, research has shown that it is possible to accurately reconstruct an image of a face given a facial embedding even if it has a small amount of noise [7]. Existing privacy preserving SFRAC methods [8–11] rely on homomorphic encryption. However, they have high computation overhead. Additionally, these methods are not very flexible and usually can only support certain feature vector representations and distance metrics that they are designed for.

In this paper we propose a new method to preserve user privacy in SFRAC. From the access control perspective, the server only needs to know that the face being authenticated belongs to the claimed user; however, the server does not have to know exactly what the face looks like. Motivated by this observation, our goal is that the system can still do face-based authentication but the server/cloud never sees or is able to recover user face images. To achieve this goal, we design a novel partially order-preserving hash-induced transformation (POP-HIT) that maps the features of a user's facial embeddings to a transformed space. The server will store POP-HIT embeddings from each user during the enrollment phase. At the time of authentication, a user's facial image is taken, the facial embedding is extracted from the image (referred to as *authentication embedding* for ease of presentation), transformed with POP-HIT, and then sent to the server for authentication. The server can either use some similarity metrics to compare the authentication embedding and the user's pre-stored embeddings to determine if the user is the claimed one, or train a classification model based on users' transformed facial embeddings and input the authentication embedding to the model to see if it belongs to the claimed user. In the entire process, the server only sees transformed facial embeddings. The transformation introduces sufficient noise to facial embeddings, and hence our method can mitigate facial embedding inversion attacks and preserve privacy.

This paper's main contributions are summarized as follows.

- We propose a novel facial feature transformation method, POP-HIT. Its partially order-preserving property ensures that, in the transformed space, the facial embeddings of the same user stay close to each other while the facial embeddings of different users stay far away from each other. This allows any distance-based similarity metrics or other machine learning methods to be used in the transformed space for classifying the facial embeddings of different users. On the other hand, the transformation process introduces random noises to facial embeddings, such that an adversary cannot recover the user's face from the transformed embeddings.
- We design a generic protocol framework that utilizes POP-HIT to preserve user face privacy in SFRAC systems. This framework is compatible with various facial classification methods at the server side.

– We perform comprehensive evaluations on the effectiveness and performance of the solution over real-world datasets, and compare it with a state-of-the-art solution.

The remainder of this paper is organized as follows. Section 2 discusses related work on protecting user privacy in SFRAC. Section 3 gives an overview of our system models and solution. Section 4 presents POP-HIT and our proposed privacy-preserving framework for SFRAC. Section 5 describes implementation details. Section 6 presents evaluation results. Lastly, Sect. 7 concludes this paper.

2 Related Work

There exists some work in privacy protection in face recognition access control with feature encryption. References [8–10] propose privacy-preserving solutions utilizing fully homomorphic encryption (FHE) schemes. They calculate the euclidean distance of encrypted facial feature vectors and then compare them to a threshold to authenticate users. Similarly, references [12,13] propose a privacy-preserving scheme utilizing a multiplicative homomorphic encryption scheme. It has multiple rounds of communication between the client and server to compute a sum of products between the face feature vector and each user's classifier parameter vector, and then it compares that resulting score against a threshold to determine the authentication result. Reference [11] proposes two privacy preservation schemes that utilize the Paillier encryption scheme. Their first method is similar to the previously mentioned methods except that it relies on oblivious transfer of server records to perform operations and authenticate users on the client device. Their second method utilizes deep neural networks on the client side to extract the features from face images; these extracted feature vectors are encrypted and sent to the server where it computes the hamming distance between the authentication face vector and the pre-stored face vector, and then compares that distance score to a threshold to determine the authentication result. Our proposed method is much different from these prior works' methods as we utilize feature transformation instead of feature encryption. Feature encryption has high computation overhead. The feature encryption methods are also limited in scope and flexibility as only a few distance metrics and facial embedding structures can be compatible, whereas our feature transformation method is very flexible and has significantly faster computation.

3 Overview

The system has a number of users, a number of cameras, and a server in the backend/cloud. Users can communicate with the server and cameras via their mobile device (e.g., smartphone). The cameras can be those commonly seen in building access systems. Without loss of generality, we focus on one user and one camera together with the server when describing our solution (as illustrated in

Fig. 1). The system works in two phases. In the *enrollment phase*, each user registers with the server. The user generates a number of facial images of themselves, extracts the facial embeddings with a facial recognition model, transforms them using POP-HIT, and sends the transformed embeddings to the server. The server stores all users' transformed embeddings. In the *authentication phase* (i.e., when a user wants to access a protected building), the user sends an access request to the server via their mobile device. This request gives the server information regarding the claimed user identity as well as what the user is trying to access. The server sends a request to the camera to capture the user's face. The camera extracts the facial embedding and sends it to the user. The user transforms it using POP-HIT, and sends the transformed embedding to the server. The server checks the received embedding against the stored facial embeddings of users to verify whether the user is the claimed one. If the authentication is confirmed, the access request is approved; otherwise, the access request is denied.

Our security model is similar to existing work in this topic [8,9]. We assume that the server and the camera are honest-but-curious. They follow our protocol but tries to access or recover users' face images from their protocol transcript. Our privacy goal is to not leak any raw facial feature data to the server, and prevent the server from recovering facial images. Similar to existing work, we assume the server and the camera do not collude. We assume that the user enrollment process is done in a trusted way (e.g., in an office setting), so that a user's own face images are used to enroll. In the authentication phase, users are not trusted in that one user might try impersonating another user.

4 Privacy-Preserving SFRAC with POP-HIT

This section presents our solution. We first describe the POP-HIT scheme. Then we present the protocol framework's enrollment phase and authentication phase in detail. After that, we describe how to set parameters and perform security analysis. Notations used in this paper are shown in Table 1.

4.1 POP-HIT

Our partially order-preserving hash-induced transformation (POP-HIT) works as follows and is exemplified in Fig. 2. At the high level, the original facial feature space (i.e., the possible values of a facial embedding) is divided into a number of ordered partitions. The transformed space is also divided into the same number of ordered partitions, but with some empty gaps between neighboring partitions. The empty gaps or "white spaces" will aid in privacy preservation and mitigation against impersonation attacks. The partitions between the two spaces are order-preserving one-to-one mapping; i.e., facial feature values in the first partition of the face vector space transform to values in the first partition of the transformed space, values in the second partition of the face vector space transform to values in the second partition of the transformed space, and so on. Under this mapping, facial features in the same partition of the original space always map to the

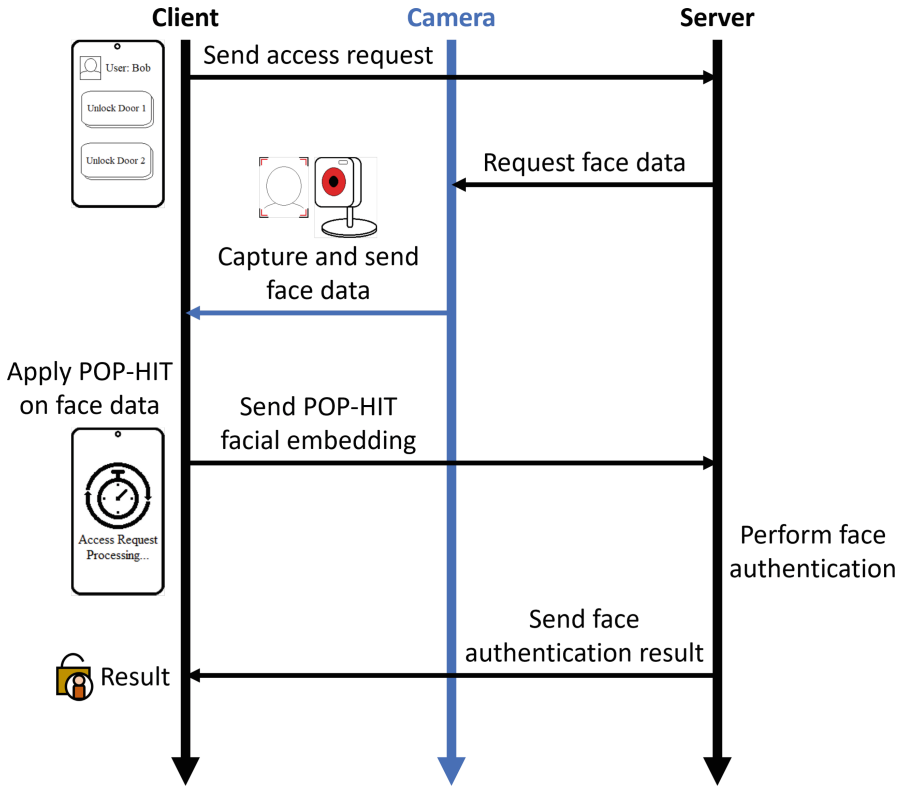

Fig. 1. System Model and the authentication phase workflow

same partition in the transformed space. However, facial features within the same partition in the original space might not preserve their order after the mapping. Their positions inside the transformed partition are determined by a pseudorandom cryptographic hash function, which does not necessarily preserve their order as in the original space.

When applying POP-HIT to a facial embedding, we will independently apply the transformation to each feature in the facial embedding vector; the result is a POP-HIT embedding, a vector of all the transformed features.

More specifically, the face vector space is derived by scaling the values in the facial embedding vectors between 0 and 1. Min max scaling is used to normalize facial embeddings to the same space regardless of which model (e.g., VGGFace [14], VGGFace2 [15], FaceNet [16]) is used for generating the facial embeddings. The scaler is initialized with a large amount of random facial embeddings. The face vector space will be split into n partitions, so each partition size will be $1/n$.

The transformed space consists of mainly two parts: the partitions that facial features will be mapped to, and the white spaces between partitions that do not

Table 1. Main Notations

u	user identity
k	transform secret
$\mathbf{x}$	facial embedding
$POP - HIT(\mathbf{x})$	transformed embedding
T	authentication threshold
t_b	basis threshold
B	max initial base value
b	number of bits to represent initial base value
L	max white space value
w	number of bits to represent each white space value
W	white space count, which equals to $n - 1$
M	partition size
n	partition count
s_m	minimum transformed space size
s_a	average transformed space size
i_{addit}	increment additive estimate
d_{dscale}	decrement downscale estimate
d_{deduct}	decrement deductive estimate

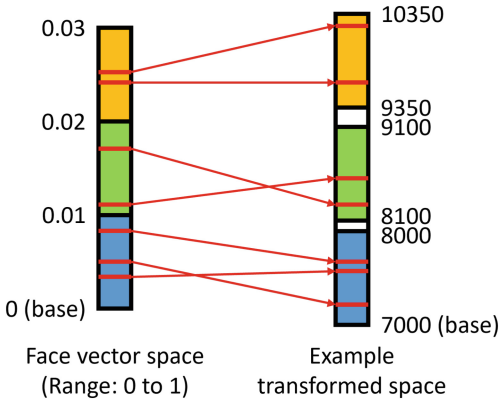

Fig. 2. POP-HIT Example

get any values mapped there. The transformed space has n partitions, as well as $n-1$ white spaces. Let M denote the partition size in the transformed space. Partition size will be the same for all users. The size of the white spaces can range from 0 to L. Each user has a unique *transform secret*, and this secret determines the white space sizes for their transformed space. Even for the same user, the sizes of different white spaces are different too.

The transform secret is structured as follows: the first b bits represent the base or start of the transformed space allowing the base to range from 0 to $B = 2^b$, and it is followed by W w-bit segments, where each w-bit segment represents the size of one white space. Each user generates their transform secret as random bits. Since partition size is fixed and the same for all users, we do not need to store that information in the transform secret. This transform secret will allow a user's device to fully construct their unique transformed space. This constructed space can be used to apply POP-HIT. To construct the transformed space, we can start with the value for the base of the transformed space, which can be retrieved from that first b bits of the transform secret; then we add the partition size to that base, and these two values represent our first partition's base and end; then we add the first white space size to the previous value to get the base of the next partition, and the white space size is represented by the next w bits of the transform secret; we repeat this process to derive all the transformed partitions.

The transformed value inside a partition can be determined with the following formula: $(\text{hash}(x_i) \bmod \text{M}) + (\text{base}(t_i))$, where x_i is a feature value belonging to the i^{th} partition in the face vector space, and $\text{base}(t_i)$ is the base value of the i^{th} partition in the transformed space.

4.2 Enrollment

The enrollment process of a user is illustrated in Fig. 3.

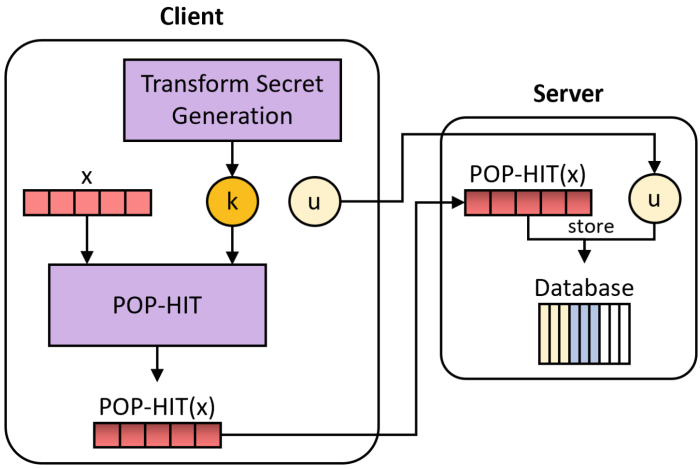

Fig. 3. Enrollment

1. Each user generates their own unique transform secret, k, for our POP-HIT method. The user stores k.
2. The user collects a number of their own face images.
3. The user gathers the facial embeddings, $\mathbf{x}$, by using a facial recognition model such as VGGFace, VGGFace2, FaceNet, or FaceNet512 on the collected images.
4. The user scales all the embeddings, such that all the values are between 0 and 1.
5. The user transforms these scaled embeddings to the transformed space using their transform secret, k, with POP-HIT.
6. The user labels these transformed embeddings, $POP - HIT(\mathbf{x})$, with the appropriate user identity, u.
7. The user sends their identity u and $POP - HIT(\mathbf{x})$ to the server.
8. The server stores u and $POP - HIT(\mathbf{x})$ in a database.

4.3 Authentication

The authentication process is illustrated in Fig. 4.

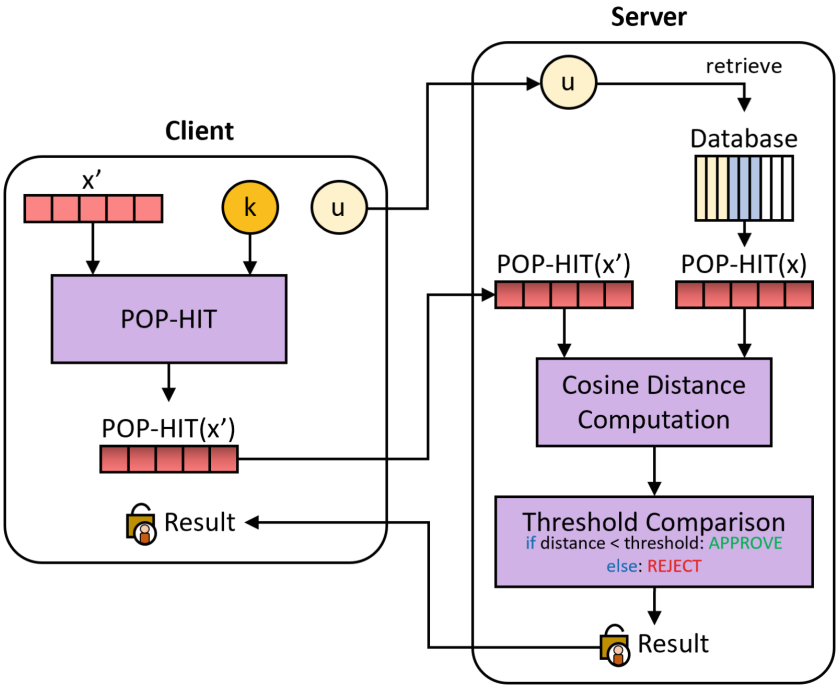

Fig. 4. Authentication

1. A user sends an access request to the server, claiming their identity u and the target (building) to access.
2. The server asks the camera to take a photo of the user. The facial embedding, $\mathbf{x}'$, is extracted by the camera, and its feature values are scaled between 0 and 1. Then the embedding is sent to the user.
3. A malicious user might not use the image captured by the camera to authenticate, but try to impersonate another user by using an image of that user obtained elsewhere (e.g., from the Internet). To mitigate such impersonation attacks, the camera randomly picks two partitions in the face vector space that each have multiple feature values in the embedding vector. Two features are randomly selected from each of the two partitions. A feature from one partition is paired with a feature from the second partition. As a result, we now have two pairs of features. For example, suppose we gathered the following pairs of features (0.328, 0.114) and (0.321, 0.113), where 0.328 and 0.321 belong to one partition, and 0.114 and 0.113 belong to the other. Although the camera does not have the user's transform key and hence does not know the exact transformed values for these four features, it can derive the indexes of these four features in the transformed space. Actually, it can derive the indexes of all features in the transformed embedding by first mapping them to partitions, then calculating each feature's index within its transformed partition through $hash(x)\ mod\ M$, and lastly ordering all the features in the transformed space. After the features are ordered, their indexes become known. The two feature pairs, the indexes of the four selected features, and the claimed user identity u are sent to the server. Since only four randomly picked feature values are exposed to the server, the server cannot recover the user face from them, and hence privacy is not violated. Step 5 will further describe how these values will be used to mitigate impersonation attacks.
4. The user, via their mobile device, transforms the scaled embedding using their transform secret, k, with POP-HIT, and then sends the transformed embedding $POP-HIT(\mathbf{x}')$ together with the user identity u directly to the server.
5. The server first verifies that the received transformed embedding was indeed generated from the picture taken by the camera. If the verification fails, the server discards this request. Let $(x1, x2)$ and $(x3, x4)$ denote the two pairs of features received from the camera, where $x1$ and $x3$ are from one partition and $x2$ and $x4$ are from the other partition. Although the server does not know the exact structure of the transformed space for this user, the distance between the transformed values of $x1$ and $x2$ could be written as $(hash(x1) - hash(x2))\ mod\ M + C$, where C is an unknown value dependent on the two partitions and the white spaces between them; the distance between the transformed values of $x3$ and $x4$ could be written as $(hash(x3) - hash(x4))\ mod\ M + C$, where C is the same unknown value. The server can calculate the difference between these two distances, which will be $d = (hash(x1) - hash(x2) - hash(x3) + hash(x4))\ mod\ M$. Since the server received the indexes of these four features in the transformed embedding from the camera, when the server receives the transformed embedding

from the user, it can find the corresponding four transformed features based on the indexes. Then it can calculate the two distances and finally the difference between the two distances, denoted by d'. If the two differences d and d' match, it means the user used the picture captured by the camera; if they do not match, this is deemed as impersonation attack. If the user attempts to use a fake image (i.e., not the one captured by the camera) to authenticate, since the user does not know which four features were chosen by the camera, the chance of success is only $1/M$, which is negligible considering that M can be set a large value.

6. The server computes cos_dist($POP - HIT(\mathbf{x})$, $POP - HIT(\mathbf{x}')$), where $POP - HIT(\mathbf{x})$ is retrieved from the server's database. Cosine distance between two vectors can be computed as follows:

$$cos\_dist(A, B) = 1 - \frac{A \cdot B}{\|A\|_2 \cdot \|B\|_2}$$

   The cosine distance metric is chosen because it is commonly used and produces the best results in state-of-the-art algorithms in facial recognition systems including VGGFace, VGGFace2, and FaceNet. Other distance metrics could also be used based on application needs.
7. The server can then compare this distance to a certain threshold T. If the distance is smaller than the threshold, the user authentication succeeds; otherwise, the authentication fails.
8. Alternative to the distance metric-based authentication in Steps 6 and 7, the server can also use machine learning methods to check if the transformed embedding received from the user really belongs to the claimed user identity, as discussed in Sect. 4.4 and 6.

4.4 Discussions

We discuss how the POP-HIT parameters affect the accuracy of face-based authentication and the privacy of user faces. The minimum transformed space size (s_m) refers to the transformed space size if the max white space values were set to zero. Similarly, the average transformed space size (s_a) refers to the transformed space size when the white space values were all the average value. If the maximum initial base value B is significantly large in comparison to s_a, it can reduce accuracy. Similarly, if the maximum white space size L is significantly large in comparison to s_m, it can reduce accuracy too. These parameters, B and L, can reduce accuracy because they can introduce significant amount of randomness into the transformed space. If both B and L are small or average in comparison to s_m and s_a, accuracy will be better as there will be less randomness. B does not directly impact privacy, whereas the larger L is the better privacy can be achieved. This is because the larger L is, the more amount of noise that is added into the POP-HIT embeddings. The larger W (which means the more partitions) is, the higher the accuracy will be. The smaller W is (which means the less partitions), the better privacy preservation will be. This is because the

fewer number of partitions there are, the more random POP-HIT will be. Lastly, if M is not too large or too small in comparison to s_m, the better accuracy and privacy will be. If M is significantly large, it will have more randomness when mapping within partitions. Whereas, if M is significantly small, it will have very little randomness and many collisions when mapping within partitions.

Our proposed framework is flexible and compatible with machine learning classification methods. To use machine learning classification at the server side, users would need to send enough POP-HIT embeddings for training in the enrollment phase. In the authentication phase, the server can simply use POP-HIT embeddings as inputs into their trained model, and the authentication results can be determined based on the model's prediction vectors. We provide evaluation results for this option as well in Sect. 6.

4.5 Distance Threshold Engineering

Setting a proper threshold T for the cosine distance in the authentication phase is important to the performance of the solution. Here, we analyze how POP-HIT parameters impact the cosine distance and in turn the threshold. First, assuming that we are using VGGFace, VGGFace2, FaceNet, or FaceNet512. The threshold that is suggested by Deepface when using these models with the cosine distance metric is 0.4 [17,18]. This threshold is intended for raw facial embeddings. In our solution, however, the scaling of features to be between 0 and 1 impacts the cosine distance, as features that are initially negative will become positive after this scaling. According to our experiments, the cosine distance decreases by about an order of magnitude after the scaling. Therefore, we set our basis threshold t_b to 0.04. Since other parameters of POP-HIT affect the cosine distance too, we need to do further adjustment to the threshold. There are four parameters that can be customized for POP-HIT: max initial base value, max white space value, white space count, and partition size. All users will have the same white space counts and partition sizes. The initial base values and individual white space values will be unique to each user's transform secret and will be upper bounded by the max initial base value and max white space value parameters respectively. If we assume that the max initial base and white space value parameters were set to zero, therefore not allowing any uniqueness among users (everyone will have the same transformed space), the best threshold will remain at 0.04 without regard to the remaining two parameters. This is because with just the white space count and partition size, the transformation will act as a direct scaling of the features which does not impact the cosine distance. The max initial base and white space value parameters are what can cause changes to the cosine distance and in turn to the threshold. Increasing the max white space value will increase the cosine distance values, as the parameter will enlarge the transformed space. Increasing the max initial base value will reduce the cosine distance values, as the parameter will shift the entire transformed space without enlarging the space. In order to compute a proper threshold, we will estimate the increase and decrease to the threshold; then we can apply them over the basis

threshold of 0.04. Based on our experimentation, we found it best to approximate the increase and decrease to the threshold as follows. We approximate the increase to the threshold, i_{addit}, based on the average white space value in comparison to s_m. Similarly, we approximate the decrease to the threshold, d_{dscale} or d_{deduct}, based on the average initial base value in comparison to s_a. Based on our experimental observations, the calculation of the adjusted threshold T should follow the steps below. The formulas were derived from empirical observations instead of theoretical deductions (the problem is very hard to approach theoretically), but as it can be seen in Sect. 6, they work well in our experiments.

$$
\begin{aligned}
t_b &= 0.04 \\
n &= W + 1 \\
s_m &= n \cdot M \\
s_a &= s_m + \frac{L}{2} \cdot W \\
i_{addit} &= \frac{L}{2} / s_m / n \cdot 10^{-1} \\
d_{dscale} &= \begin{cases} \frac{B}{2} / s_a \cdot 8, & \text{if } (\frac{B}{2} / s_a \cdot 8) \geq 1 \\ 1, & \text{otherwise} \end{cases} \\
d_{deduct} &= \begin{cases} \frac{B}{2} / s_a \cdot 8 \cdot 10^{-2}, & \text{if } (\frac{B}{2} / s_a \cdot 8) < 1 \\ 0, & \text{otherwise} \end{cases} \\
T &= \frac{t_b + i_{addit} - d_{deduct}}{d_{dscale}}
\end{aligned}
$$

4.6 Security Analysis

Privacy. Our method can preserve user's facial privacy against the server or other eavesdroppers. This is because the method never sends raw face data, and all of the facial embeddings are transformed to mitigate facial reconstruction attacks, as shown in Sect. 6. The reconstruction attacks will fail because POP-HIT adds significant noise to the facial embeddings to where the reconstructed face will have very different facial features compared to the original.

Additionally, provided with a user's POP-HIT embedding, an attacker cannot infer or estimate the original facial embedding without knowledge of the user's transformed secret. This is due to the use of the one-way cryptographic hash function in POP-HIT. Also, it is difficult to brute force the base and white space values of the user's transformed space, since the values can be very large and there are too many possibilities.

Impersonation. Impersonation attacks will fail. This is due to the verification step performed by the server to ensure that the transformed embedding received from the user was indeed generated based on the image captured by the camera. As described in Sect. 4.3, if an attacker sends a transformed embedding of a different feature vector than the camera's extracted vector, the server will be able to detect this.

5 Implementation

We chose to work with the Labeled Faces in the Wild (LFW) dataset [19,20], which is a publicly available benchmark for facial authentication. The LFW dataset contains over 13,000 images of over 5,000 people's faces from the web. We had created a modified version of the LFW dataset (Trimmed_LFW). To create the Trimmed_LFW dataset, we gathered 150+ users in the LFW dataset that contained at least two images per user. Overall, the Trimmed_LFW dataset contained 737 images. We implemented our proposed method along with baseline methods for comparison. For the implemented methods, we chose two unique images for each user from the Trimmed_LFW dataset. One image is used to enroll the user and the other is used to test the authentication.

When implementing our method, we used the PyTorch [21], torchvision [22], and facenet-pytorch python modules to gather FaceNet512 facial embeddings. FaceNet512 was chosen because it is one of the top state-of-the-art face recognition models. We utilize the cosine distance metric in this implementation. The cosine distance metric is chosen because it is one of the best performing distance metrics in the facial recognition domain. The authentication threshold is based on the suggestion of Deepface, 0.4 for FaceNet with the cosine distance metric, and adjusted based on the methods described in Sect. 4.5. Besides cosine distance-based verification, we also implemented a classification approach with POP-HIT to demonstrate our framework's flexibility to support machine learning models. In particular, we used linear support vector machines (SVMs) for this purpose.

We also implemented a state-of-the-art method [9] for comparison. Their method relies on feature encryption with FHE as described in Sect. 2. We utilized reference [23] to help implement their method. We use the Deepface python module to gather FaceNet facial embeddings. FaceNet was chosen because it is also one of the top state-of-the-art face recognition models, and since it only extracts 128 features, it will minimize feature encryption time (thus, the performance result will not be biased against the baseline method). We utilize TenSEAL [24] to perform FHE for this method. This method uses the euclidean distance squared metric, as it can be fully computed on the ciphertext when using FHE.

For comparison, we also implemented a traditional SFRAC method that does not aim to preserve privacy. This method simply extracts FaceNet512 facial embeddings from the client and sends it to the server. The server computes a cosine distance between the test and pre-stored facial embeddings. This distance is compared to a threshold to determine the authentication result. As we can observe, this method does not utilize feature encryption or transformation.

6 Evaluation Results

The experiments were run on a computer with an Intel® Core™ i9-10900K 10-Core Processor running at 3.7 GHz using 16 gb of ram, running Windows 10 Enterprise version 22H2.

By default, we set the number of partitions $n = 100$, the transformed partition size $M = 1e + 17$, the maximum size of white space $L = 16,383$ (representable by 14 bits, i.e., $w = 14$), and $b = 24$ which means $B = 16,777,215$.

6.1 Enrollment and Authentication Times

We evaluate the enrollment and authentication times in total after the client sends a POP-HIT embedding to the server. First, we timed how long it takes for the server to enroll users when provided POP-HIT, encrypted, or raw facial embeddings for our proposed method, FHE method, and traditional method respectively. Similarly we timed how long it takes for the server to authenticate users when provided access requests with the appropriate type of embedding for each method. Lastly, we timed the client side operations for each method. The enrollment and authentication timing results are shown in Fig. 5. Here, "POP-HIT" refers to our proposed method; "FHE" refers to the FHE method [9]; "Traditional" refers to the traditional method without privacy protection.

We observed that the average enrollment time per user is approximately 42.4, 33.5, and 31.8 microseconds for our proposed method, FHE method, and traditional method respectively. The enrollment times are very short for all of the methods. We observed that the average authentication time per user is approximately 78.6, 17728.5, and 80.6 microseconds for our proposed method, FHE method, and traditional method respectively. The authentication times are very short for our proposed method and traditional method, but the FHE method's authentication time is significantly larger. We observed that the average client side time per user is approximately 1.5, 1.2, and 1.5 s for our proposed method, FHE method, and traditional method respectively. The client side times are reasonably short for all of the methods.

6.2 Accuracy of Authentication

We evaluate the accuracy by observing precision, recall, and F1 scores. True positives occur when a legitimate access request is permitted. False negatives occur when a legitimate access request is rejected. False positives occur when an impersonated access request is permitted. True negatives occur when an impersonated access request is rejected. Impersonated access requests attempt to utilize a facial embedding not belonging to the claimed identity to pass authentication.

When there were 150 users enrolled in the system, the FHE method had a 98.5% precision, 88.0% recall, and 92.9% F1 score. The traditional method had a 100% precision, 98.6% recall, and 99.3% F1 score. Our proposed method had a 98.6% precision, 99.3% recall, and 99.0% F1 score (parameters: $B = 10000$, $L = 1000$, $W = 99$, and $M = 1000$). The precision is very good for all of the three methods. Our method and the traditional method have very good recall and F1 scores, but the FHE method's recall and F1 score is considerably smaller.

Next, we explored the effect of setting a proper threshold T on the accuracy of authentication. Specifically, we compare the performance of using our threshold setting method described in Sect. 4.5 and that of using the basis threshold

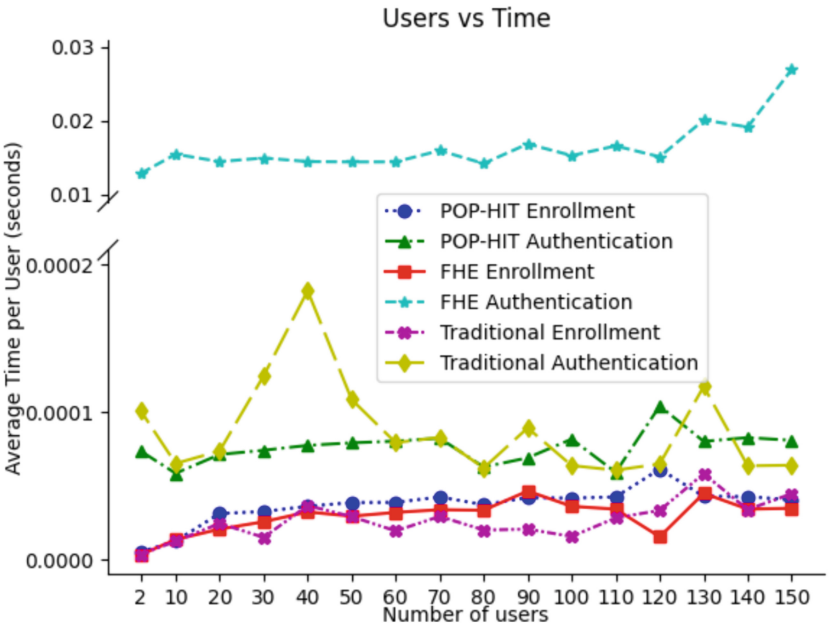

Fig. 5. Enrollment and Authentication Times

$t_b = 0.04$. The results are shown in Fig. 6, where "Basis" refers to the basis threshold and "Estimated" refers to the estimated authentication threshold, T, in our approach. In these experiments, we set $L = 1000$, $W = 99$, and $M = 1000$. The estimated authentication thresholds were 0.037, 0.031, 0.029, 0.021, and 0.016 respectively when the max initial base value B is at 10,000, 30,000, 50,000, 70,000, and 90,000. We observed that utilizing our estimated threshold consistently has better or similar performance than utilizing the basis threshold.

In order to demonstrate our proposed method's flexibility, we trained linear Support Vector Machines (SVMs) on POP-HIT embeddings. We enrolled 60 users who had 20 or more images in the LFW dataset. Five images of each user were kept for the testing dataset, while the remaining was used for training the SVMs. When utilizing the default parameters, we observed a 100% precision, 99.7% recall, and 99.9% F1 score on the training data, and we observed a 100% precision, 96.0% recall, and 97.9% F1 score on the testing data. When utilizing different sets of POP-HIT parameters that are similar to the default, we observed that the SVMs still had similar performance.

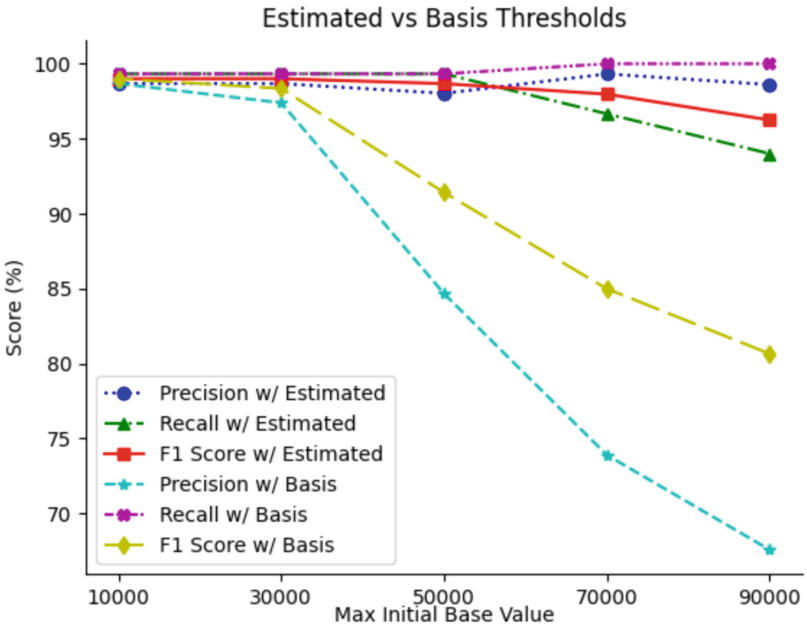

Fig. 6. Estimated and Basis Threshold Accuracy Results

6.3 Privacy Preservation

In order to evaluate that privacy is preserved by our proposed method and not by the traditional method, we used the 5 Celebrity Faces Dataset (5CF) to conduct empirical tests [25]. This test will also demonstrate the tradeoff between privacy preservation and accuracy. The 5CF dataset is a small dataset of face photos of five different celebrities. Each celebrity has at least 19 images. Overall, the 5CF dataset contained 118 images. We selected one image per user to conduct our tests. We performed facial reconstruction for two users in the 5CF dataset by using POP-HIT embeddings as well as the raw embeddings to see whether privacy is protected in our method and the traditional method. We implemented the reconstruction attack proposed in [7]. For our method, in order to convert from the transformed space to a normal face vector space for facial reconstruction, we scale the POP-HIT embedding values between zero and one, and then re-scale the values to be in the normal face vector space. *It is important to note that an attacker would require thorough knowledge regarding a victim's transformed space in order to conduct such an attack against our proposed method, and gaining such knowledge is difficult as analyzed in Sect.* 4.6. But for the purpose of performing face reconstruction tests, we assume that an attacker has such knowledge to conduct this attack. We exclude the FHE method [9] from these tests because it is based on feature encryption and should preserve privacy as long as the private key is not leaked.

We performed facial reconstruction on an example face of two celebrities from the 5CF dataset. The key parameters used for facial reconstruction are as follows: 400 iterations, simulated annealing in the reconstruction algorithm, and use of a pregenerated embedding to select the starting location. The results of performing facial reconstruction on the raw embeddings and converted POP-HIT embeddings can be seen in Fig. 7. We also show the accuracy of each set of POP-HIT parameters in Fig. 8, to demonstrate the privacy preservation and accuracy tradeoff. Here, we set $B = 10000$, $L = 1000$, and $M = 1000$ while changing W. The estimated authentication thresholds were 0.037, 0.037, 0.014, 0.007, and 0.003 respectively when W was 99, 24, 9, 4, and 1.

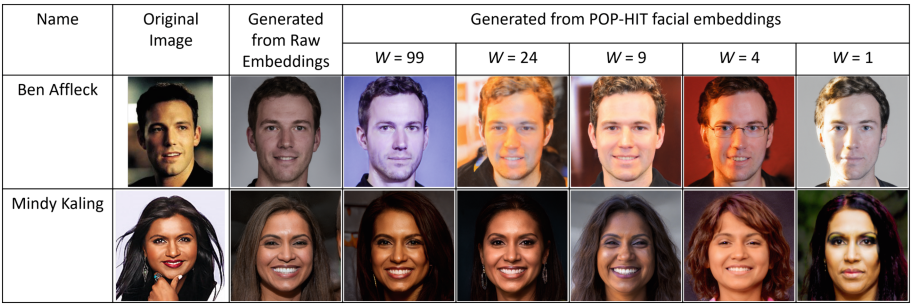

Fig. 7. Facial Reconstruction Effectiveness When W Changes

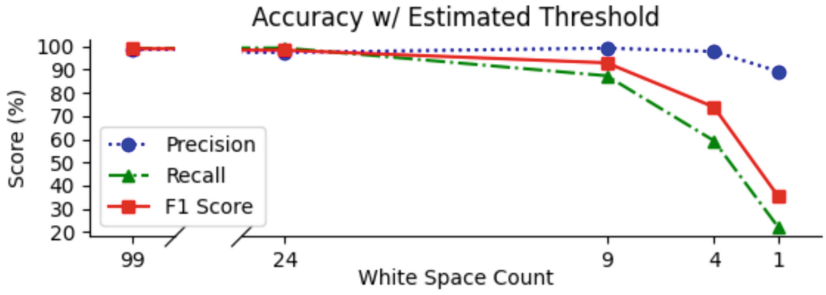

Fig. 8. Accuracy of Authentication When W Changes

We observed that the raw embeddings derived from the traditional method can be used to reasonably reconstruct a user's face. However, the POP-HIT embeddings derived from our method cannot be used to reconstruct a user's face, and therefore our method can preserve facial privacy. We also observed that, when W changes, there is a tradeoff between privacy preservation and accuracy. However, a good balance can be found based on deployment needs, e.g., $W = 24$ for this group of experiments.

6.4 Comments

We observed that our method has very strong and similar performance to the traditional method in authentication accuracy. However, our method also provides privacy preservation that is not offered by the traditional method. Our method considerably outperforms the FHE method in authentication times and accuracy; it only marginally has larger enrollment and client side times in comparison to the FHE method. In regards to privacy preservation, the FHE method has very strong privacy preservation. Our method also has very strong privacy preservation, because an attacker would need to know the victim's transform key to carry out a facial reconstruction attack against POP-HIT embeddings. Even in that case, we empirically showed that our method is still able to protect privacy due to the relatively significant amount of noise generated by POP-HIT. In regards to the flexibility of the methods in face authentication, the FHE method is limited to utilizing face recognition models that extract relatively small sized feature vectors, so as to minimize the feature encryption times. Additionally, the FHE method is limited to only a few distance metrics, as the distances need to be computed on the ciphertext to preserve privacy. However, our method is very flexible as POP-HIT can be applied on feature vectors derived from any face recognition model such as VGGFace, VGGFace2, and FaceNet; similarly any distance metric can be utilized in the transformed space. We also demonstrated that POP-HIT can be used with classification methods. Lastly, our method can be further fine tuned by modifying POP-HIT parameters to meet privacy, accuracy, and deployment needs.

7 Conclusion

We proposed a method to protect user privacy for face recognition access control. Our privacy-preserving SFRAC method utilizes POP-HIT, a novel feature transformation method that adds random noise in the transformation to preserve facial privacy, while still allows the server to perform face-based authentication and access control. Analysis and experimental results showed that our approach preserve privacy while maintaining high accuracy and performance in face-based authentication.

Acknowledgment. This material is based upon work supported in part by the National Science Foundation EPSCoR Cooperative Agreement OIA-1946391.

References

1. https://aws.amazon.com/rekognition/
2. https://arstechnica.com/tech-policy/2018/05/police-use-of-amazons-face-recognition-service-draws-privacy-warnings/
3. https://www.nec.com/en/global/techrep/journal/g18/n02/180208.html
4. https://www.intelli-vision.com/facial-recognition/
5. https://www.swiftlane.com/face-recognition-access-control/

6. https://www.swiftlane.com/blog/face-recognition-door-access-control/
7. Vendrow, E., Vendrow, J.: Realistic face reconstruction from deep embeddings. In: NeurIPS 2021 Workshop Privacy in Machine Learning (2021). https://openreview.net/forum?id=-WsBmzWwPee
8. 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)
9. Yang, Y., et al.: Design on face recognition system with privacy preservation based on homomorphic encryption. Wirel. Pers. Commun. **123**(4), 3737–3754 (2022)
10. Troncoso-Pastoriza, J.R., González-Jiménez, D., Pérez-González, F.: Fully private noninteractive face verification. IEEE Trans. Inf. Forensics Secur. **8**(7), 1101–1114 (2013)
11. Jin, X., et al.: Efficient blind face recognition in the cloud. Multimed. Tools Appl. **79**(17), 12533–12550 (2020)
12. Upmanyu, M., Namboodiri, A.M., Srinathan, K., Jawahar, C.V.: Efficient biometric verification in encrypted domain. In: Tistarelli, M., Nixon, M.S. (eds.) ICB 2009. LNCS, vol. 5558, pp. 899–908. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01793-3_91
13. Upmanyu, M., et al.: Blind authentication: a secure crypto-biometric verification protocol. IEEE Trans. Inf. Forensics Secur. **5**(2), 255–268 (2010)
14. Parkhi, O.M., Vedaldi, A., Zisserman, A.: Deep face recognition (2015)
15. Cao, Q., et al.: VGGFace2: a dataset for recognising faces across pose and age. In: 2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018), pp. 67–74. IEEE (2018)
16. 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)
17. Serengil, S.I., Ozpinar, A.: LightFace: a hybrid deep face recognition framework. In: 2020 Innovations in Intelligent Systems and Applications Conference (ASYU), pp. 23–27. IEEE (2020). https://doi.org/10.1109/ASYU50717.2020.9259802
18. Serengil, S.I., Ozpinar, A.: HyperExtended LightFace: a facial attribute analysis framework. In: 2021 International Conference on Engineering and Emerging Technologies (ICEET), pp. 1–4. IEEE (2021). https://doi.org/10.1109/ICEET53442.2021.9659697
19. Huang, G.B., et al.: Labeled faces in the wild: a database for studying face recognition in unconstrained environments. Technical report, 07-49. University of Massachusetts, Amherst (2007)
20. Huang, G.B., Learned-Miller, E.: Labeled faces in the wild: updates and new reporting procedures. Technical report, UM-CS-2014-003. University of Massachusetts, Amherst (2014)
21. Paszke, A., et al.: PyTorch: an imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems, vol. 32 (2019)
22. TorchVision maintainers and contributors. TorchVision: PyTorch's Computer Vision library (2016). https://github.com/pytorch/vision
23. Serengil, S.I.: Homomorphic Facial Recognition with TenSEAL (2021). https://sefiks.com/2021/12/01/homomorphic-facial-recognitionwith-tenseal/
24. Benaissa, A., et al.: TenSEAL: A Library for Encrypted Tensor Operations Using Homomorphic Encryption (2021). arXiv: 2104.03152 [cs.CR]
25. https://www.kaggle.com/datasets/dansbecker/5-celebrity-facesdataset