Efficient and Secure Template Blinding for Biometric Authentication

Siddhant Deshmukh
University of Florida
siddhantdeshmukh@ufl.edu

Henry Carter
Villanova University
henry.carter@villanova.edu

Grant Hernandez, Patrick Traynor, Kevin Butler
University of Florida
grant.hernandez@ufl.edu,
{traynor, butler}@cise.ufl.edu

Abstract—**Fingerprints as biometric authenticators are rapidly increasing in popularity, with fingerprint scanners available on many modern smartphones and laptops. Because these authenticators are non-revocable, special care must be taken to prevent leakage of the representative feature information of a user's fingerprint. While secure multiparty computation protocols have been designed to maintain fingerprint privacy during authentication, they do not protect the data stored on the authentication server. In this work, we develop a technique for blinding the stored biometric template such that the authentication server never observes biometric information in the clear, and cannot accidentally leak this information in the event of a breach. We show how our blinding technique can be combined with the privacy-preserving GSHADE protocol to privately compare biometric feature vectors using a variety of distance metrics with negligible overhead in computation time. We then construct a complete privacy-preserving remote fingerprint authentication system based on the Euclidean Distance metric, and show that a user can authenticate using privacy-preserving techniques in as little as 1.5 seconds. This work provides a template for designing low-cost blinding techniques for biometric authentication systems, and shows a practical use-case for secure multiparty computation protocols in remote authentication systems.**

I. Introduction

Fingerprint biometrics have been used for identification for centuries. More recently, fingerprints are becoming an increasingly popular means for electronic authentication for both device access and remote service authentication. A variety of smartphones from Apple [1], Samsung [42], and Google's Nexus line [22] of devices include fingerprint scanners for identifying the user, which can then be used to authenticate to cloud-based applications [3]. In addition, projects like the FIDO Alliance [18] are attempting to develop standard protocols for authentication using fingerprints and other biometric authenticators. As computing power increases and allows adversaries to crack conventional passwords more easily [14], these alternative authenticators will be critical for authenticating smartphone or netbook users to the growing number of cloud-based services.

To improve the security of biometric authentication and add privacy-preserving guarantees, secure multi-party computation (SMC) protocols offer the ability to compare and verify biometric credentials in a way that maintains the privacy of each party's input. Several such protocols have been developed to allow for a wider range of biometric applications, such as anonymous login and searching private biometric databases [27], [7], [6]. However, many of these protocols assume that the authentication server possesses the user's biometric template in plaintext. Should an attacker breach the security of the authentication server, this lack of template protection would allow the attacker to collect biometric data for all the system users. Such a breach of privacy occurred in 2015 in the United States Office of Personnel Management (OPM), where hackers stole 5.6 million fingerprint records belonging to federal employees [23]. As another example, an NSA document leaked by Edward Snowden indicated that India's Aadhaar database, a state-run database of biometric data for Indian citizens [44], may have been breached by national intelligence agencies [46]. These types of privacy breaches are especially dangerous for biometric data, which cannot be revoked or renewed. To allow for widespread use of biometrics as a remote authenticator, template security must be strengthened to resist accidental leakage.

In this work, we develop a novel technique for blinding biometric templates stored on an authentication server. Building on the recent GSHADE [6] protocol for privately computing distance metrics, we modify the protocol to allow the user to input an additive blind at registration that hides the template stored on the authentication server. Then, each authentication request requires the user to input their biometric information as well as the blind. Without possessing both of these components, an imposter cannot successfully authenticate to the server. Furthermore, if the blinded templates are leaked through a data breach, we provide a simple reblinding technique that allows a user to rotate the blind on their biometric template and continue authenticating through the system. We implement our protocol and demonstrate that blinding incurs minimal computational overhead when compared to the standard GSHADE protocol, requiring instead an expansion in the template size as small as 0.6%.

Our work presents the following major contributions:

- **A novel template blinding technique:** We develop a novel technique for blinding the authentication template in the GSHADE protocol. Our blinding technique works with any distance metric that is computable using the GSHADE protocol, incurring an expansion in the size of the stored template rather than an increase in computation time. Unlike existing template encryption schemes that rely on costly homomorphic encryption [36], our scheme maintains a minimal overhead in execution time by using additive blinds that do not require cryptographic primitives to add or remove.
- **A privacy-preserving biometric authentication system:** We demonstrate the practicality of our blinded GSHADE protocol by constructing a complete remote authentication system using garbled circuits to verify the measurements computed in our protocol. Our execution results show that the required storage expansion in this application is only 0.6%, and a user can authenticate in a private manner in only 1.5 seconds.

The remainder of this work is organized as follows: Section II describes related research, Section III outlines underlying primitives and definitions of security, Section IV defines our blinded version of the GSHADE protocol, Section V provides sketches of the security proofs of our protocol, Section VI describes a remote authentication scheme based on our blinded protocol, Section VII presents our implementation and evaluation results, and Section VIII provides concluding remarks.

II. Related Work

A vast number of fingerprint matching algorithms exist in the literature. A comprehensive survey [41] of fingerprint matching techniques compared the accuracy of techniques that use basic minutiae information [45], texture information [17], and 3-D data structures [8]. While highly accurate, techniques measuring local minutiae do not perform as efficiently as those that examine global structures [30]. In particular, the FingerCode encoding technique [28] allows for fingerprints to be represented with a fixed-length integer vector. Comparison between fingerprints is achieved by calculating the Euclidean Distance between two vectors. The fixed length representation and simplified comparison between vectors makes FingerCode particularly attractive to privacy-preserving computation techniques. We discuss the details of FingerCode further in Section III.

Secure multiparty computation was first developed by Yao with the garbled circuit [47], allowing mutually distrustful parties to jointly compute a result without revealing their private inputs. Since then, a variety of SMC protocols have been developed [21], [19] and expanded to provide security against a variety of adversaries [43], [2], [26]. Recent work has focused on developing optimized implementations by streamlining circuit compilation [39], [35], re-using previously encrypted values [37], and combining multiple SMC techniques [24], [34], [16]. Specially designed protocols for resource-constrained devices such as smartphones [25], [9] use cloud infrastructure [31], [11], [12], [29], [10], [32], [13], and secure hardware tokens [15] to minimize the computational load of the cryptographic operations. While these techniques produce more efficient implementations, their general-purpose nature and worst-case adversary modeling produce protocols that could be streamlined depending on the application.

Using techniques from general-purpose SMC, a number of privacy-preserving authentication protocols have been developed for various biometrics. In SciFi [40], Osadchy implemented facial recognition as a secure comparison of hamming distance between two strings. Their protocol is based on homomorphic encryption and oblivious transfers (OTs). Iris matching protocols based on a weighted hamming distance comparison have been developed using homomorphic encryption [4], oblivious transfer [7], and probabilistic estimation of the Jaccard index using MinHashes [5]. Finally, and most importantly to this work, protocols for secure fingerprint matching have been developed using both FingerCode [27], [6] as well as minutiae-based matching techniques [33], [36]. These protocols all allow for comparison between a query feature vector and one or more templates stored in an authentication database. Using this model, the authors construct a variety of applications, including searching a private identity database, anonymous login, and remote user authentication. Unfortunately, the vast majority of these protocols assume that the authentication database possesses the biometric templates in an unencrypted form. Because of this, if the authentication server is ever breached, all the biometric templates stored there would be leaked to the adversary, rendering them useless as authenticators. The only protocol that provides template protection is the minutiae-based fingerprint matching protocol by Li et al. [36], which uses costly homomorphic encryption operations to execute inefficient minutiae-based comparisons, and is not optimized for practical use. In this work, we seek to improve the security of fast global structure biometric comparison by adding template blinding to the values stored on the authentication server. This will allow for efficient biometric comparison and authentication while preserving the security of the user's biometric template in the event of an authentication server compromise.

III. Background

We outline the underlying protocols and assumptions in our privacy-preserving biometric authentication system in the following section.

A. FingerCode

To encode fingerprints in our scheme, we use the FingerCode [28] mechanism, as in several existing protocols [27], [6]. FingerCode uses Gabor edge detection filters to detect fingerprint features at different rotations, then combines these measurements into a fixed-length vector representation. Comparison using this representation is achieved efficiently using Euclidean distance. While minutiae-based comparison techniques are more common in practice [36], they require each minutiae in a query fingerprint be compared to *all* minutiae in the template, incurring quadratic complexity in execution time. Furthermore, the minutiae vectors are not required to be the same length, which makes developing SMC circuits to process these vectors inefficient. FingerCode lends itself better to automated analysis than minutiae-based schemes and is particularly suited to SMC applications due to its fixed-length representation and linear comparison algorithm with respect to the template length.

B. GSHADE

Our protocol for comparing two FingerCode representations in a privacy-preserving manner is a modified version of the GSHADE, or Generalized SHADE protocol [6]. This protocol allows for privacy-preserving computation of the family of functions $\mathcal{F}^{GSHADE}$ that can be represented as a sum of the form:

$$f(X,Y) = f_X(X) + \sum_{i=1}^{n} f_i(x_i, Y) + f_Y(Y)$$

Where X is the private input of the client, Y is the private input of the server, n is the bit length of X, and x_i is the ith bit of X. The key insight to their protocol is that the two functions $f_X(X)$ and $f_Y(Y)$ can be computed locally by the owners of those input values. The middle component can then be evaluated with OTs that return the bit-wise multiplication of Y by each bit in X. As long as this function can be represented with linear operations in both X and Y, the function can be evaluated in a privacy-preserving manner that hides each party's input from the other party. The authors of GSHADE show how this comparison technique can be used to evaluated Hamming distance, scalar product, squared Euclidean distance, and squared Mahalanobis distance. They then observe that this comparison technique can be combined with SMC protocols to produce a complete biometric comparison system. Unfortunately, their protocol relies on the server possessing the biometric templates Y in plaintext, which does not provide the users of the system any privacy guarantees in the event that the authentication server is breached and the templates are leaked.

1) Scalar Product: As a simple example, the GSHADE authors show how their protocol can evaluate the scalar product of two integer vectors defined as:

$$X = (X_1, \ldots, X_K) \text{ and}$$
$$Y = (Y_1, \ldots, Y_K)$$

The bits of each element in X and Y are defined as:

$$X_i = (x_{K(i-1)+1}, \ldots, x_{K(i-1)+\ell}) \text{ and}$$
$$Y_i = (y_{K(i-1)+1}, \ldots, y_{K(i-1)+\ell})$$

For each vector, K is the number of integers in the vector and ℓ is the bit length of each integer. Given these inputs, the component functions are defined as:

$$f_X(X) = f_Y(Y) = 0 \text{ and}$$
$$f_{K\cdot(i-1)+j}(x_{K(i-1)+j}, Y) = 2^{j-1} \cdot x_{K(i-1)+j} \cdot Y_i$$

Where $i = 1, \ldots, K$ and $j = 1, \ldots, \ell$.

2) Squared Euclidean Distance: For our FingerCode based authentication system, we must measure the Euclidean distance between two FingerCode vectors to authenticate the user. The original GSHADE work shows that the squared Euclidean distance can be computed between two K-integer vectors as defined in the previous example. The component functions are defined as:

$$f_X(X) = \sum_{i=1}^{K} (X_i)^2,$$

$$f_Y(Y) = \sum_{i=1}^{K} (Y_i)^2 \text{ and}$$

$$f_{K\cdot(i-1)+j}(x_{K(i-1)+j}, Y) = -2^j \cdot x_{K(i-1)+j} \cdot Y_i$$

Where $i = 1, \dots, K$ and $j = 1, \dots, \ell$.

C. Adversary Model and Security Definitions

As with the original GSHADE protocol, we assume that both parties are semi-honest during protocol execution. While our template hiding technique provides confidentiality for users if the templates are leaked to an adversary, it does not maintain privacy against a compromised and arbitrarily malicious authentication server. In a real-world deployment, other intrusion detection mechanisms would be necessary to ensure that a breach is detected and patched before users attempt to authenticate again. To demonstrate the security of our scheme, we use the real-ideal security definition that is standard for SMC protocols [20]. We show that a simulator given the output of the computation by a trusted arbitrator in the ideal world can simulate the opposing party's view of a real-world protocol execution. We refer the reader to the canonical literature for details [20].

D. Notation

Throughout the remainder of the work, we refer to the following variables. Let n be the bit-length of any input value X and m be the bit-length of the function output $f(X, Y)$. We denote the set of integers of bit length n as $\mathbb{Z}_n$, and a uniformly random integer r from that set as $r \in_\$ \mathbb{Z}_n$. Finally, When referring to integer vectors of length K we denote them as $\mathbb{Z}_n^K$.

IV. PROTOCOL

In this section, we define the challenges with blinding stored templates and our solution to preserving template privacy in the GSHADE protocol.

Shared inputs: The distance function $f(X, Y) \in \mathcal{F}^{GSHADE}$ represented by three component functions $f_X(X), f_Y(Y), f_i(x_i, Y)$. The length of the output $|f(X, Y)| = m$ bits.

Private inputs: The client C inputs an n-bit string $X = (x_1, \dots, x_n)$ and two blinding values $B_1 \in \{0,1\}^m$ and $B_2 \in \{0,1\}^n$. The server S inputs the blinded template bit string $Y + B_2$ and the precomputed, blinded value $f_Y(Y) + B_1$.

Outputs: The server S obtains the blind R, while the client obtains the blinded result $f(X, Y) + R$.

1) S generates $r_1, \dots, r_n \in_\$ \mathbb{Z}_m$ and computes $R = \sum_{i=1}^{n} r_i$.
2) For $i = 1, \dots, n$, S and C run a 1-out-of-2 OT where:
    - S is the sender and C is the receiver
    - C's input is the bit x_i
    - S's inputs are $(r_i + f_i(0, Y + B_2), r_i + f_i(1, Y + B_2))$
    - C obtains $t_i = f_i(x_i, Y + B_2) + r_i$
3) For $i = 1, \dots, n$, C locally computes $k_i = f_i(x_i, B_2)$.
4) C computes the result $T = \sum_{i=1}^{n}(t_i - k_i) + f_X(X) - B_1$
5) S computes the result $Z = R - (f_Y(Y) + B_1)$
6) To recover the function output, the parties can subtract their results $T - Z = f(X, Y)$

Fig. 1. The GSHADE protocol modified with template blinding.

A. Challenges in GSHADE

The core idea of our template hiding scheme is to use additional additive blinds similar to the blinds used to hide the result of the GSHADE distance function from the client after the first phase of their protocol. Rather than store the actual template Y, the server will store a blinded template $Y + B$, while the client will provide the blind B to allow for authentication against the original template Y. As we show in Section V, the client must provide both the blind B and a matching input X to properly authenticate. In addition, as long as the blinds are chosen from a large bit length, this additive blind will be statistically hiding of the real template. Furthermore, we show that unlike other template hiding protocols using homomorphic encryption [36], our protocol adds a negligible computational overhead and allows for simple key rotation in the event of a compromise.

As stated in the previous section, GSHADE allows for the privacy-preserving computation of any distance metric that can be represented in the form $f(X, Y) = f_X(X) + \sum_{i=1}^{n} f_i(x_i, Y) + f_Y(Y)$. Given that the client possesses input X and the server possesses input Y, the task of performing the same computation using a blinded Y value presents two challenges. The first is that the computation of $f_Y(Y + B)$ can no longer be computed independently by either party since the server does not possess B and the client does not possess the template Y. The second challenge is that the interactive computation of the $\sum_{i=1}^{n} f_i(x_i, Y)$ component now requires additional offset based on the blinding value B.

B. Our Solution

We solve both challenges with our approach to template blinding. To solve the first challenge, we observed that the function based on the template $f_Y(Y)$ can be computed once and reused each time the client authenticates. Thus, rather than develop an additional, potentially costly protocol to compute $f_Y(Y + B)$, we allow the client to provide two blinds, B_1 and B_2, then blind the precomputed $f_Y(Y)$ at registration. This means that the server now stores the value $f_Y(Y) + B_1$ and $Y + B_2$. While this solution slightly increased the storage required, it allows the value $f_Y(Y) + B_1$ to be un-blinded using a trivial additive operation.

To solve the second challenge, we observed that as long as the computation of the function $f_i(x_i, Y + B)$ is linear with respect to x_i, any multiplicative operation based on the value x_i will distributed between the server-provided values as $x_iY + x_iB$. For functions that meet this property, we have that $f_i(x_i, Y + B) = f_i(x_i, Y) + f_i(x_i, B)$ since the client knows both x_i and B, it can correctly compute the offset value containing the x_iB term and subtract it from the result, eliminating the offset caused by the additive blind B. We formally define the modified GSHADE protocol in Figure 1.

V. Security

The security of our modified scheme follows closely from the security of the underlying GSHADE protocol. We discuss the security provided by blinding here, and briefly sketch the security of the protocol when each party is corrupt. For a more complete proof of security, see the original GSHADE [6] and SHADE [7] protocols.

A. Blinding

Our protocol calls for the template Y to be stored as the sum $Y + B_2$. Additionally, we store $f_Y(Y)$ added to the blind B_1. With the value Y represented as n bits and the value $f_Y(Y)$ as m bits, we sample uniformly random values $B_1 \in \{0,1\}^m$ and $B_2 \in \{0,1\}^n$. Given these uniformly random strings, it is clear that the sum $f_Y(Y) + B_1$ is uniformly random in $\{0,1\}^m$ and $Y + B_2$ is uniformly random in $\{0,1\}^n$, revealing no information about Y to the authentication server.

B. Simulator Proof of Security

We can prove the security of the protocol by examining two cases, where the authentication server S is corrupt, and where the authenticating client C is corrupt:
Case 1: S is corrupt. Simulating the view of S follows directly from the security of the underlying OT protocol, as S never receives any messages outside of the OTs executed. Thus, assuming the OT is secure (i.e., is simulatable), then the view of S is also simulatable.
Case 2: C is corrupt. Because the output of the OT for the client C is blinded by the random values r_i for $i \in 1, .., n$, the view of C can be simulated by providing random strings as the server input to the OTs. The distribution of the output values, in both the simulated protocol and the real protocol, will be uniformly random. Note also that, after the simulator receives the output of the computation $f(X, Y)$ from the trusted third party in the ideal world, the correct result can be revealed by returning the un-blinding value $R - f(X, Y)$ to C, who can recover the result as $R - (R - f(X, Y))$.

VI. Application

Our blinded GSHADE protocol offers a privacy preserving method for computing several distance functions between a querying client and a server holding blinded templates. However, this protocol alone does not make a complete remote authentication service. We describe the architecture and requisite functionality to build a remote authentication service using fingerprint biometrics.

A. Server Setup

Our application consists of two participants. The Server is a remote service that authenticates users by their fingerprints. The Client is a user with a pre-established account on the Server that authenticates by providing a fingerprint on her local device, which is then used to access the remote application. We assume the Client's local device possesses a fingerprint scanner and that it is capable of securely storing a blinding value that is only accessed locally (e.g., a smartphone or laptop computer). We assume the Server knows the Client who is attempting to authenticate, so no anonymity is

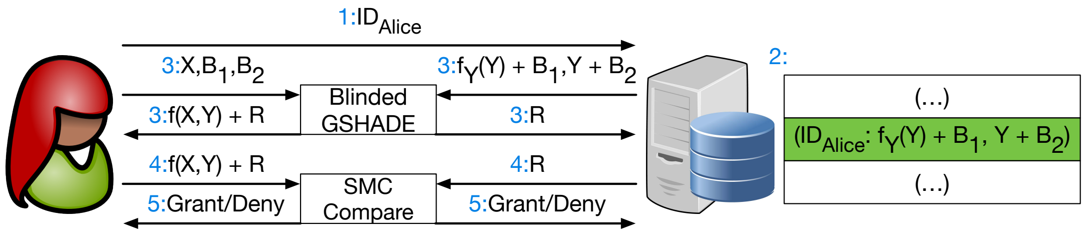

Fig. 2. The privacy-preserving login procedure. Recall that $f(X, Y)$ is the squared Euclidean distance between two FingerCode vectors.

provided. Our primary security goal is to authenticate the CLIENT based on a biometric measurement while keeping the biometric template stored on the SERVER blinded throughout the authentication.

B. Registration

Before a CLIENT can authenticate to the SERVER, she must register her identity along with a template measurement of her fingerprint. We note that registration must occur using an out-of-band communication channel with a separate authentication method (e.g., in person registration with a new company). Throughout the registration, we let K be the number of features in the fingerprint template, and we let ℓ be the number of bits representing a single feature:

1) The CLIENT measures her fingerprint and produces a FingerCode representation $Y \in \mathbb{Z}_\ell^K$.
2) She then evaluates her portion of the squared Euclidean distance based on this value, $f_Y(Y)$. Given input vectors of bit length $n = K \times \ell$, we note that the maximum number of bits needed to represent the squared Euclidean distance is $m = 2\ell + \lceil \log_2 K \rceil$.
3) The CLIENT concludes by generating two blinds, $B_1 \in_\$ \mathbb{Z}_m, B_2 \in \mathbb{Z}_\ell^K$ and transmits $B_1 + f_Y(Y)$ and $B_2 + Y$ to be stored on the SERVER. The SERVER stores the two blinded values, while the CLIENT stores the blinds B_1 and B_2.

C. Login

The login procedure, shown in Figure 2, consists of two phases: computing the distance between the queried fingerprint and the stored template, and comparing that distance to a threshold using a privacy-preserving SMC protocol. We use our blinded GSHADE to accomplish the first step. While several techniques using a variety of SMC primitives have been proposed for the second step, our application uses a simple garbled circuit-based comparison implemented with the compiler and execution environment developed by Mood et al. [38]. Because this application does not require comparison with an entire database of templates, we do not need the sophisticated techniques applied in previous work [27], [6]. However, we note that the SMC comparison in our application can be implemented using *any* SMC technique.

Login proceeds according to the following steps:

1) The CLIENT contacts the SERVER and presents her identity ID_{Alice} to the SERVER. At this point, she may use the fingerprint scanner in her local device to produce a FingerCode representation of her fingerprint X.
2) Using ID_{Alice}, the SERVER retrieves the template stored for that identity, $B_1 + f_Y(Y)$, $B_2 + Y$.
3) The CLIENT and SERVER run the blinded GSHADE protocol in Section IV to compute the squared Euclidean distance between the query fingerprint X and the template Y. The CLIENT inputs B_1, B_2, X and receives the blinded squared Euclidean distance $f(X, Y) + R$ as output. The SERVER inputs $B_1 + F_Y(Y)$, $B_2 + Y$ and receives a new random blind R as output.
4) The CLIENT and SERVER run a garbled circuit computation that takes $f(X, Y) + R$ and R as their respective inputs and performs the following steps:
   a) Compute the unblinded squared Euclidean Distance as $D = (f(X, Y) + R) - R$
   b) Compare the result as the boolean value $D \leq \delta$ for some fixed threshold δ
   c) If $D \leq \delta$ is true, output `Grant` to both parties. Otherwise, output `deny`.
5) If the previous phase of the login outputs `Grant`, the SERVER grants the CLIENT access to the remote service. Otherwise, access is denied.

D. Key Refresh

In the event of a compromise of the templates or blinding values, our protocol allows for simple blind rotation. As in the registration step, a CLIENT can contact the

SERVER through an authenticated, out-of-band channel to update the blinds on the stored templates.

1) The CLIENT first chooses new blinding values $B_1' \in_\$ \mathbb{Z}_m, B_2' \in \mathbb{Z}_\ell^K$, and calculates the difference between these blinds and the previous blinds as $\Delta_1 = B_1' - B_1, \Delta_2 = B_2' - B_2$.
2) The CLIENT sends these rotation values to the SERVER, who can add them to the stored values as $B_1 + f_Y(Y) + \Delta_1 = B_1' + f_Y(Y)$ and $B_2 + Y + \Delta_2 = B_2' + Y$.
3) For future login attempts, the CLIENT authenticates with the new blinds B_1', B_2' as inputs.

E. Limitations

We note two limitations that restrict the applications of our blinding technique. The first is that it can only evaluate functions in the family $\mathcal{F}^{GSHADE}$. While this covers many distance metrics, it is not universally applicable. Second, because our protocol requires each entry to be blinded by unique blinds, the scheme cannot be used for identifying a fingerprint in a database. This operation would require comparison of the query fingerprint to all templates, which would either require the client to possess *all* blinding values or to have a single blinding value used across the database, which would break the one-time security of the blind.

VII. EVALUATION

In this section, we evaluate the blinded GSHADE protocol by measuring the overhead in execution time and storage required when compared to the original GSHADE protocol. We then measure the total time for a user to authenticate in our authentication application.

A. Experimental Setup

We implemented our blinded GSHADE in C on the original GSHADE code by Bringer et al. We ran both participants on an Ubuntu 64-bit server with 2 6-core 2.3 GHz Intel Xeon processors and 32 GB memory, with parties communicating over localhost. We measured the execution time for the squared Euclidean distance between integer vectors from 100-640 entries, with each entry represented as an 8-bit integer (matching the input sizes evaluated in [27], [6]). Each result is the average execution time over 10 trials with 95% confidence.

B. Execution time

The execution times of both the blinded and unblinded GSHADE protocols are shown in Figure 3. For each input length tested, the average execution time for both the blinded and un-blinded versions of the protocol had overlapping confidence intervals, indicating that the additional arithmetic operations for blinding and unblinding added a negligible amount of overhead. At the largest input size (640 integer vectors), the overhead cost of blinding increased the execution time from 990 ms to 1022 ms, an increase of 3%. Our protocol achieves this minimal increase in execution time because the only additional operations over the original GSHADE protocol are a constant number of addition operations and the evaluation of the function $f(X, B_2)$, which can be evaluated by the client offline before the authentication begins. Furthermore, because the value $f_Y(Y)$ is stored at the authentication server in a blinded manner rather than evaluated during authentication, we eliminate the computation of $f_Y(Y)$ that is required by the original GSHADE protocol. These results clearly show that template security can be achieved with virtually no noticeable cost in execution time for the client.

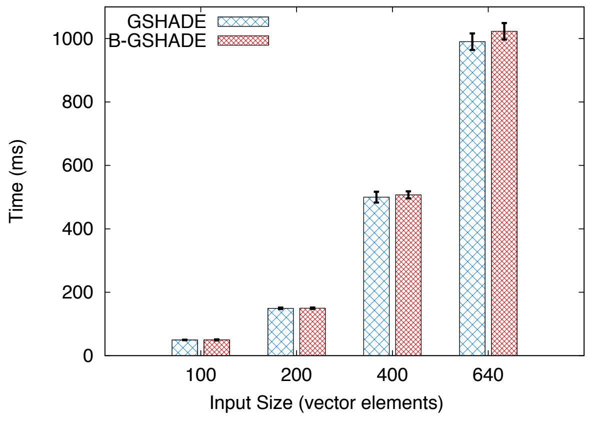

Fig. 3. Execution times comparing standard GSHADE to our blinded version. Note that the confidence intervals for each input length overlap, indicating a negligible execution overhead.

Vector Length	GSHADE	B-GSHADE	% incr.
100	100	103	3.0
200	200	203	1.5
400	400	404	1.0
640	640	644	0.6

TABLE I
STORAGE REQUIRED FOR A SINGLE TEMPLATE. VECTOR LENGTH IS THE NUMBER OF INTEGERS STORED AND STORAGE IS IN BYTES. NOTE THAT THE GSHADE STORAGE GROWS LINEARLY WITH THE VECTOR WHILE THE OVERHEAD INCURRED BY BLINDING GROWS LOGARITHMICALLY, CAUSING THE OVERHEAD PERCENTAGE TO DECREASE.

C. Storage Overhead

In Table I, we list the total amount of storage required for the authentication database using the blinded and un-blinded versions of GSHADE. As described in Section VI, for each entry in the authentication database, our scheme requires $m = 2\ell + \lceil \log_2 K \rceil$ bits to store the blinded value $f_Y(Y)$ in addition to the $n = K \times \ell$ bits needed to store the actual fingerprint template. Thus, for a given input size, the original GSHADE requires n bits for each database entry, while our scheme requires $n+m$ bits. However, this overhead diminishes as the length of the template vector grows, since the linear growth of n dominates the logarithmic growth of m. With 0.6% increase in storage for a template length of 640 elements, in a database of $100,000$ users, our blinding scheme would need 64.4 MB of storage compared to the 64 MB required by the original GSHADE protocol. With almost no overhead cost in execution time, this small storage expansion represents the only appreciable tradeoff for the increased security and resilience against data breaches afforded by our template blinding technique. While these results represent the overhead cost for the squared Euclidean distance metric in particular, we note that for distance metrics where $f_Y(Y) = 0$ (e.g., hamming distance and scalar product), *no additional storage overhead is incurred beyond the original GSHADE protocol.*

D. User Authentication

To demonstrate the practicality of our template blinding scheme, we implemented the login procedure outlined in Section VI. We implemented the SMC comparison portion of the computation in the semi-honest secure garbled circuit system developed by Mood et al. [38]. Previous SMC protocols for biometric matching [27], [6] were designed for general biometric search, and are built to compare the query fingerprint to *every* template in a fingerprint database. Because our application only compares the query element to *one* entry in the authentication database, we do not need sophisticated implementation optimizations such as backtracking [27].

We computed the total authentication time by adding the average execution time of the Euclidean distance between 640-entry feature vectors with the average execution time of the SMC blind removal and comparison. With a comparison circuit of 265 gates, the garbled circuit computation only requires $0.512(\pm 0.01)$ seconds to compute. Adding this to the blinded GSHADE execution results shows that a user can authenticate in only 1.5 seconds. This execution time demonstrates the computational feasibility of our authentication system, and represents a practical application of SMC that is achievable with current techniques.

VIII. Conclusion

As fingerprint scanners become more common on consumer devices, biometric authentication will continue to develop as a popular technique for identifying users to cloud-based applications. As biometric authentication becomes more common, maintaining the privacy of a user's authenticating template will be critical since revocation is essentially impossible. In this work, we demonstrated a technique for adding rotatable blinds to the user's biometric template in the GSHADE protocol. Unlike current privacy-preserving biometric protocols, in the event of a breach, our scheme hides the user's private biometric information, and allows for the rotation and continued use of the same biometric template. After implementing and benchmarking our protocol, we show that it can be implemented with essentially no overhead in execution time and a 0.6% expansion in the storage required at the authentication server. Ultimately, this work demonstrates a practical solution to template security, which will allow biometric authentication to be more secure in wide-scale implementations.

Acknowledgments

This work was supported in part by the US National Science Foundation under grant numbers CNS-1464088 and CNS-1540217. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References

[1] Apple Inc. iPhone. http://www.apple.com/iphone/, 2016.

[2] Y. Aumann and Y. Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. *Journal of Cryptology*, 18(3):554–343, 2010.

[3] Bank of America. Online Banking and Mobile Banking Features. https://www.bankofamerica.com/online-banking/mobile-and-online-banking-features/, 2016.

[4] M. Blanton and P. Gasti. Secure and efficient protocols for iris and fingerprint identification. In *Proceedings of the European Conference on Research in Computer Security (ESORICS)*, 2011.

[5] C. Blundo, E. De Cristofaro, and P. Gasti. EsPRESSO: Efficient privacy-preserving evaluation of sample set similarity. *Journal of Computer Security*, 22(3):355–381, 2014.

[6] J. Bringer, H. Chabanne, M. Favre, A. Patey, T. Schneider, and M. Zohner. GSHADE: Faster privacy-preserving distance computation and biometric identification. In *Proceedings of the ACM workshop on Information hiding and multimedia security*, 2014.

[7] J. Bringer, H. Chabanne, and A. Patey. SHADE: Secure hamming distance computation from oblivious transfer. In *Financial Cryptography and Data Security (FC)*, 2013.

[8] R. Cappelli, M. Ferrara, and D. Maltoni. Minutia cylinder-code: a new representation and matching technique for fingerprint recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, (32), 2010.

[9] H. Carter, C. Amrutkar, I. Dacosta, and P. Traynor. For your phone only: custom protocols for efficient secure function evaluation on mobile devices. *Journal of Security and Communication Networks (SCN)*, 7(7):1165–1176, 2014.

[10] H. Carter, C. Lever, and P. Traynor. Whitewash: Outsourcing garbled circuit generation for mobile devices. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, 2014.

[11] H. Carter, B. Mood, P. Traynor, and K. Butler. Secure outsourced garbled circuit evaluation for mobile devices. In *Proceedings of the USENIX Security Symposium*, 2013.

[12] H. Carter, B. Mood, P. Traynor, and K. Butler. Outsourcing secure two-party computation as a black box. In *Proceedings of the International Conference on Cryptology and Network Security (CANS)*, 2015.

[13] H. Carter and P. Traynor. OPFE: Outsourcing computation for private function evaluation. Cryptology ePrint Archive, Report 2016/067, 2016. http://eprint.iacr.org/2016/067.

[14] L. S. Clair, L. Johansen, W. Enck, M. Pirretti, P. Traynor, P. McDaniel, and T. Jaeger. Password exhaustion: Predicting the end of password usefulness. In *Proceedings of the International Conference on Information Systems Security (ICISS)*, 2006.

[15] D. Demmler, T. Schneider, and M. Zohner. Ad-hoc secure two-party computation on mobile devices using hardware tokens. In *Proceedings of the USENIX Security Symposium*, 2014.

[16] D. Demmler, T. Schneider, and M. Zohner. ABY – a framework for efficient mixed-protocol secure two-party computation. In *Proceedings of the ISOC Symposium on Network and Distributed Systems Security*, 2015.

[17] H. Deng and Q. Huo. Minutiae matching based fingerprint verification using delaunay triangulation and aligned-edge-guided triangle matching. In *Proceedings of the International Conference on Audio- and Video-Based Biometric Person Authentication (AVBPA)*, 2005.

[18] FIDO Alliance. Specification Overview. https://fidoalliance.org/specifications/overview/, 2016.

[19] C. Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009.

[20] O. Goldreich. *Foundations of Cryptography*, volume 2. Cambridge University Press, 2001.

[21] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the Annual ACM Symposium on Theory of Computing*, 1987.

[22] Google. Nexus 5X. http://www.google.com/nexus/5x/, 2016.

[23] A. Greenberg. OPM now admits 5.6m feds' fingerprints were stolen by hackers. https://www.wired.com/2015/09/opm-now-admits-5-6m-feds-fingerprints-stolen-hackers/, 2015.

[24] W. Henecka, S. Kögl, A.-R. Sadeghi, T. Schneider, and I. Wehrenberg. TASTY: tool for automating secure two-party computations. In *Proceedings of the ACM conference on Computer and Communications Security*, 2010.

[25] Y. Huang, P. Chapman, and D. Evans. Privacy-preserving applications on smartphones. In *Proceedings of the USENIX Workshop on Hot Topics in Security*, 2011.

[26] Y. Huang, J. Katz, and D. Evans. Quid-pro-quo-tocols: Strengthening semi-honest protocols with dual execution. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2012.

[27] Y. Huang, L. Malka, D. Evans, and J. Katz. Efficient privacy-preserving biometric identification. In *Proceedings of the ISOC Network and Distributed System Security Symposium (NDSS)*, 2011.

[28] A. K. Jain, S. Prabhakar, L. Hong, and S. Pankanti. FingerCode: a filterbank for fingerprint representation and matching. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1999.

[29] T. P. Jakobsen, J. B. Nielsen, and C. Orlandi. A framework for outsourcing of secure computation. In *Proceedings of the ACM Workshop on Cloud Computing Security (CCSW)*, 2014.

[30] X. Jiang and W. Yau. Fingerprint minutiae matching based on the local and global structures. In *International Conference on Pattern Recognition (ICPR)*, 2000.

[31] S. Kamara, P. Mohassel, and B. Riva. Salus: A system for server-aided secure function evaluation. In *Proceedings of the ACM conference on Computer and communications security (CCS)*, 2012.

[32] F. Kerschbaum. Oblivious outsourcing of garbled circuit generation. In *Proceedings of the Annual ACM Symposium on Applied Computing*, 2015.

[33] F. Kerschbaum, M. J. Atallah, D. M'Raïhi, and J. R. Rice. Private fingerprint verification without local storage. In *Proceedings of the First International Conference on Biometric Authentication*, 2004.

[34] F. Kerschbaum, T. Schneider, and A. Schröpfer. Automatic protocol selection in secure two-party computations. In *Proceedings of the International Conference on Applied Cryptography and Network Security*, 2014.

[35] B. Kreuter, B. Mood, a. shelat, and K. Butler. PCF: A portable circuit format for scalable two-party secure computation. In *Proceedings of the USENIX Security Symposium*, 2013.

[36] M. Li, Q. Feng, J. Zhao, M. Yang, L. Kang, and L. Wu. Minutiae matching with privacy protection based on the combination of garbled circuit and homomorphic encryption. *The Scientific World Journal*, 2014(525387), 2014.

[37] B. Mood, D. Gupta, K. Butler, and J. Feigenbaum. Reuse it or lose it: More efficient secure computation through reuse of encrypted values. In *Proceedings of the ACM conference on Computer and communications security (CCS)*, 2014.

[38] B. Mood, D. Gupta, H. Carter, K. Butler, and P. Traynor. Frigate: A validated, extensible, and efficient compiler and interpreter for secure computation. In *Proceedings of the IEEE European Symposium on Security and Privacy*, 2016.

[39] B. Mood, L. Letaw, and K. Butler. Memory-efficient garbled circuit generation for mobile devices. In *Proceedings of the IFCA International Conference on Financial Cryptography and Data Security (FC)*, 2012.

[40] M. Osadchy, B. Pinkas, A. Jarrous, and B. Moskovich. SCiFI-a system for secure face identification. In *Proceedings of the IEEE Symposium on Security & Privacy*, 2010.

[41] D. Peralta, M. Galar, I. Triguero, D. Paternain, S. Garca, E. Barrenechea, J. M. Bentez, H. Bustince, and F. Herrera. A survey on fingerprint minutiae-based local matching for verification and identification: Taxonomy and experimental evaluation. *Information Sciences*, 2015.

[42] Samsung. Galaxy S7 Edge. http://www.samsung.com/us/mobile/cell-phones/SM-G935AZDAATT, 2016.

[43] a. shelat and C.-H. Shen. Fast two-party secure computation with minimal assumptions. In *Proceedings of the ACM conference on Computer and communications security (CCS)*, 2013.

[44] Unique Identification Authority of India. Aadhaar Service. http://www.uidai.gov.in/what-is-aadhaar.html, 2016.

[45] C. Watson, M. Garris, E. Tabassi, C. Wilson, R. Mccabe, S. Janet, and K. Ko. User's guide to NIST biometric image software (NBIS). Technical report, NIST, 2010.

[46] Y. Yadav. Aadhaar Data Minefield Threatens to Blow Up in Government's Face. http://www.newindianexpress.com/thesundaystandard/Aadhaar-Data-Minefield-Threatens-to-Blow-Up-in-Government's-Face/2014/06/08/article2268540.ece, 2014.

[47] A. C. Yao. Protocols for secure computations. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, 1982.