Secure Outsourced Biometric Authentication with Performance Evaluation on Smartphones

Jaroslav Šeděnka†, Sathya Govindarajan, Paolo Gasti*, Kiran S. Balagani
New York Institute of Technology
{jsedenka,sgovin,pgasti,kbalagan}@nyit.edu

***Abstract*—We design privacy-preserving protocols for Scaled Manhattan and Scaled Euclidean verifiers, secure against malicious clients and honest-but-curious server. We then augment our protocols with principal component analysis (PCA), which can help improve authentication accuracy. We evaluate the performance of our protocols on an emerging application—namely, continuous authentication of smartphone users. We compare the performance of protocols secure under the malicious client model, with three protocols secure in the honest-but-curious model. We report tradeoffs between computation overhead, communication cost, and authentication accuracy. Our key observations are: 1) Scaled Manhattan without PCA gives the best tradeoff between security, accuracy, and overhead; and 2) with PCA, memory availability on current smartphones limits the number of features that can be used with Scaled Manhattan, and prevents the Scaled Euclidean protocol from running. Our extended evaluation on a laptop client shows that PCA with both Scaled Manhattan and Scaled Euclidean verifiers is feasible given sufficient memory.**

***Index Terms*—Privacy-preserving authentication, behavioral biometrics, cryptographic protocols, secure multiparty computation, garbled circuits, homomorphic encryption**

I. Introduction

Outsourcing biometric authentication involves delegating authentication to third party service providers (e.g., AdmitOne Security [1] and BehavioSec [4]), who specialize in deploying and maintaining biometric systems. Despite the advantages that outsourcing could offer in terms of convenience and cost savings, it raises privacy and security concerns. Biometric data is sensitive, and disclosing it to a third party is undesirable.

In this paper, we address the problem of securely outsourcing biometric authentication. We characterize outsourced authentication as a two-party problem which involves: (1) a *client* (a device in the hands of a *user*); and (2) an *authentication server*. Our goal is to design protocols that allow both parties to perform authentication without disclosing any information—besides the outcome of the authentication (to the client) and the authentication score (to the server). Specifically, our protocols conceal the authentication *sample* collected by the client from the server, and the user's *template* from both parties.

A. Our Contributions

Protocols Secure Against Malicious Clients. We design protocols for Scaled Manhattan and Scaled Euclidean verifiers, secure against *malicious clients*. Previous work in this area (e.g., [19], [8]) mostly focused on providing security against honest-but-curious (HBC) adversaries. However, security in the HBC model may be inadequate for authentication because it assumes that the client and the server faithfully follow all protocol specifications. For instance, the HBC model does not guarantee that a malicious client cannot get authenticated by "cheating" or deviating from the protocol.

On the other hand, there are generic ways to transform HBC to fully malicious protocols (e.g., cut-and-choose [20], [33]). However, these transformations increase the overall protocol cost by *at least* two orders of magnitude [23]. To achieve a balance between security and cost, we designed protocols that are secure against malicious clients and HBC server. We believe our *hybrid* model is more practical for authentication: unlike HBC, our model ensures that a malicious client does not benefit from deviating from the protocol. At the same time, it is reasonable to assume that the authentication server can be audited, to expose any misbehavior not captured by the HBC model.

Augmenting Authentication Protocols with PCA. We augment our authentication protocols with principal component analysis (PCA), mainly for the following two reasons: (1) the verifiers considered in this paper assume that features are independent. PCA is commonly used as a preprocessing step to suppress covariance among features, to bring the data closer to the model assumption and thus possibly improve performance; and (2) although PCA has been used many times in biometric authentication (e.g., [48], [28]), to our knowledge no study has introduced secure protocols for verification in PCA space.

Protocol Evaluation for Smartphone Authentication. We evaluate the performance of the new protocols for continuous authentication of smartphone users. We believe this is both *timely* and *important*. The former, because more and more companies have started to offer smartphone user authentication

†J. Šeděnka (sedenka@mail.muni.cz) is a student of Faculty of Science, Masaryk University, Czech Republic. This work was done while visiting the New York Institute of Technology.

*Corresponding author. Mailing address: Room 808, 1855 Broadway, NY 10023, USA. Phone: 212-261-1609. Fax: 516-686-7933.

This work was supported in part by DARPA Active Authentication grant FA8750-13-2-0266 and a 2013 NYIT ISRC grants. The views, findings, recommendations, and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the sponsoring agencies or the U.S. Government.

as a service (e.g., [4]). The latter—because in order to make biometric outsourcing viable in the long term, privacy issues must be fully addressed. Another motivation for choosing smartphones for evaluating our protocols is that even though they have become ubiquitous, limited number of studies evaluated secure protocols on this platform. In fact, to our knowledge, only three studies ([19], [43], [8]) have explicitly addressed privacy-preserving continuous authentication in the smartphone domain. Through our evaluation, we aim to mainly address three questions: (1) are our protocols, secure in the hybrid model, practical for continuous authentication on commodity smartphones? (2) How do these protocols compare in terms of computation and communication overhead with protocols secure in the HBC model? And (3) how does PCA impact the cost of the protocols?

Our findings can be summarized as follows: (1) Our Scaled Manhattan and Scaled Euclidean protocols in the hybrid model are feasible on current commodity smartphones. Moreover, Scaled Manhattan outperforms Scaled Euclidean in terms of both computation and communication costs. (2) Our protocols secure in the hybrid model have lower communication cost than their HBC counterparts; however, they are more computationally expensive. (3) Although PCA provides a notable improvement in verification accuracy, it also significantly increases the computation and communication overhead. Scaled Manhattan in PCA space *can* be run on commodity smartphone devices, albeit with limited number of features. However, Scaled Euclidean in PCA space requires more memory than what the OS provides to apps.

Comparison with a New Lightweight Protocol. As a further point of comparison between protocols secure in our hybrid model versus protocols in the HBC model, we design a simple protocol for privacy-preserving 1-Prob verifier, secure in HBC. To our knowledge, our 1-Prob verifier is the fastest protocol for continuous authentication on smartphone devices.

B. Organization

In Section II, we review related research. In Section III, we present the required biometric and cryptographic background for our protocols. In Section IV, we introduce privacy-preserving protocols for authentication, and present their PCA counterparts in Section V. We evaluate the security of our protocols in Section VI and present their performance in Section VII. We conclude in Section VIII.

II. Related Work

Since the seminal work on garbled circuit evaluation [51], [18], it has been shown that any function can be securely evaluated by representing it as a boolean circuit. Similar results exist for secure evaluation of any function using secret sharing techniques, e.g., [40], or homomorphic encryption, e.g., [9].

In recent years, a number of tools have been developed for automatically creating a secure protocol from its function description written in a high-level language. Examples include Fairplay [34], VIFF [12] and TASTY [21]. However, "custom" optimized protocols for specific applications are often more efficient than such general techniques.

Garbled circuits offer security in the honest-but-curious model. However, a technique called *cut-and-choose* can be used to make garbled circuits secure in the malicious model. With cut-and-choose, the server creates multiple garblings of a circuit. The client randomly selects a subset of these garblings, and asks the server to reveal these circuits' input keys. The client verifies that all circuits are constructed properly, and evaluates the remaining circuits to obtain the result of the computation. Then, both parties switch roles and repeat this process. There are several approaches to implement cut-and-choose, each requiring a different number of circuits to achieve a given level of security. (See, e.g., [20] or [33].) In particular, recent results by Huang [23] achieve κ-bit security with approximately κ circuits.

Another approach to privacy-preserving computation is fully homomorphic encryption (FHE), first constructed by Gentry in [15]. FHE allows computation of arbitrary ring operations in the encrypted domain, and as such can be used for outsourcing biometric authentication. Despite advancements in the area, current implementations are still too slow, and impractical for our setting. (See for example [16] and [39].)

A number of publications address the problem of privacy-preserving biometric authentication and identification. Bringer et al. [7] were the first to introduce a general security model for biometric user authentication. The model assumes low trust between the involved parties, and formalizes privacy for biometric authentication. Furthermore, the paper introduces a privacy-preserving protocol that computes the Hamming distance of two bit strings, representing a biometric sample and a template.

Barbosa et al. [2] extend the framework of Bringer et al. [7] with a classifier to improve authentication accuracy and propose an instantiation based on Support Vector Machine (SVM) using homomorphic encryption.

Schoenmakers et al. [45] introduced a protocol for secure privacy-preserving iris matching. The protocol is implemented using threshold ElGamal, and computes (encrypted) Hamming distance between two bit strings representing a template and a user sample, encoded using IrisCode. The result of the Hamming distance is then compared, in encrypted form, with a threshold.

Secure face recognition was first addressed by Erkin et al. [13]. In this paper, the authors designed a privacy-preserving face recognition protocol based on Eigenfaces. Sadeghi et al. [42] subsequently improved the performance of the protocol of Erkin et al. More recently, Osadchy et al. [36] designed a new face recognition algorithm together with its privacy-preserving realization called SCiFI. SCiFI simultaneously improves robustness and efficiency of [42].

Blanton et al. [5] focused on privacy-preserving iris and fingerprint matching. The authors rely on a hybrid approach based on garbled circuits and homomorphic encryption for optimal performance. Barni et al. [3] presented a privacy-preserving protocol for fingerprint identification using Finger-Codes [25], which is not as discriminative as techniques based on location of minutiae points, but is particularly suited for efficient privacy-preserving implementations.

Govindarajan et al. [19] present two protocols for smart-

phone user authentication. They implemented Scaled Manhattan and Scaled Euclidean verifiers secure in the HBC model. In Section VII, we compare the performance of our protocols secure against malicious clients with the protocols in [19].

Safa et al. [43] present a protocol for outsourcing continuous authentication with a Scaled Manhattan verifier on smartphones, and consider security against malicious clients. However, the security argument presented in their HBC protocol does not take into account information disclosed by order-preserving encryption, even in ciphertext-only scenarios. (See, e.g., the analysis of Boldyreva et al. [6].) As such, there is substantial amount of information leaked by the client during the protocol. In contrast, our scheme *provably* leaks no information.

Chun et al. [8] introduced a garbled-circuit based protocol for outsourcing biometric authentication. Their protocol is secure in the honest-but-curious model and the performance reported by the authors is highly impractical: the lowest reported protocol execution time is approx. 39 minutes.

Techniques based on fuzzy commitments (e.g., [26], [49], [27] and [50]) are commonly used to provide template protection and to implement access control on encrypted documents. However, such techniques require biometric comparisons to be performed in a feature space different from that of the original biometrics, possibly increasing equal error rate (EER) [31]. In contrast, our protocols do not affect the EER of the underlying biometric modality, since the comparison between the user sample and the template is functionally the same as the comparison in the unencrypted domain.

III. Background

A. Biometric Verification

We implemented privacy-preserving protocols for three verifiers: (1) Scaled Euclidean, (2) Scaled Manhattan, and (3) first-order probabilistic (1-Prob).

Let $\{Y_1, Y_2, \ldots, Y_M\}$ denote M n-dimensional training data points. Let $x = \{x_1, \ldots, x_n\}$ denote an n-dimensional test point. A brief description of the verifiers follows.

Scaled Euclidean and Scaled Manhattan Verifiers. Let $y = \{y_1, \ldots, y_n\}$ be the *mean* vector computed from the feature vectors in the training set of a user. The *Scaled Euclidean* verifier calculates the verification score as:

$$\mathsf{D}_E(x, y) = \sqrt{\sum_{i=1}^{n} \frac{(x_i - y_i)^2}{\sigma_i^2}} \tag{1}$$

and the *Scaled Manhattan* verifier calculates the score as:

$$\mathsf{D}_M(x, y) = \sum_{i=1}^{n} \frac{|x_i - y_i|}{\sigma_i}, \tag{2}$$

where σ_i is the standard deviation of the i-th feature.

First-order Probabilistic Verifier (1-Prob). This verifier outputs the probability score $P(U_j, x)$, which is the joint probability of the j^{th} user U_j and the discrete test vector x. The score is calculated using the multiplication rule of probability as follows:

$$P(U_j, x) = P(U_j).\prod_{i=1}^{n} P(x_i|U_j) \tag{3}$$

where x_i is the i-th feature and probabilities of the type $P(x_i|U_j)$ are estimated from j^{th} user's training set using:

$$P(x_i|U_j) = \frac{r_i + mp}{r_j + m}, \tag{4}$$

where r_j is the number of training samples that belong to U_j, r_i is the number of samples in the i^{th}-bin of x_i for user U_j, and m is the m-estimate. We assumed all users are equally likely, so $P(U_j)$ is same for all users. To estimate probabilities of the type $P(x_i|U_j)$, we discretized each user's features into equal-width bins and created a probability histogram. Because Equation (3) is a product of probabilities, if the probability of one bin is zero, the entire score becomes zero. To avoid this, we used $\sum_{i=1}^{n} \log P(x_i|U_j)$ instead of $\prod_{i=1}^{n} P(x_i|U_j)$, and treated $\log 0$ as 0.

Choice of Verifiers. Scaled Manhattan and Scaled Euclidean are fairly popular verifiers in the biometric authentication literature (see for example, [25], [44], [28], [52] and [46]). The 1-Prob verifier implements a 1-class version of the Naive Bayes classifier and was chosen because of its simplicity and the surprisingly good performance of Naive Bayes in classification tasks despite its feature independence assumption [41].

Verification in PCA Space. We performed PCA to decorrelate features, thereby making the features compatible with the *feature independence* assumption of Scaled Euclidean, Scaled Manhattan, and 1-Prob verifiers. Furthermore, there is compelling evidence that PCA improves authentication accuracy for several biometric modalities (see, e.g., [48], [47]).

Though the feature decorrelation property of PCA is well known, we briefly review it for completeness. Let $Y_{M\times n}$ be a data-matrix containing M n-dimensional zero-centered training samples of a user. The covariance matrix of $Y_{M\times n}$ is $Y^T Y$. PCA transforms $Y_{M\times n}$ to $Y'_{M\times n}$ ($YA = Y'$), where $A_{n\times n}$ is a matrix containing n eigenvectors of $Y^T Y$. It is easy to show that $Y'^T Y'$, the covariance matrix of the transformed data is a *diagonal* matrix with eigenvalues $\lambda_1, \ldots, \lambda_n$ as diagonal elements.

B. Cryptographic Preliminaries

In this section, we review the cryptographic tools used in our constructions.

Garbled Circuit Evaluation. Originally proposed in [51], garbled circuits allow two parties to securely evaluate any function represented as a boolean circuit. The basic idea is that given a circuit composed of gates connected by wires, the server "garbles" the circuit by assigning two randomly chosen encryption keys, $\omega_{j,0}$ and $\omega_{j,1}$, to each wire w_j. These keys represent, respectively, 0 and 1. (In the garbled circuits literature, keys are usually referred to as *labels*.) The server then encrypts a truth table corresponding to each gate. Values in the table are also represented using labels, and each label

is encrypted with two keys, ω_{j,b_j} and ω_{l,b_l}, corresponding to the values on the gate's input wires. Therefore, computing the "output label" of each gate requires knowing two of its input labels.

The output of the circuit is encoded in its *output wires*, constructed and interpreted as follows. Let $\omega_{i,b} \in \{0,1\}^\kappa$ be the label of output wire i corresponding to output bit b, and s the number of output wires. The server selects a pair of random labels $(\omega_{i,0}, \omega_{i,1})$ for each output wire $i \in [1, s]$. Then, it builds a table $T = (\omega_{1,0}, \omega_{2,0}, \ldots, \omega_{s,0})$ and sends it to the client as part of the circuit. The client uses T to interpret the output of the circuit as follows: after evaluating the circuit, the client *decrypts* exactly one label ω_{i,b_i} for each output wire i, where $b_i \in \{0,1\}$ is i-th bit of the protocol's output. For each i, if ω_{i,b_i} appears in T then b_i is 0, otherwise 1.

Recent literature provides optimizations that reduce computation and communication overhead associated with circuit construction and evaluation. Kolesnikov et al. [30] describe a modification that permits XOR gates to be evaluated *for free*, i.e., there is no communication overhead associated with such gates and their evaluation does not involve cryptographic functions. With this modification, elements of a pair $(\omega_{i,0}, \omega_{i,1})$ might not be selected independently. Instead, $\omega_{i,0} = \omega_{i,1} \oplus r$ for some value r, constant throughout the circuit. Pinkas et al. [38] additionally give a mechanism for reducing communication complexity of binary gates by 25%: now each gate can be specified by encoding only three outcomes of the gate instead of all four. Finally, Kolesnikov et al. [29] improve the complexity of certain commonly used operations such as addition, multiplication, comparison, etc. by reducing the number of non-XOR gates.

Before garbled circuit evaluation, the client engages in oblivious transfer (OT) protocol, described next.

Oblivious Transfer. In 1-out-of-2 Oblivious Transfer (OT_1^2), one party (denoted as *sender*) has two strings m_0, m_1, and the other party (the *receiver*) has one bit (b) as its input. At the end of the protocol, the receiver learns m_b and the sender learns nothing. Similarly, in 1-out-of-N OT the receiver obtains one of the N strings held by the sender.

In this paper we use an efficient implementation of OT_1^2 from [35] as well as techniques from [24] that reduce a large number of OT protocol executions to κ (where κ is the security parameter).

Homomorphic Encryption. Our constructions use a semantically secure additively homomorphic encryption scheme. In an additively homomorphic encryption scheme, $\mathsf{Enc}(m_1) \cdot \mathsf{Enc}(m_2) = \mathsf{Enc}(m_1 + m_2)$, which also implies that $\mathsf{Enc}(m)^a = \mathsf{Enc}(a \cdot m)$. While any encryption scheme with the above properties (such as the well known Paillier encryption scheme [37]) suffices for the purposes of this work, we use the construction due to Damgård et al. [11], [10] (DGK) because it is fast and it produces small ciphertexts. In the rest of the paper, we use $[\![m]\!]$ to refer to the DGK encryption of message m under the server's public key. (The server is assumed to have access to the corresponding decryption key.)

Our privacy-preserving protocol for computing Scaled Manhattan distance requires privacy-preserving comparison of two encrypted values. For this task, we rely on the comparison protocol of Erkin et al. [13].

Representation of Values in the Encrypted Domain. All inputs to our privacy-preserving protocols are integer values, so we mapped real-valued inputs to $2^e - 1$ equal-width bins corresponding to integers in the interval $[-2^{e-1} + 1, 2^{e-1} - 1]$. We represented integers in two's-complement notation throughout our protocols. The size of the discretized domain was more than e bits, to account for addition and multiplication operations throughout the protocol.

We used the following formula for discretization:

$$\mathsf{discretize}_{e,\mathsf{F}}(x_i) = \left\lceil \frac{(2^e - 1) \cdot x_i}{\mathsf{max}_\mathsf{F}} \right\rceil \tag{5}$$

where F is the feature being discretized, x_i is an instance in F and max_F is the maximum of absolute values in F.

IV. Privacy-Preserving Protocols

We now introduce our privacy-preserving protocols for secure outsourcing of authentication. Our protocols are divided in two phases: *enrollment* and *verification*. During enrollment, a mobile device (*client*) is used to acquire biometric signals from the user. With the help of an *enrollment server*, the biometric signals are processed to build the user's encrypted template. Processing is performed either: (a) by the client—here, the enrollment server simply binds the user's identity with the encrypted template; or (b) by the enrollment server—in this case, the computational burden is offloaded from the client at the cost of revealing the template to the server.

During enrollment, all parties involved (the user, the mobile device, and the enrollment server) are trusted. In practice, this is necessary to ensure that data collection is performed correctly.

In all our protocols based on garbled circuits, the client needs an additive share of user's template for authentication. To lower the amount of information stored on the client, the user's additive share r can be encrypted under the client's symmetric key c during enrollment, and stored as $\mathsf{AES}_c(r)$ on the authentication server. Before each authentication attempt, the authentication server sends $\mathsf{AES}_c(r)$ to the client. This allows the client to store only one key, c, instead of the user's share. We do not further elaborate on this modification, as its implementation is straightforward and the cost is insignificant compared to the rest of the protocol.

A. Modification of the Garbled Circuits Protocol to Achieve Security Against Malicious Clients

To achieve security against malicious client, we implement three modifications to the garbled circuits protocol:

(1) Output Labels Are Selected Independently. The security of our protocol against malicious clients relies on the security of the garbled circuits in the HBC model. To our knowledge, the only available proof of security for garbled circuits against HBC adversaries is that of Lindell et al. [32]. This proof requires that output labels are selected independently at random—we modified [22] to do this.

(2) Client Does Not Learn T. The security of our protocol relies on the assumption that the client learns only one output label for each pair $(\omega_{i,0}, \omega_{i,1})$. For this reason, we modify garbled circuits so that the server does not send T to the client. Instead, the client sends the output labels to the server.

(3) Server Aborts on Invalid Output Labels. In the HBC model, the labels returned by the client are always valid because all parties faithfully follow the protocol. Therefore, the behavior of the server when labels are invalid need not be defined. However, in our setting we assume that the client can arbitrarily deviate from the protocol.

Let $M = (\omega'_1, \ldots, \omega'_s)$ be a s-tuple of strings in $\{0,1\}^\kappa$. M is a *valid* tuple of output labels iff there exist $(b'_1, \ldots, b'_s) \in \{0,1\}^s$ s.t. $M = (\omega_{1,b'_1}, \ldots, \omega_{s,b'_s})$. The server verifies if the labels returned by the client are valid. If not, the server aborts.

The server incurs negligible overhead for checking the validity of the labels—for each label ω'_i that is not in T (i.e., ω'_i does not encode 0), the server verifies that $\omega'_i = \omega_{i,1}$.

Modifications (2) and (3) are implemented by at least one existing garbled circuits instantiations [22], although *not* for improving the security of the protocol.

B. Scaled Manhattan and Scaled Euclidean Protocols Secure Against Malicious Client

We implemented the Scaled Manhattan and Scaled Euclidean verifiers with our modifications to garbled circuits.

Enrollment Phase. A user's scaled and discretized template $y = (y_1, ..., y_n)$ is computed from the mean template y', as is described Figure 1. Standard deviations $\sigma = (\sigma_1, ..., \sigma_n)$ used for scaling are taken as the mean of standard deviations of all users in the training set. The same σ is used for all users and is made public. The additive share r of the user's template can be stored either on the client, or encrypted as $\mathsf{AES}_c(r)$ and stored on the server.

Verification Phase. The server's input is the additive share of the biometric template $-(y + r)$ and the client's input is $x + r$, where x is the current biometric (test) sample and r is the additive share of the template. The layout of the verification circuit is in Figure 2. The client and server run the verification circuit on their inputs. The server obtains the Manhattan/Euclidean distance between x and y. The server then makes an authentication decision based on the distance. In case of Scaled Euclidean, the circuit outputs the *square* of the distance, to avoid the costly computation of square root. Because square root is a monotonous function, this does not impact the outcome of authentication.

One-bit Output Instead of Distance. Our protocols can be modified to output an authentication decision (i.e., whether or not the distance between the biometric template and the client's input is smaller than a threshold). To achieve this, we added a subcircuit that computes the "smaller than" functionality. This subcircuit takes the (unencrypted) threshold and the (garbled) output of the sum subcircuit of Figure 2 as input. This modification improves client's privacy because the server does not learn any information besides the authentication decision.

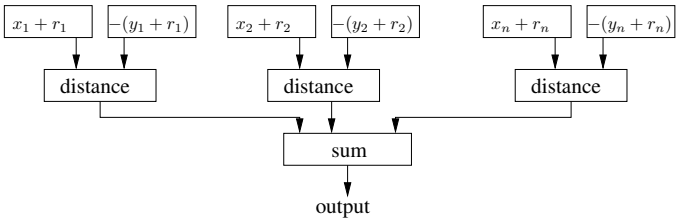

Fig. 2: Generic authentication circuit for Scaled Manhattan/Euclidean distance. The distance component is instantiated using Manhattan or Euclidean distance subcircuit. Output of the circuit is the verification score.

C. Protocols Secure in the HBC Model

Scaled Manhattan and Scaled Euclidean. We summarize the privacy-preserving protocols for Scaled Manhattan and Scaled Euclidean in the Appendix (Figures 6 and 7, respectively). Full description and analysis of the protocols is available in [19]. Next, we discuss our 1-Prob protocol.

1-Prob. We designed a new 1-Prob protocol as a way to implement a fast, yet accurate verifier. Our protocol is secure in the HBC model, and is implemented using homomorphic encryption instead of garbled circuits.

Reasons for not implementing a 1-Prob protocol with garbled circuits to secure against a malicious client follow. Under this setting, it is difficult to guarantee both template protection *and* verification correctness. In 1-Prob, the client must select a set of probabilities from a matrix $\mathcal{Y}'$, which represents the user template. Clearly $\mathcal{Y}'$ cannot be shared with the client, since it would disclose the entire biometric information stored in the system for a particular user. Similarly, $\mathcal{Y}'$ cannot be revealed to the authentication server.

As an alternative, the server could store an additive share of $\mathcal{Y}'$ (say, $\overline{\mathcal{Y}}$), where a different random value $r_{i,j}$ is added to each element $\mathcal{Y}'_{i,j}$. The input to the authentication protocol would include $\overline{\mathcal{Y}}$ for the server, and $r_{i,j}$-s for the client. Although this approach provides template protection, it also allows a malicious client to trivially authenticate without knowing the user's biometric. The client could, for example, replace each random value $r_{i,j}$ with $r'_{i,j} = r_{i,j} - \delta$ for some $0 < \delta \leq 1/n$. The output of the protocol would therefore be $n\delta + \sum_{i=0}^{n} \mathcal{Y}_{i,j}$ instead of $\sum_{i=0}^{n} \mathcal{Y}_{i,j}$, allowing the malicious client to authenticate with overwhelming probability. Although techniques based on zero-knowledge proofs can be used to prevent this attack, the additional overhead from computation, communication, and number of rounds would make the 1-Prob protocol too expensive for mobile devices (and thereby defeat our primary motivation for using 1-Prob). Next, we discuss our design of 1-Prob protocol.

In the enrollment phase, biometric samples are collected and processed. Each sample is represented as an n-dimensional feature vector. For each feature, we define ℓ bins, which correspond to different ranges of values for that feature. For each feature, bins contain the logarithm of the probability associated with the corresponding range. The output of this

Biometric template	$y' = (y'_1, \ldots, y'_n)$	Vector y' is the mean of user's biometric measurement from the training session.
Scaled and Discretized template	$y = (d_{e,1}(y'_1/\sigma_1), \ldots, d_{e,n}(y'_n/\sigma_n))$	Discretization ranges are the same for all users, per feature.
Final template for enrollment	$y + r = (d_{e,1}(y'_1/\sigma_1) + r_1, \ldots, d_{e,n}(y'_n/\sigma_n) + r_n)$	Addition is modulo 2^e. The vector r is selected uniformly at random.

Fig. 1: Step by step overview of processing user's biometric template $y' = (y'_1, ..., y'_n)$ during enrollment for the Scaled Manhattan and Scaled Euclidean protocols without PCA. Randomization is omitted for protocols secure in the HBC model.

Input: Client: sample $x = (x_1, \ldots, x_n)$, server's public key and decryption key for $\mathsf{AES}_c(\cdot)$; Server: encrypted $n \times m$ probability template matrix $\mathsf{AES}_c(\mathcal{Y}) = \mathsf{AES}_c([\![-\log p_{1,1}]\!], \ldots, [\![-\log p_{n,\ell}]\!])$ and decryption key for the homomorphic encryption scheme.
Output: The server learns $\mathsf{Score}_{NB}(x, \mathcal{Y})$.
Protocol steps:

1) The server sends $\mathsf{AES}_c(\mathcal{Y})$ to the client, which decrypts it, obtaining $[\![-\log p_{0,0}]\!], \ldots, [\![-\log p_{n,\ell}]\!]$.
2) For $i = 1, \ldots, n$ Client selects $[\![-\log p_{i,x_i}]\!]$ from $\mathcal{Y}$ and computes:

$$[\![d]\!] = \left[\!\!\left[\left(\sum_{i=1}^{n} -\log p_{i,x_i} \right) \right]\!\!\right] = \prod_{i=1}^{n} [\![-\log p_{i,x_i}]\!]$$

3) Client randomizes $[\![d]\!]$ as $\overline{[\![d]\!]} = [\![d]\!][\![0]\!]$.
4) Client sends $\overline{[\![d]\!]}$ to the server, which decrypts it and outputs $\mathsf{Score}_{NB}(x, \mathcal{Y})$ as d.

Fig. 3: Computation of privacy-preserving 1-Prob score

Input: Client: sample $x = (x_1, \ldots, x_n)$, server's public key, and decryption key for $\mathsf{AES}_c(\cdot)$; Server: encrypted $n \times n$ Eigen matrix $\mathsf{AES}_c(A) = \mathsf{AES}_c([\![a_{1,1}]\!], \ldots, [\![a_{n,n}]\!])$, encrypted n-dimensional template vector in the PCA space $\mathsf{AES}_c(y') = \mathsf{AES}_c([\![y'_1]\!], \ldots, [\![y'_n]\!])$ and decryption key for the homomorphic encryption scheme.
Output: The client learns encryption of x transformed in PCA space.
Protocol steps:

1) The server sends $\mathsf{AES}_c(y')$ and $\mathsf{AES}_c(A)$ to the client, which decrypts it, obtaining $[\![y'_1]\!], \ldots, [\![y'_n]\!]$ and $[\![a_{1,1}]\!], \ldots, [\![a_{n,n}]\!]$.
2) Let $([\![x'_1]\!], \ldots, [\![x'_n]\!])$ denote the component-wise encryption of Ax. For $i = 1, \ldots, n$ the client computes:

$$[\![x'_i]\!] = \left[\!\!\left[\left(\sum_{j=1}^{n} a_{i,j} \cdot x_j \right) \right]\!\!\right] = \prod_{j=1}^{n} [\![(a_{i,j})]\!]^{x_j}$$

3) For $i = 1, \ldots, n$ the client computes

$$[\![x''_i]\!] = [\![x'_i]\!] \cdot [\![y'_i]\!]^{-1} = [\![x'_i - y'_i]\!]$$

4) Client outputs x'', which corresponds the encryption of x transformed in PCA space.

Fig. 4: Computation of privacy-preserving PCA transformation

process is the following $n \times \ell$ matrix:

$$\mathcal{Y}' = \begin{bmatrix} -\log p_{1,1} & \cdots & -\log p_{1,\ell} \\ \vdots & \ddots & \\ -\log p_{n,1} & & -\log p_{n,\ell} \end{bmatrix} \tag{6}$$

where $\log p_{i,j}$ is the logarithm of the probability that feature i falls within bin j. (If computation is performed in PCA space, feature vectors are pre-processed as discussed in Section III-A.)

Matrix $\mathcal{Y}$ is computed by encrypting each element of $\mathcal{Y}'$ separately using homomorphic encryption. That is, the element on the i-th row, j-th column of $\mathcal{Y}$ is $[\![-\log p_{i,j}]\!]$. Finally, $\mathcal{Y}$ is encrypted as $\mathsf{AES}_c(\mathcal{Y})$ (c known to the client) and sent to the authentication server. Since the server does not have access to the decryption key for $\mathsf{AES}_c(\cdot)$, it cannot extract any information from $\mathsf{AES}_c(\mathcal{Y})$ (besides its size).

Verification phase is illustrated in Figure 3.

V. Privacy-Preserving Verification in PCA Space

In addition to the results reported in this paper, PCA's positive impact on authentication accuracy has been evidenced in many behavioral modalities—see, e.g., [47]. For the Scaled Manhattan and for the 1-Prob verifier, PCA de-correlates features, thus bringing them closer to 1-Prob's assumption that features are independent.

From a privacy perspective, however, the eigenmatrix A obtained with PCA reveals important information about a user's template, such as: (a) the relative importance of features, especially in eigenvectors corresponding to top eigenvalues and (b) the structure of the feature covariance matrix. Note that concealing the content of A while revealing the authentication attempt in eigenspace will not resolve the privacy issue: if the adversary obtains multiple instances of the authentication vector and its counterpart in eigenspace, it can estimate A.

Scaling in the PCA space has to be done differently compared to feature space:

1) The PCA transformation matrix is different for each user,

so the average of the standard deviations in PCA space is not meaningful. Therefore, we use standard deviation calculated user-wise.

2) The eigenvalues corresponding to individual Principal Components (PCA space dimensions) are again different for each user, and therefore are calculated user-wise.

The above items are not published, unlike in the Scaled Manhattan/Euclidean verifiers in the feature space. Instead, these are "embedded" in the PCA matrix rows/columns, respectively.

The number of basis vectors/elements in the PCA space m is lower than or equal to the number of vectors in the feature space (now denoted as n).

A. PCA with protocols based on Garbled Circuits

The PCA subcircuit is the same for Scaled Manhattan and Scaled Euclidean verifiers, and is illustrated in Figure 5. The transformation matrix A and the biometric template y are additively shared between the server and client as follows. During enrollment, the client generates a random matrix $R \leftarrow \mathbb{Z}_{2^e}^{m \times n}$ and a random vector $s \leftarrow \mathbb{Z}_{2^e}^{n}$. The client stores $(A+R)$ and s, while the server stores $-R$ and $-(y+s)$.

During authentication, the client acquires biometric sample x. It then evaluates the circuit in Figure 5. The client input is $(x+s)$ and $(A+R)$ while the server's is $-(y+s)$ and $-R$. The circuit computes $A = (A+R) - R$ and the zero-centered biometric sample $x - y = (x+s) - (y+s)$; then, it maps $x-y$ to PCA space, obtaining $A(x-y)$, and uses the circuit in Figure 2 to compute the Manhattan/Euclidean distance between $A(x-y)$ and the 0-vector.

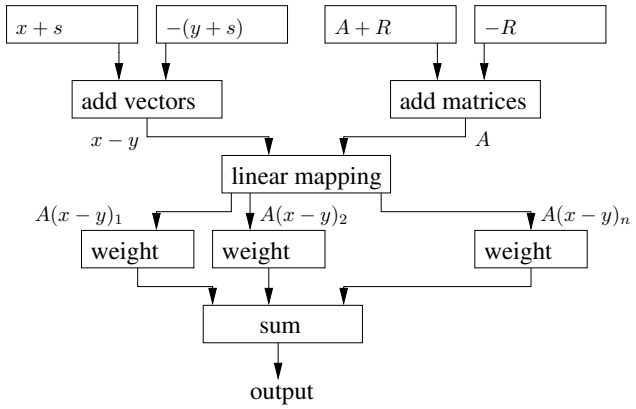

Fig. 5: Circuit for authentication using Scaled Manhattan/Euclidean distance in PCA space. The Scaled Manhattan and Scaled Euclidean circuits differ in the *weight* subcircuit.

B. PCA with protocols based on Homomorphic Encryption

To prevent information leakage via A or Ax, we modify our protocols so that the client's input Ax is provided in encrypted form. The unencrypted version of Ax is never observed by any of the parties. Additionally, we design a short protocol, shown in Figure 4, to compute Ax in encrypted form. Below, we detail the changes to the verification protocols that must be implemented to allow distance computation in PCA space. To zero-center user's biometric sample $y = (y_1, \ldots, y_n)$ using the corresponding biometric sample $x = (x_1, \ldots, x_n)$ and transform it to PCA space defined by transformation matrix A, we need to compute $A(x-y)$. In the encrypted domain, it is more convenient to use the linearity of matrix multiplication and compute $Ax - Ay$ instead. A and Ay is stored encrypted on the server and sent to client.

Scaled Euclidean. In order to compute $[\![-2\alpha_i x_i y_i]\!]$ in Figure 7, Step 2, client and server perform a short interactive protocol [13] to calculate the product of $-x_i$ and $2\alpha_i y_i$ in the encrypted domain.

Scaled Manhattan. No change is required to the Scaled Manhattan protocol because it operates on encrypted inputs for both client and server.

1-Prob. During enrollment, probability matrix $\mathcal{Y}'$ is computed using biometric samples in PCA space. Each row is randomly shuffled, and two values $l_{i,j}$ and $h_{i,j}$ (representing the lower and higher limit of the bin corresponding to $\log p_{i,j}$) are associated with each element of $\mathcal{Y}'$. $l_{i,j}$ and $h_{i,j}$ are encrypted separately using homomorphic encryption. Ciphertexts $[\![l_{i,j}]\!]$ and $[\![h_{i,j}]\!]$ are then encrypted, together with $\mathcal{Y}$ ($\mathcal{Y}$ is the encryption of $\mathcal{Y}'$ as discussed in Section IV-C), using $\mathsf{AES}_c(\cdot)$. During verification, the client transforms vector x to PCA space using the protocol in Figure 4, obtaining $Ax = (x'_1, \ldots, x'_n)$. Then, the client decrypts $\mathsf{AES}_c(\mathcal{Y}, l_{i,j}, h_{i,j})$ and, for all i, j, it interacts with the server to compute $c_{i,j} = (x'_i \overset{?}{<} h_{i,j}) + (x'_i \overset{?}{>} l_{i,j}) - 1$. Note that $c_{i,j}$ is 1 if x'_i falls within bin i, j, and 0 otherwise. Then the client computes $\left[\!\left[\sum_{j=1}^{n} \left(-\log p_{i,x'_i}\right)\right]\!\right] = \prod_{j=1}^{n} [\![-\log(p_{i,j}) \cdot c_i]\!]$. In order to compute $-\log(p_{i,j}) \cdot c_i$ in the encrypted domain, client and server perform the short interactive multiplication protocol in [13].

VI. Security Analysis

In this section, we analyze the security of our protocols based on garbled circuits. We show that the protocols are secure against a malicious client.

Security against a HBC server is shown in [32]. We argue that our modifications to garbled circuits, presented in Section IV-A, do not invalidate the proof in [32]. In particular, modification (1) does not affect the analysis in [32], which assumes that labels are selected independently. Modification (2) also does not affect the security of the protocol: to the server, knowing the protocol's output is equivalent to knowing the corresponding output labels. Finally, modification (3) is irrelevant when the client follows the protocol faithfully.

Security Model. We use the term *adversary* to refer to insiders, i.e., protocol participants. This allows us to model both misbehaving *and* compromised clients and servers. We do not consider outside adversaries, since their actions can be mitigated via standard secure protocols, such as TLS.

Informally, a HBC party follows the prescribed protocol behavior, and might try to learn additional information from the protocol transcripts. Formally [17]:

Definition 1. *Let P_1 and P_2 participate in protocol π that computes function $f(\mathsf{in}_1, \mathsf{in}_2) = (\mathsf{out}_1, \mathsf{out}_2)$, where*

in_i *and* out_i *denote* P_i*'s input and output, respectively. Let* $\text{VIEW}_\pi(P_i)$ *denote the view of participant* P_i *during the execution of protocol* π*.* P_i*'s view is formed by its input, internal random coin tosses* r_i*, and messages* $m_1, \ldots, m_t$ *passed between the parties during protocol execution:* $\text{VIEW}_\pi(P_i) = (\mathsf{in}_i, r_i, m_1, \ldots, m_t)$*. We say that protocol* π *is secure against HBC party* P_j*,* $j \in \{1, 2\}$ *if for party* P_j *there exists a probabilistic polynomial time simulator* S_j *such that* $\{S_j(\mathsf{in}_j, f_j(\mathsf{in}_1, \mathsf{in}_2))\} \equiv \{\text{VIEW}_\pi(P_j), \mathsf{out}_j\}$*, where "*$\equiv$*" denotes computational indistinguishability.*

We use the standard "ideal world" vs. "real world" definition of security for malicious adversaries, which we briefly summarize here. (Complete definition is available in Section 7.2.3 of [17].) A malicious party can use any PPT algorithm during protocol execution. The protocol is deemed secure if all attacks that are possible during protocol executions (i.e., in the "real world") can also be executed when the protocol is replaced by a trusted third party (in the "ideal world"):

Definition 2. *We say that protocol* π securely computes f in the presence of malicious party P_2*, if for every PPT algorithm* $\mathcal{A}$ *that is admissible for the real model, there exists a PPT algorithm* $\mathcal{B}$ *that is admissible in the ideal model such that*

$$\{\text{IDEAL}_{f,(P_1(z),\mathcal{B}(z))}(x, y)\} \equiv \{\text{REAL}_{\pi,(P_1(z),\mathcal{A}(z))}(x, y)\}$$

Security against malicious clients is stated in Theorem 1.

Theorem 1. *Assuming that the oblivious transfer protocol is secure against malicious adversaries, and that the encryption scheme has indistinguishable encryptions under chosen plaintext attacks, our modified garbled circuit protocol is secure in the random oracle model with respect to a malicious client.*

Proof (sketch). We first show that, with our modified garbled circuits, a malicious client can construct a valid message for the server only by faithfully executing the circuit. Then, we show that the client cannot extract any information on the server's inputs and outputs from the protocol execution. We conclude by showing that these two properties allow us to build a simulator that uses a malicious client to implement an attack in the ideal world.

Valid Message by the Client Can Only be Constructed by Faithfully Executing the Protocol. Recall that the message sent from the client to the server is $M = (\omega'_1, \ldots, \omega'_s)$. All encryption keys are selected independently. All ω_{i,b_i} are also selected independently, and the encryption scheme is semantically secure; therefore, the client can obtain valid output labels only by decrypting the corresponding ciphertexts.

Keys used for encrypting output labels are also encrypted, and so are all previous "layers" of the circuit. Hence, to obtain the decryption keys for the output labels, decryption must start at the topmost layer of the circuit, using the input wire keys. The client needs one key corresponding to the server's input and one key corresponding to client input for each input gate. Server's keys are made public, while client's keys are exchanged using OT. Because OT is secure in the malicious model, the client only learns one client key per input gate. Therefore, the client can only decrypt one input label per each gate/wire in the circuit, including output wires.

Because the labels were chosen uniformly at random by the server, the probability of guessing a valid label without decrypting the output label is negligible.

When M was not constructed according to the protocol, it will be invalid with overwhelming probability, and the server will abort the protocol.

Server Input Privacy. If the client is faithfully following the protocol, security in HBC model of garbled circuits implies that the client does not learn any information about the server's input. We have shown that the client can either follow the protocol, or the server aborts with overwhelming probability and without sending additional messages to the client. Therefore, our modified garbled circuit protocol provides server input privacy against malicious client.

Correctness. Correctness follows from the fact that the client either follows the protocol (then, the result is correct), or the server aborts with overwhelming probability and there is no output.

Server Output Privacy. The client either follows the protocol or the server aborts. In the first case, the security of garbled circuits in HBC implies the server output privacy.

Real vs. Ideal World Security. Let C^* be a malicious client in the real world. From server input/output privacy and correctness, we know that C^* does not learn any information about the server's input and output. C^* can influence the server's output only via its input.

For any malicious client C^*, we show that it is possible to construct a simulator S that uses C^* to implement the same attack in the ideal world. Using the OT in [35], S can extract the input of C^* exploiting the random oracle. S can then present the same input with the trusted third party in the ideal world.

From the correctness of the protocol in the real world, the output of the trusted third party in the ideal world must be the same as the output of the protocol. We conclude that C^*'s strategy is not an attack on the protocol, as it can be simulated in the ideal world. □

VII. Performance Analysis

We evaluated the overhead of our protocols for continuous authentication of smartphone users. Running the protocols requires choosing parameters that impact communication and computation (for example, number of features and level of discretization). Optimal parameters are dataset- and features-specific. Because these parameters were not disclosed in existing smartphone authentication literature, we determined them by performing biometric experiments on two publicly available datasets, Dataset-LTU [46] and Dataset-Frank [14].

Dataset Description. Dataset-LTU was collected from 190 users, in two separate sessions for each user. We used the first session for training, and the second for testing. The dataset is divided into four partitions, based on phone orientation (landscape or portrait) and the orientation of strokes on the screen (horizontal or vertical). Although we ran experiment on all four subsets, for the sake of presentation we include two most representative settings: landscape-horizontal swipes (646

genuine scores) and portrait-vertical swipes (2748 genuine scores). These settings illustrate where PCA has a significant impact on error rates and where it does not, respectively. Dataset-Frank was collected in one session only, so we split the feature vectors belonging to each user into two sets: training and testing. We used the first 90 percent of a user's vectors to perform 1-class training. We used the remaining 10 percent (188 genuine scores) for zero-effort testing (i.e., impostor attempts of one user are genuine attempts of the remaining users). In Supplement [53], we give further details on our biometric experiments.

In the rest of this section, we report biometric accuracies, protocol costs, and tradeoffs.

A. Biometric Accuracy

Here we report the equal error rates (EERs) achieved on Dataset-LTU and Dataset-Frank. Note that the main purpose of our experiments was to observe the impact of discretization and PCA on ERRs.

Impact of PCA. In Table I, we summarized EERs with and without PCA, with no discretization. The use of PCA improves the EER by about 3% in the 'portrait-vertical' subset of Dataset-LTU, as well as in Dataset-Frank. For the 'landscape-horizontal' subset of Dataset-LTU, PCA does not improve EERs. These results are in agreement with existing research in other behavioral biometrics (e.g., [48], [47]), and confirm that PCA is worth considering and can improve the authentication accuracy—although not in all circumstances.

Impact of Discretization. Our cryptographic building blocks require that all operations are performed on *integers* rather than floating point numbers. Therefore, we evaluated the biometric accuracy of our protocols with different discretization (e) values, corresponding to number of bits used for discretization of protocol inputs.

In order to achieve roughly the same authentication accuracy as with floating point arithmetics with the Dataset-LTU, $e = 8$ was sufficient for 1-Prob without PCA, while $e = 12$ was needed for all other verifiers with and without PCA. With Dataset-Frank, floating-point-arithmetic-precision was achieved with the same discretization parameters in all configurations, except for Scaled Manhattan and Scaled Euclidean in PCA space, which required $e = 20$.

B. Protocol Performance

Precomputation for Garbled Circuits. In continuous authentication, an *authentication window* represents the time slot in which a biometric sample is collected. The size of the window (plus time taken by the verification algorithm) specifies the unavoidable delay in the authentication process. The precomputation phase of our protocols runs during the authentication window. Therefore, precomputation does not increase the authentication delay, as long as the time taken for precomputation is within the authentication window.

The most expensive part of the OT protocol can be performed ahead of time by the client and the server [5]. Moreover, the server can construct the garbled circuit independently from the client. This reduces the cost of the protocols once authentication data is available. In the subsequent *online* phase, the parties complete the OT and the client evaluates the circuit.

We report the time and bandwidth overhead for the precomputation and authentication phases separately in Tables II, III and IV. Communication required for the precomputation phase is constant and consist of 22 KB, as is required for the OT. We only include total communication time in the tables.

Asymptotic Complexity. The protocols secure against malicious clients are implemented using garbled circuits, so their complexity (both time and space) is linear in the number of the circuit gates. Let n denote the number of features, m the number of PCA features (in case PCA is used) and e the number of bits per feature or element used for discretization. The Scaled Manhattan protocol without PCA has complexity $\mathcal{O}((e + \log(n)) \cdot n)$, and our Scaled Euclidean protocol has higher complexity $\mathcal{O}((e + \log(n))^2 \cdot n)$ due to the multiplication required for computing Euclidean distance. The Scaled Manhattan protocol with PCA has complexity $\mathcal{O}((e+\log(n)+\log(m))^2 \cdot mn)$ and the Scaled Euclidean protocol with PCA has complexity $\mathcal{O}((e+\log(n)+\log(m))^4 \cdot mn)$.

The computational complexity of our Scaled Manhattan protocol in PCA space based on homomorphic encryption is $\mathcal{O}(n)$ for both client and server, where n is the number of features. The complexity of our Scaled Euclidean protocol is $\mathcal{O}(n)$ for the client and $\mathcal{O}(1)$ for the server, i.e., the cost for the server does not depend on the number of features. In particular, the server only needs to decrypt one ciphertext regardless of the number of features used for authentication.

In terms of communication, both protocols require the server to send $\mathsf{AES}_c(\mathcal{Y})$ to the client. Additionally, the protocol that computes Scaled Manhattan distance also requires the parties to run multiple instances of the comparison and multiplication protocols, both of which exchange constant number of messages. As an optimization, if the client caches a copy of $\mathsf{AES}_c(\mathcal{Y})$, then the communication cost of computing the Scaled Euclidean distance is reduced to $\mathcal{O}(1)$. This latter optimization can be used for all protocols, therefore reducing the amount of data exchanged between client and server—or entirely removing the need for further communication, in case of the protocol in Figure 4.

The cost of our privacy-preserving 1-Prob protocol is $\mathcal{O}(1)$ for the server, and $\mathcal{O}(n)$ for the client. When performing the same protocol in PCA space, the cost increases to $\mathcal{O}(n\ell)$ for both client and server, where ℓ is the number of bins. For this reason, the protocol, although *feasible* for the wide range of parameters tested in this paper, is *practical* only for a relatively small number of features/bins when run on a commodity smartphone.

Protocols Implementation and Evaluation. We used the Huang et al. [22] implementation of Garbled Circuits, written in Java, on a Samsung Galaxy S4 smartphone (1.9 GHz Krait 300 CPU) and Macbook Pro laptop as server (2.6 GHz Intel Core i5 CPU). The protocols based on homomorphic encryption were implemented in Java using the BigInteger library. All experiments were run in a single thread.

Performance results in this section are reported, for all

	Scaled Manhattan		Scaled Euclidean		1-Prob	
	no PCA	PCA	no PCA	PCA	no PCA	PCA
Dataset-LTU portrait-vertical	0.314 (26)	0.262 (22/28)	0.305 (26)	0.269 (22/28)	0.344 (28)	0.261 (23/24)
Dataset-LTU landscape-horizontal	0.231 (27)	0.231 (15/28)	0.243 (23)	0.245 (15/28)	0.215 (26)	0.248 (12/24)
Dataset-Frank	0.213 (3)	0.183 (26/27)	0.210 (8)	0.183 (9/12)	0.267 (15)	0.241 (26/27)

TABLE I: Equal error rates for three verifiers when features are not discretized. For each verifier, we report the best EER achieved across different parameter settings (see Supplement [53] for details). We give the number of features in parenthesis. For PCA, we report the number of features as *number of PCA features/number of original features used to generate PCA matrix*. Except for the 'landscape-horizontal' subset of Dataset-LTU, PCA improves the authentication accuracy by at least 2.5%.

	Protocols secure against malicious client			Honest-but-curious model protocols	
	precomp.	auth.	comm.	auth.	comm.
	Smartphone client			*Smartphone client*	
Manhattan (n=8)	1.6s	1.5s	0.04 MB	0.85s	4 MB
Manhattan (n=16)	2.5s	3.2s	0.06 MB	1.8s	5 MB
Manhattan (n=28)	3.5s	5.6s	0.09 MB	2.2s	5.5 MB
Euclidean (n=8)	13.2s	39s	0.46 MB	0.5s	2.2 MB
Euclidean (n=16)	27.9s	105.5s	0.95 MB	1.0s	4.2 MB
Euclidean (n=28)	52.8s	237.6s	1.66 MB	2.0s	8 MB
1prob (n=8)	n/a			34ms	16.2 MB
1prob (n=16)				35ms	32.5 MB
1prob (n=28)				36ms	57.5 MB

TABLE II: Performance of our protocols without PCA. We benchmarked the protocols on a smartphone client and Macbook Pro server. All results are for $e = 12$ bits discretization. We only implemented the 1-prob verifier in the HBC model, as its main purpose is low-latency authentication.

	Protocols secure against malicious client			Honest-but-curious model protocols	
	precomp.	auth.	comm.	auth.	comm.
	Smartphone client			*Smartphone client*	
Manhattan (n=4, m=3)	16.4s	60.2s	0.7 MB	0.85s	5.4 MB
Manhattan (n=8, m=7)	151.1s	838s	3.6 MB	1.31s	12.2 MB
Manhattan (n=10, m=9)	363.6s	2051s	6.1 MB	1.86s	22.9 MB
1prob (n=8, m=10, 10 bins)	n/a			12.2s	18.2 MB
1prob (n=16, n=27, 10 bins)				18.7s	40.5 MB
1prob (n=28, n=27, 10 bins)				45.9s	174.0 MB

TABLE III: Performance of our protocols with PCA. As the amount of memory available to applications is limited on Android, we evaluated our Scaled Euclidean protocol and the Scaled Manhattan protocol for higher amount of features on a laptop client.

	Protocols secure against malicious client			Honest-but-curious model protocols	
	precomp.	auth.	comm.	auth.	comm.
	Laptop client			*Laptop client*	
Manhattan (n=10, m=9)	7.1s	3.7s	6.2 MB	0.1s	5.4 MB
Manhattan (n=16, m=15)	20.0s	12.0s	17.3 MB	0.3s	12.2 MB
Manhattan (n=28, m=27)	87.2s	71.2s	61.1 MB	0.4s	22.9 MB
Euclidean (n=10, m=9)	31.0s	27.1s	21.4 MB	23ms	14.2 MB
Euclidean (n=16, m=7)	83.4s	70.3s	57.1 MB	37ms	40.3 MB
Euclidean (n=28, m=27)	642s	655s	187.3 MB	65ms	129.4 MB

TABLE IV: Generalization of our results—performance for protocols with PCA on a *laptop* client. The performance gain for the Scaled Manhattan protocol with PCA secure against malicious client is more than two orders of magnitude faster on the laptop compared to the smartphone, because there are no memory restrictions and no need for garbage collection. The performance of HBC Scaled Manhattan protocol with PCA on laptop is less than five times faster compared to the smartphone.

protocols, when input was discretized with $e = 12$ bits, both with and without PCA. We evaluated the performance of our protocols for n varying from 8 to 28 features. Performance of our protocols is summarized in Tables II, III and IV.

Performance without PCA. For hybrid-model protocols, Scaled Manhattan without PCA is practical for continuous user authentication of smartphones (running time for authentication is from 1.5 to 5.6 seconds with less than 100 KB data transfer, for 8 to 28 features). The Scaled Euclidean protocol without PCA is feasible, but not as practical as Scaled Manhattan, with running time of the authentication protocol of 39 seconds to 238 seconds and 0.5 MB to 1.6 MB communication for the same number of features. This difference is mainly due to the expensive multiplication operations in the Euclidean distance computation.

While the protocols secure in the HBC model have very low running time for authentication (between 0.85 and 2.2 seconds) the communication cost is higher. With Scaled Manhattan, the communication overhead is over 50 times higher compared to our hybrid-model protocol. The difference is not as large with Scaled Euclidean protocols.

As a sidenote, we provide the performance results without

PCA on a *laptop* client in the Supplement [53].

Performance with PCA. The number of features in PCA space was chosen as $m = n - 1$. In practice, we expect applications to use less than m PCA features. Therefore, our measurements provide an upper bound on the computation and communication cost of our garbled circuit PCA protocols.

Smartphone devices have limited amount of memory compared to desktop computers and the Android platform limits the amount of memory available to applications. The per-application memory limit on our client device is 512 MB; when the amount of memory used by the application approaches this limit, garbage collection consumes a significant portion of the application's running time. Even though the available memory suffices for all our experiments without PCA, we were unable to complete experiments with more than 9 PCA features (computed from 10 original features) with Scaled Manhattan verifier and to complete any experiments with the Scaled Euclidean verifier.

It is important to note that the amount of memory available in smartphones has been increasing staidly, often doubling every two years. By extrapolating from this trend, we can assume that protocols limitations due to memory availability will become less relevant in the near future.

As an additional point of comparison, we evaluated the performance of our protocols using the MacBook Pro laptop also as client. The Scaled Manhattan protocol with PCA is roughly four times faster on the laptop, compared to the smartphone, in the honest-but-curious model. The performance difference between smartphone and laptop client is much higher for the protocol secure against malicious clients, mainly due to memory constraints on the smartphone and high cost of repeated garbage collection.

One-bit Output Instead of Distance. The server's output of our protocols is the distance between the template and the authentication sample. The cost of modifying our protocols, secure against malicious client, to output only one bit as the authentication result is less than 1 second of computation. Amount of communication does not increase significantly, as the additional cost of the *smaller than* subcircuit is compensated by the reduction in the number of output wires to one.

Performance in Malicious Model. The cut-and-choose technique allows transformation of garbled circuits to achieve security in the malicious model (in this model, either the client or the server can be malicious). This technique increases both the time and communication cost roughly κ-fold to achieve κ-bit security.

VIII. Conclusions

We introduced the first efficient privacy-preserving protocols for securely outsourcing authentication using Scaled Euclidean and Scaled Manhattan verifiers, in feature space and PCA space. Our protocols are secure in the hybrid model, where the client is assumed to act maliciously and the server to be HBC. We also designed a light-weight protocol, secure in the HBC model, for 1-Prob verifier.

We performed experiments to demonstrate the accuracy and practicality of our protocols on an emerging area—outsourcing of continuous smartphone user authentication. Our results can be summarized as follows:

- The choice of discretization parameter e is crucial for all our privacy-preserving protocols. For all protocols on Dataset-LTU and for most protocols on Dataset-Frank, $e = 12$ is sufficient and provides similar accuracy to using floating point arithmetics.
- When security against malicious clients or limited communication is required, Scaled Manhattan without PCA gives the best tradeoff between cost and EER.
- If HBC security is sufficient, privacy-preserving Scaled Manhattan and Scaled Euclidean protocols in PCA space offer the best tradeoff between accuracy and performance (with a slight performance edge for Scaled Manhattan).
- Privacy-preserving 1-Prob verifier without PCA in HBC model provides remarkably low overhead. This is, to the best of our knowledge, the fastest protocol designed for secure outsourcing of continuous authentication on smartphones.

References

[1] AdmitOne Security. http://www.admitonesecurity.com.

[2] M. Barbosa, T. Brouard, S. Cauchie, and S. de Sousa. Secure biometric authentication with improved accuracy. In *Proc. of the 13th Australasian conference on Information Security and Privacy*, 2008.

[3] M. Barni, T. Bianchi, D. Catalano, M. Di Raimondo, R. Labati, P. Failla, D. Fiore, R. Lazzeretti, V. Piuri, F. Scotti, and A. Piva. Privacy-preserving fingercode authentication. In *Proc. of the 12th ACM Workshop on Multimedia and Security*, 2010.

[4] Behaviosec. http://www.behaviosec.com.

[5] M. Blanton and P. Gasti. Secure and efficient protocols for iris and fingerprint identification. In *Proc. of the 16th European Conf. on Research in Computer Security*, 2011.

[6] A. Boldyreva, N. Chenette, and A. O'Neill. Order-preserving encryption revisited: Improved security analysis and alternative solutions. In *CRYPTO*, 2011.

[7] J. Bringer, H. Chabanne, M. Izabachene, D. Pointcheval, Q. Tang, and S. Zimmer. An application of the Goldwasser-Micali cryptosystem to biometric authentication. In *Proc. of the 12th Australasian Conference on Information Security and Privacy*, 2007.

[8] H. Chun, Y. Elmehdwi, F. Li, P. Bhattacharya, and W. Jiang. Outsourceable two-party privacy-preserving biometric authentication. In *ASIACCS*, 2014.

[9] R. Cramer, I. Damgård, and J. Nielsen. Multiparty computation from threshold homomorphic encryption. In *Proc. of the Intl. Conf. on the Theory and Application of Cryptographic Techniques*, 2001.

[10] I. Damgård, M. Geisler, and M. Krøigård. A correction to efficient and secure comparison for on-line auctions. Cryptology ePrint Archive, Report 2008/321, 2008.

[11] I. Damgård, M. Geisler, and M. Krøigård. Homomorphic encryption and secure comparison. *Journal of Applied Cryptology*, 1(1), 2008.

[12] I. Damgård, M. Geisler, and M. Krøigård. Asynchronous multiparty computation: Theory and implementation. In *PKC*, 2009.

[13] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft. Privacy-preserving face recognition. In *PETS*, 2009.

[14] M. Frank, R. Biedert, E. Ma, I. Martinovic, and D.Song. Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication. *TIFS*, 8(1), 2013.

[15] C. Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, 2009.

[16] C. Gentry and S. Halevi. Implementing gentry's fully-homomorphic encryption scheme. In *EUROCRYPT*, 2011.

[17] O. Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, 2004.

[18] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC*, 1987.

[19] S. Govindarajan, P. Gasti, and K. Balagani. Secure privacy-preserving protocols for outsourcing continuous authentication of smartphone users with touch data. In *BTAS*, 2013.

[20] V. Goyal, P. Mohassel, and A. Smith. Efficient two party and multi party computation against covert adversaries. In Nigel Smart, editor, *EUROCRYPT*, 2008.

[21] W. Henecka, S. Kogl, A.-R. Sadeghi, T. Schneider, and I. Wehrenberg. TASTY: Tool for Automating Secure Two-partY computations. In *CCS*, 2010.

[22] Y. Huang, D. Evans, J. Katz, and L. Malka. Faster secure two-party computation using garbled circuits. In *USENIX*, 2011.

[23] Y. Huang, J. Katz, and D. Evans. Efficient secure two-party computation using symmetric cut-and-choose. In *CRYPTO*, 2013.

[24] Y. Ishai, J. Kilian, K. Nissim, and E. Petrank. Extending oblivious tranfers efficiently. In *CRYPTO*, 2003.

[25] A. Jain, S. Prabhakar, L. Hong, and S. Pankanti. Filterbank-based fingerprint matching. *IEEE Trans. on Image Processing*, 9(5), 2000.

[26] A. Juels and M. Sudan. A fuzzy vault scheme. In *ISIT*, 2002.

[27] A. Juels and M. Wattenberg. A fuzzy commitment scheme. In *CCS*, 1999.

[28] K. Killourhy and R. Maxion. Comparing anomaly-detection algorithms for keystroke dynamics. In *Proc. of the Annual IEEE/IFIP Intl. Conf. on Dependable Systems and Networks*, 2009.

[29] V. Kolesnikov, A.-R. Sadeghi, and T. Schneider. Improved garbled circuit building blocks and applications to auctions and computing minima. In *CANS*, 2009.

[30] V. Kolesnikov and T. Schneider. Improved garbled circuit: Free XOR gates and applications. In *International Colloquium on Automata, Languages and Programming*, 2008.

[31] G. Kumar, S. Tulyakov, and V. Govindaraju. Combination of symmetric hash functions for secure fingerprint matching. In *Proc. of the 20th Intl. Conf. on Pattern Recognition*, 2010.

[32] Y. Lindell and B. Pinkas. A proof of security of Yao's protocol for two-party computation. *Journal of Cryptology*, 22(2), 2009.

[33] Y. Lindell and B. Pinkas. Secure two-party computation via cut-and-choose oblivious transfer. *Journal of cryptology*, 25(4), 2012.

[34] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella. Fairplay – a secure two-party computation system. In *USENIX*, 2004.

[35] M. Naor and B. Pinkas. Efficient oblivious transfer protocols. In *ACM-SIAM Symposium On Discrete Algorithms (SODA)*, 2001.

[36] M. Osadchy, B. Pinkas, A. Jarrous, and B. Moskovich. SCiFI – A system for secure face identification. In *IEEE Symp. on Security and Privacy*, 2010.

[37] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, 1999.

[38] B. Pinkas, T. Schneider, N. Smart, and S. Williams. Secure two-party computation is practical. In *ASIACRYPT*, 2009.

[39] T. Plantard, W. Susilo, and Z. Zhang. Fully homomorphic encryption using hidden ideal lattice. *TIFS*, 8(12), 2013.

[40] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *STOC*, 1989.

[41] I. Rish. An empirical study of the Naive Bayes classifier. In *IJCAI-01 workshop on Empirical Methods in AI*, 2001.

[42] A.-R. Sadeghi, T. Schneider, and I. Wehrenberg. Efficient privacy-preserving face recognition. In *Intl. Conf. on Information Security and Cryptology*, 2009.

[43] N. Safa, R. Safavi-Naini, and S. Shahandashti. Privacy-preserving implicit authentication. In *ICT Systems Security and Privacy Protection*. 2014.

[44] C. Sanchez-Avilaa and R. Sanchez-Reillob. Two different approaches for iris recognition using Gabor filters and multiscale zero-crossing representation. *Pattern Recognition*, 38(2), 2005.

[45] B. Schoenmakers and P. Tuyls. *Security with Noisy Data: Private Biometrics, Secure Key Storage and Anti-Counterfeiting*, chapter "Computationally secure authentication with noisy data". Springer-Verlag, 2007.

[46] A. Serwadda, V. Phoha, and Z. Wang. Which verifiers work?: A benchmark evaluation of touch based authentication algorithms. In *BTAS*, 2013.

[47] C. Shen, Z. Cai, X. Guan, Y. Du, and R. Maxion. User authentication through mouse dynamics. *TIFS*, 8(1), 2013.

[48] Matthew Turk and Alex Pentland. Eigenfaces for recognition. *Journal of cognitive neuroscience*, 3(1), 1991.

[49] U. Uludag, S. Pankanti, and A. Jain. Fuzzy vault for fingerprints. In *Audio-and Video-Based Biometric Person Authentication*, 2005.

[50] J. Šeděnka, K. Balagani, V. Phoha, and P. Gasti. Privacy-preserving population-enhanced biometric key generation from free-text keystroke dynamics. In *IJCB*, 2014.

[51] A. Yao. How to generate and exchange secrets. In *FOCS*, 1986.

[52] X. Zhao, T. Feng, and W. Shi. Continuous mobile authentication using a novel graphic touch gesture feature. In *BTAS*, 2013.

[53] Supplementary material to this paper. Available on http://ieeexplore.ieee.org/.

APPENDIX

A. HBC Scaled Manhattan and Scaled Euclidean

In Section VII, we compared our new protocols, secure against malicious clients, with the protocols secure in the HBC model from [19]. For reference, we include the Scaled Manhattan and Scaled Euclidean protocol in Figures 6 and 7, respectively. Security proofs and detailed description can be found in [19].

B. Security Analysis of 1-Prob Verifier in HBC

The semantic security of $\mathsf{AES}_c(\cdot)$ prevents the server from extracting any information from $\mathsf{AES}_c(\mathcal{Y})$, besides its size.

The server's view of the protocols consists of the encryption and decryption key for the homomorphic encryption scheme, encrypted matrix $\mathsf{AES}_c(\mathcal{Y})$, and ciphertext $\overline{[\![d]\!]}$ from the client. The server's output is d.

Simulator S_s provides the server with $\mathsf{AES}_c(\mathcal{Y})$ and the decryption key for the homomorphic encryption scheme as input. It then uses d to construct $\overline{[\![d]\!]}$, and sends it to the server. Since $\overline{[\![d]\!]}$ is properly distributed, i.e., is a valid encryption of d, the server cannot distinguish between the simulation and a real execution of the protocol. Therefore, the protocol is secure against a curious server.

The client's view of the protocol consists in the server's public key, the symmetric key for $\mathsf{AES}_c(\cdot)$, x and $\mathsf{AES}_c(\mathcal{Y})$. The client has no output. Simulator S_c selects a random set of values $y_{1,1}, \ldots, y_{n,\ell}$, with $0 \leq y_{i,j} \leq 2^{e-1} - 1$. It then constructs $\mathcal{Y}'$ and sends $\mathsf{AES}_c(\mathcal{Y}')$ to the client. The semantic security of the homomorphic encryption scheme prevents the client from determining that $\mathcal{Y}'$ corresponds to the encryption of random values. Therefore, $\mathsf{AES}_c(\mathcal{Y}')$ is properly distributed. For this reason, the client cannot distinguish between interaction with the S_c and with a honest server. Hence the protocol is secure against a curious client.

Jaroslav Šeděnka received his M.Sc. in Mathematics and B.Sc. in Computer Science from Masaryk University, Brno, Czech Republic, in 2010 and 2007, respectively. He is currently pursuing a Ph.D. degree at the same university. His current research interests include both mathematical and applied cryptography, cryptographic protocols, algebraic number theory and algebra.

Input: Client: sample $x = (x_1, \ldots, x_n)$, server's public key and decryption key for $\mathsf{AES}_c(\cdot)$; Server: encrypted template $\mathsf{AES}_c(y) = \mathsf{AES}_c(\llbracket \alpha_1 y_1 \rrbracket, \ldots, \llbracket \alpha_n y_n \rrbracket)$ (where $\alpha_i = 1/\sigma_i$) and decryption key for the homomorphic encryption scheme.
Output: The server learns $\mathsf{D}_M(x, y)$.
Protocol steps:

1) The server sends $\mathsf{AES}_c(y)$ to the client, which decrypts it, obtaining $\llbracket \alpha_1 y_1 \rrbracket, \ldots, \llbracket \alpha_n y_n \rrbracket$.
2) For $i = 1, \ldots, n$, the client and the server interact in a privacy-preserving comparison protocol. At the end of the protocol the client learns the encryption of bit $b_i = (\alpha_i x_i < \alpha_i y_i)$
3) For $i = 1, \ldots, n$, the client computes: $\llbracket d_i \rrbracket = \llbracket\ |\alpha_i x_i - \alpha_i y_i|\ \rrbracket = \llbracket MAX(\alpha_i x_i, \alpha_i y_i) - MIN(\alpha_i x_i, \alpha_i y_i) \rrbracket = \llbracket (b_i \cdot (\alpha_i y_i - \alpha_i x_i) + \alpha_i x_i) - (b_i \cdot (\alpha_i x_i - \alpha_i y_i) + \alpha_i y_i) \rrbracket$ as:
$$\llbracket d_i \rrbracket = \llbracket b_i \cdot \alpha_i y_i \rrbracket^2 \cdot \llbracket b_i \cdot \alpha_i x_i \rrbracket^{-2} \cdot \llbracket \alpha_i x_i \rrbracket \cdot \llbracket \alpha_i y_i \rrbracket^{-1}$$
The computation of $\llbracket d_i \rrbracket$ requires client and server to perform a short interactive protocol [13] for computing $\llbracket b_i \cdot \alpha_i y_i \rrbracket$.
4) Then, the client computes:
$$\llbracket d \rrbracket = \left\llbracket \left(\sum_{i=1}^{n} d_i \right) \right\rrbracket = \prod_{i=1}^{n} \llbracket d_i \rrbracket$$
5) The client sends $\llbracket d \rrbracket$ to the server, which decrypts it and outputs $\mathsf{D}_M(x, y)$ as d.

Fig. 6: Computation of privacy-preserving Scaled Manhattan distance secure in the HBC model

Input: Client: sample $x = (x_1, \ldots, x_n)$, decryption key for $\mathsf{AES}_c(\cdot)$ and server's public key; Server: encrypted template $\mathsf{AES}_c(y) = \mathsf{AES}_c(\llbracket \alpha_1 y_1^2 \rrbracket, \llbracket \alpha_1 2 y_1 \rrbracket, \ldots, \llbracket \alpha_n y_n^2 \rrbracket, \llbracket \alpha_n 2 y_n \rrbracket)$ (where $\alpha_i = 1/\sigma_i^2$) and decryption key for the homomorphic encryption scheme.
Output: The server learns $\mathsf{D}_E(x, y)$.
Protocol steps:

1) The server sends $\mathsf{AES}_c(y)$ to the client, which decrypts it as $\llbracket \alpha_1 y_1^2 \rrbracket, \llbracket \alpha_1 2 y_1 \rrbracket, \ldots, \llbracket \alpha_n y_n^2 \rrbracket, \llbracket \alpha_n 2 y_n \rrbracket$.
2) For $i = 1, \ldots, n$, the client computes:
$$\llbracket d_i \rrbracket = \llbracket (\alpha_i \cdot (x_i - y_i)^2) \rrbracket = \llbracket (\alpha_i x_i^2) \rrbracket \cdot \llbracket (\alpha_i y_i^2) \rrbracket \cdot \llbracket (\alpha_i 2 y_i) \rrbracket^{-x_i}$$
3) Then, the client computes
$$\llbracket d \rrbracket = \left\llbracket \left(\sum_{i=1}^{n} d_i \right) \right\rrbracket = \prod_{i=1}^{n} \llbracket d_i \rrbracket$$
4) The client sends $\llbracket d \rrbracket$ to the authentication server, which computes d and outputs $\mathsf{D}_E(x, y)$ as $\sqrt{d}$.

Fig. 7: Computation of privacy-preserving Scaled Euclidean distance secure in the HBC model

Sathya Govindarajan received his B.Tech. degree in electronics and communication engineering from the Jawaharlal Nehru Technological University, Hyderabad, India, in 2009, and M.S. degree in computer science from the New York Institute of Technology, Old Westbury, New York, U.S.A., in 2013. His research interests include data analytics, digital image and video processing, human computer interaction, and new advancements in computer graphics that include simulation, rendering, and scientific visualization.

Dr. Paolo Gasti is an assistant professor of Computer Science at the New York Institute of Technology (NYIT), School of Engineering and Computing Sciences. Dr. Gasti's research focuses on behavioral biometrics, privacy-preserving biometric authentication and identification, secure multi-party protocols and network security. Before joining NYIT, he worked as a research scholar at University of California, Irvine. His research has been sponsored by the Defense Advanced Research Project Agency and the U.S. Air Force. He received his B.S., M.S., and Ph.D. degrees from University of Genoa, Italy. He is a Fulbright scholar, and member of IEEE.

Dr. Kiran Balagani is an assistant professor of computer science at the New York Institute of Technology. His research interests are in cyber-behavioral anomaly detection (e.g., unauthorized user-access behaviors), behavioral biometrics, and privacy-preserving biometrics. Balaganis work has appeared in several peer-reviewed journals, including the IEEE Transactions on Pattern Analysis and Machine Intelligence, the IEEE Transactions on Information Forensics and Security, the IEEE Transactions on Knowledge and Data Engineering, the IEEE Transactions on Systems, Man, and Cybernetics, and Pattern Recognition Letters. He holds three U.S. patents in network-centric attack detection. His teaching interests include development of graduate and undergraduate courses in network security and biometrics. Balagani received the Ph.D. degree and two M.S. degrees from Louisiana Tech University; and the B.S. degree from Bangalore University, India.