Journal of Computer Security -1 (2018) 1–14
DOI 10.3233/JCS-171017
IOS Press

The impact of application context on privacy and performance of keystroke authentication systems

Kiran S. Balagani [a], Paolo Gasti [a,*], Aaron Elliott [b], Azriel Richardson [c] and Mike O'Neal [c]
[a] *Department of Computer Science, New York Institute of Technology, NY, USA*
E-mails: kbalagan@nyit.edu, pgasti@nyit.edu
[b] *Aegis Research Lab, LLC, LA, USA*
E-mail: aaron@aegisresearchlabs.com
[c] *Department of Computer Science, Louisiana Tech University, LA, USA*
E-mails: azriel_richardson@yahoo.com, mike@coes.latech.edu

Abstract. In this paper, we show that keystroke latencies used in continuous user authentication systems disclose application context, i.e., in which application user is entering text. Using keystroke data collected from 62 subjects, we show that an adversary can infer application context from keystroke latencies with 95.15% accuracy.

To prevent leakage from keystroke latencies, and prevent exposure of application context, we develop privacy-preserving authentication protocols in the outsourced authentication model. Our protocols implement two popular matching algorithms designed for keystroke authentication, called Absolute ("A") and Relative ("R"). With our protocols, the client reveals no information to the server during authentication, besides the authentication result. Our experiments show that these protocols are fast in practice: with 100 keystroke features, authentication was completed in about one second with the "A" protocol, and in 595 ms with the "R" protocol. Further, because the asymptotic cost of our protocols is linear, they can scale to a large number of features. On the other hand, by leveraging application context we were able to reduce HTER from 14.7% with application-agnostic templates, to as low as 5.8% with application-specific templates.

Keywords: Biometrics, keystroke authentication, privacy, privacy-preserving protocols, application context

1. Introduction

Commercial providers of keystroke-based behavioral authentication services, e.g., Behaviosec [11], often operate under the outsourced authentication model. In this model, the user's device collects aggregated keystroke data (e.g., average inter-keystroke latencies computed from hundreds of keystrokes typed within an authentication window), and sends it to a remote server for authentication. It is therefore natural to ask what the authentication server can learn from such aggregated latencies.

Multiple studies have shown that aggregated keystroke latencies can be used to infer private information such as demographic and affective information [5, 10, 16]. One aspect that has received less attention is the leakage of *application context* through aggregated keystroke latencies. In this paper, we aim to address this gap. We demonstrate that access to aggregated latencies enables the authentication

*Corresponding author. E-mail: pgasti@nyit.edu.

server to infer which application is being used by the user. We address this newly identified privacy leakage by designing two novel cryptographic protocols that allow the implementation of secure outsourced authentication without disclosing *any information* to the authentication server, other than the result of the authentication process.

While the primary contribution of our paper is to address the leakage of information from aggregated keystroke latencies, we also show that application context can be leveraged to improve authentication performance in scenarios where authentication is done locally on the device and does not involve sending latencies to third parties.

The key contributions of this paper can be summarized as follows.

(1) We demonstrate that aggregated keystroke data, such as average key press latencies, can be used to predict application context with high (95.15%) accuracy. This result was achieved *without using individual keystrokes typed by the user*, and is quite worrisome because an authentication server with access to aggregated latencies can exploit them to learn about user's routine, activities, and application usage. For instance, an adversarial server could determine, for each day of the week, how much time the user is spending on word processing, browsing, or working on spreadsheets. This represents an unnecessary violation of the user's privacy, particularly because application usage information should not have been exposed to a third party if the goal was to authenticate users using keystrokes.

(2) To address privacy concerns arising from the exposure of keystroke data, we develop novel cryptographic privacy-preserving protocols that prevent information leakage under the outsourced authentication model [1]. Specifically, we introduce two protocols which calculate Absolute ("A") and Relative ("R") measures [20] (defined in Section 5.2), and compare them with protocols that compute scaled Euclidean and scaled Manhattan distance from Govindarajan et al. [19]. Performance evaluation on a standard desktop computer shows that our protocols are practical, because their overhead is between 35 ms and 4.5 s.

(3) We report the improvement in authentication performance that can be achieved using application-specific template matching. In our experiments, application-agnostic matching led to an half total error rate (HTER) of 14.7%, while we were able to achieve significantly better HTER-s with application-specific matching (5.8% for Microsoft Word, 5.9% with Outlook, and 13.8% with Internet Explorer). Further, authentication with application-specific templates significantly outperformed authentication with mismatched templates (i.e., when probe keystrokes from application X were matched against a template from a different application Y).

Organization. The rest of this paper is organized as follows. In Section 2 we review the related work. The dataset, matching algorithms, and fusion techniques used in our experiments are summarized in Section 3. Section 4 reports our results on leakage of application context from keystroke data. We introduce new privacy-preserving protocols for "A" and "R" metrics in Section 5. Authentication accuracy using application-specific templates is reported in Section 6. We conclude in Section 7.

2. Related work

Next, we briefly review the state of the art on information leakage from keystroke timing information, on privacy-preserving protocols for continuous authentication, and on the impact of context on behavioral authentication. For a general survey on keystroke-based authentication, we refer the reader to Banerjee et al. [2].

2.1. Information leakage from keystroke timing information

Song et al. [44] demonstrated a side-channel attack in which latencies between consecutive key press events were used to reconstruct what the user was typing. The authors used a Hidden Markov Model (HMM) to predict character sequences from keystroke latencies collected during SSH sessions, and validated their technique by successfully predicting 8-character passwords for 4 users.

Zhang and Wang [48] demonstrated that inter-key times for all local users can be obtained from the `procfs` filesystem exposed by Linux, and that this information can be used to reconstruct character sequences being typed. The authors demonstrated that the characters inferred from inter-key latencies reduced password search space by 50 to 2000 times.

There is growing evidence that keystroke latencies can be used to predict demographic information (e.g., gender, native vs. non-native English speakers), and affective information (e.g., mood, stress level, engagement, and cognitive load). For instance, Fairhurst et al. [16] achieved less than 5% error rates in predicting gender from press-press, release-release, press-release, and release-press times. In [5], Bixler demonstrated that keystroke timing features, combined with task appraisal (e.g., interest in a topic *before* performing a writing task on it) and information on the user's stable traits (e.g., scholastic aptitude), successfully predicted affective states such as boredom and engagement during writing tasks.

In [10], Brizan et al. observed that keystroke timing information can be used to predict gender and other demographic information. Their results show that the transition from a punctuation symbol to the spacebar was faster for male subjects than for female subjects, while the times before and after common digraphs, such as '`ou`' and '`er`' were shorter for female subjects than for male subjects. The authors also demonstrated that keystroke latencies can be used to distinguish native from non-native English speakers using the timing information "before and after a period", "before and after a common digraph", and "before and after function keys".

The research presented in our paper extends the state of the art on keystroke authentication by showing that application context can be reliably inferred from keystroke timing information.

2.2. Privacy-preserving protocols for continuous authentication

There is extensive literature on the use of privacy-preserving protocols in the context of biometric authentication and identification. Bringer et al. [9] were the first to introduce a general security model for biometric user authentication. Their model assumes low trust between the involved parties, and formalizes privacy for biometric authentication. Furthermore, [9] introduces a privacy-preserving protocol that computes the Hamming distance of two bit strings representing a biometric sample and a template respectively. Barbosa et al. [3] extended the framework of Bringer et al. [9] with a classifier to improve authentication accuracy and propose an instantiation based on Support Vector Machine (SVM) using homomorphic encryption.

Schoenmakers and Tuyls [38] introduced a protocol for secure privacy-preserving iris matching. Their 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 the encrypted domain, with a threshold.

Erkin et al. [15] introduced the first privacy-preserving face recognition protocol. Sadeghi et al. [36] subsequently improved the performance of the protocol of Erkin et al. [15]. More recently, Osadchy et al. [33] designed a new face recognition algorithm together with its privacy-preserving realization called SCiFI. SCiFI simultaneously improves robustness and efficiency of [15, 36].

Blanton and Gasti [6] 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. [4] presented a privacy-preserving protocol for fingerprint identification using FingerCodes [23]. Their technique is not as discriminative as techniques based on location of minutiae points, but is particularly well suited for efficient privacy-preserving implementations.

Govindarajan et al. [19] introduced two protocols for continuous smartphone user authentication based on touchscreen features. Their protocols compute Scaled Manhattan and Scaled Euclidean verifiers, and are secure in the semi-honest model. Subsequently, Šeděnka et al. [40] improved on the work of Govindarajan et al. [19] by introducing new scaled Manhattan and scaled Euclidean protocols secure in the malicious model.

Safa et al. [37] presented a protocol for outsourcing continuous authentication with a Scaled Manhattan verifier on smartphones, and considered security against malicious clients. However, the security argument presented in their protocol does not take into account information disclosed by order-preserving encryption. (See, e.g., the analysis of Boldyreva et al. [7].) As such, there is a substantial amount of information leaked by the client during the protocol, even in the ciphertext-only scenario. In contrast, our schemes *provably* leak no information.

Techniques based on fuzzy commitments [25, 26, 39, 46] 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) [29]. 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.

While there is prior work on privacy-preserving biometric authentication, this paper is the first to introduce secure protocols for "A" and "R" metrics. These metrics have been shown to perform well in multiple keystroke authentication studies (see, for example, [21, 35, 47]).

2.3. Use of context on behavioral authentication

Several papers have leveraged different types of context to enhance the performance of keystroke authentication systems on desktops. Most popular among these contexts are (1) application context; (2) linguistic context (e.g., keystroke features collected from specific words); (3) cognitive context (e.g., keystroke features collected from typing bursts); and (4) motion context (i.e., keystroke features collected while walking or while sitting).

Application context. Dowland et al. [14] conjectured that the use of application-specific templates might impact authentication performance. By analyzing typing data from one user, the authors suggested that Messenger and Microsoft Word benefit from application-specific templates more than Internet Explorer and Microsoft PowerPoint.

In contrast to [14], our work uses data from 62 users, and therefore takes into account both false accept and false reject errors. Further, our work establishes that the use of application context improves authentication accuracy for most applications by comparing the use of application-specific templates with application-agnostic and mismatched templates.

Linguistic context. Sim and Janakiraman [42] demonstrated that the discriminability of digraph latencies increased when digraphs were associated with the word they came from. Their experiments, performed on a dataset collected from 22 users, showed that the inter-user Manahalobis distance of word-specific digraphs was substantially lower than that of digraphs without word context.

Goodkind et al. [18] showed that using word context decreased authentication errors. Specifically, when adding word context to digraphs, EER was reduced by about 1%.

Cognitive context. Locklear et al. [32] performed a study on typing data collected from 486 users. In their experiments, keystroke features outperformed cognitive features for continuous authentication. However fusion of the two classes of features led to the lowest authentication EER.

Body motion context. Sitova et al. [43] showed that body motion affects the performance of keystroke-based authentication on smartphones. Using data from 99 subjects, the authors demonstrated that a combination of signals from keystroke timings, gyroscope, and accelerometer sensors leads to higher authentication accuracy when users were walking, compared to sitting.

3. Dataset and authentication algorithms

Dataset. The dataset used in this work was collected from 62 subjects, who at the time of data collection were cadets of the United States Military Academy at West Point, New York.[1] Each user ran the data collection application on her/his personal computer for a period of two weeks. The application collected data 24 hours a day, 7 days a week. The entire data collection process was performed within a 6-week period. The dataset contains keystroke dynamics data collected during the following activities: (1) web browsing using Microsoft Internet Explorer; (2) word processing using Microsoft Word; (3) email composition with Microsoft Outlook; and (4) performing scientific calculations with Wolfram Mathematica.

Feature categories. We extracted six different *categories* of features from our dataset. Each category is a collection of individual keystroke features. The categories are: (1) KHL: contains key hold latency (down-up time) of each key as features; (2) IK: contains key interval latency (up-down time) of each pair of consecutive keys; (3) KRL: contains key release latency (up-up time) of each pair of consecutive keys; (4) KeyHoldWithNextVKCode: contains key hold with next key context preserved (e.g., key hold latency when typing letter '`a`' followed by typing letter '`m`' is treated differently from the latency of '`a`' followed by letter '`n`'); and (5) KeyHoldWithPrevVKCode: contains key hold with previous key context preserved (e.g., key hold latency while typing letter '`h`' when it is preceded by letter '`t`' is treated differently from the latency of '`h`' when preceded by letter '`g`'); and (6) KH2: contains key hold with word context (e.g., key hold of '`a`' in the word '`ball`' is different from key hold of '`a`' in '`lab`').

Prediction and matching algorithms. For predicting application context from aggregated keystroke information, we used Random Forest [8], Naïve Bayes [31], and Tree-augmented Naïve Bayes [49] classifiers.

For biometric verification, we used five different matching algorithms that have been used previously in keystroke authentication studies. They are: (1) Scaled Manhattan [28], (2) Scaled Euclidean [28], (3) "A" [20], (4) Similarity [24]; and (5) "R" [20].

Fusion. Decisions from all verifier-feature pairs were combined, or fused together, to make an overall authentication decision. We used a weighted decision fusion method to fuse the individual authentication results [22, 30]. We implemented weighing of classifier decisions using the Simultaneous Perturbation Stochastic Approximation (SPSA) algorithm [45].

[1] An IRB approval was obtained prior to performing data collection.

4. Prediction of application context from aggregated keystroke latencies

While it is straightforward to infer application context from individual keystrokes, it might appear that the same information cannot be inferred from aggregated keystroke timings (e.g., average inter-key latencies computed from hundreds of keystrokes typed within an authentication window). In this section, we show that aggregated keystroke latencies can be used to predict application context with high accuracy. We formulate this problem as a classification problem, where the underlying applications form the class labels, and aggregated keystroke data collected from each application was used to train the classification model. The goal of this task was to correctly identify the application, given a vector of keystroke statistics.

For training, we used all keystroke data from 31 users, while for testing we used all keystrokes from a separate group of 31 users who were not included in training. For each user, we divided the keystrokes from each application into two sets. Each set was used to construct one vector, as long as the set contained at least 8,800 keystrokes. This allowed us to build 206 training vectors from the first group of 31 users, and 206 testing vectors from the remaining users.

We computed the following three statistics from the individual keystroke features: *standard deviation*, *mean*, and *normalized count* (the number of occurrences of a feature, divided by the number of occurrences of all features). Using the following example, we illustrate how these statistics were computed. Let '`aaa bbb ccc`' be the text typed by a subject. For the KHL category, we computed the average, standard deviation, and normalized count of keyhold timings of all `a`'s, then of all `b`'s, and finally of all `c`'s. In a similar fashion, we computed the statistics for all the remaining feature categories. We used the statistics from all categories as input to the classifier.

We used four classes: Microsoft Word, Microsoft Internet Explorer, Microsoft Outlook, and Wolfram Mathematica. Our results are presented in Tables 1 and 2.

Table 1

Context identification accuracies, listed by feature categories. Darker cells corresponds to higher accuracy. We use RF to indicate Random Forests; NB for Naïve Bayesian; and TAN to indicate Tree-Augmented Naïve Bayesian

	RF Accuracy	**NB Accuracy**	**TAN Accuracy**	**RF Accuracy**	**NB Accuracy**	**TAN Accuracy**	**RF Accuracy**	**NB Accuracy**	**TAN Accuracy**
	100 Most Available Features			**50 Most Available Features**			**10 Most Available Features**		
All	94.66%	87.86%	85.92%	91.75%	88.35%	84.47%	80.58%	79.13%	78.16%
All but IK	94.66%	86.41%	88.35%	92.23%	87.86%	84.47%	79.13%	80.58%	75.24%
All but KeyHoldWithNextVKCode	94.66%	85.44%	87.38%	91.75%	88.35%	85.92%	81.07%	79.61%	77.67%
All but KeyHoldWithPrevVKCode	94.66%	86.41%	85.92%	92.72%	87.38%	84.95%	80.1%	78.16%	78.16%
All but KH2	94.66%	87.86%	84.95%	91.75%	88.35%	87.38%	80.1%	78.64%	78.16%
All but KPL	94.17%	85.92%	86.89%	93.2%	88.83%	84.95%	80.1%	81.07%	75.73%
All but KRL	93.69%	86.41%	87.38%	91.75%	88.35%	84.95%	79.13%	81.07%	77.18%
All but KH	93.2%	88.35%	85.92%	89.32%	83.5%	84.95%	76.21%	78.16%	76.21%

In Table 1 we report our results for the 100, 50, and 10 most available features for each feature category. "All" means that the classifier was trained using all statistics computed on all feature categories. "All but IK" means the features in the IK category were excluded from the experiments. Classification accuracy was calculated as number of *correctly classified sessions* divided by the *total number of sessions*. Our results with Random Forest (the best performer among the tested classifiers) show that excluding a single feature category does not significantly affect performance (see Table 1). In fact, by removing each individual feature category, the drop in accuracy is always below 3% with the 100 and 50 most available features, and below 5% with the 10 most available features.

Table 2

	RF Accuracy	NB Accuracy	TAN Accuracy	RF Accuracy	NB Accuracy	TAN Accuracy	RF Accuracy	NB Accuracy	TAN Accuracy
	100 Most Available Features			50 Most Available Features			10 Most Available Features		
STDDEV, NORMALIZEDCOUNT	95.15%	88.83%	86.89%	91.75%	88.35%	84.95%	79.61%	81.55%	78.16%
MEAN, STDDEV, NORMALIZEDCOUNT	94.66%	87.86%	85.92%	91.75%	88.35%	84.47%	80.58%	79.13%	78.16%
MEAN, NORMALIZEDCOUNT	94.17%	88.35%	87.38%	92.72%	89.32%	87.38%	80.58%	79.61%	77.67%
NORMALIZEDCOUNT	94.17%	89.81%	87.38%	92.23%	89.32%	87.86%	80.1%	81.55%	77.67%
MEAN, STDDEV	58.25%	62.14%	57.28%	61.17%	57.28%	53.88%	47.57%	49.51%	46.6%

In Table 2, we report our results for the 100, 50, and 10 most available features for each statistic. For example, the *normalized count* row in Table 2 shows the application prediction accuracy achieved using the normalized counts of the 100, 50, and 10 most available features in each feature category.

We were able to achieve over 95% classification accuracy with the 100 most available features, using *standard deviation* and *normalized count* statistics with Random Forests. *Normalized count* is the statistic that contributes the most to the classification accuracy. As shown in the last row of Table 2 (MEAN, STDDEV), when this statistic was excluded, classification accuracy dropped to 62.1% or less.

Sub-contexts within an application. The results presented in this section were achieved without isolating overlapping sub-contexts within each application. For instance, it is possible that users composed emails and performed word processing tasks using Internet Explorer. The former task would overlap with Microsoft Outlook, while the second with Microsoft Word. Despite such possibility of overlap, we were able to reliably distinguish between application context.

5. Privacy-preserving protocols for keystroke authentication

To address application context leakage under the outsourced keystroke authentication model, we introduce new privacy-preserving protocols for "A" and "R" matching algorithms. Our protocols allow a client to authenticate its user by exchanging encrypted information with a server. The server stores an encrypted copy of the user's template, and learns the outcome of the authentication process. Our protocols provably reveal no information about the keystroke data to the server, which is therefore unable to infer application context or any other ancillary information. Moreover, our protocols reveal no information on the template to the client and the server. Before we introduce our protocols, we briefly review cryptographic preliminaries.

5.1. Cryptographic preliminaries

Security model. We use the term *adversary* to refer to protocol participants (i.e., the client and the authentication server). Outside adversaries can be mitigated via standard network security techniques (e.g., TLS), and are therefore not considered in our analysis.

Our protocols are secure against semi-honest (or honest-but-curious) adversaries. Semi-honest participants are assumed to follow prescribed protocol behavior. However, they might try to learn additional information beyond the protocol output by analyzing messages exchanged during protocol execution. 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 semi-honest adversaries if, for $i \in \{1,2\}$, there exists a probabilistic polynomial time simulator S_i such that:

$$\{S_i(\mathsf{in}_i, f_i(\mathsf{in}_1, \mathsf{in}_2))\} \equiv \{\text{VIEW}_\pi(P_i), \mathsf{out}_i\}$$

Homomorphic encryption and comparison. 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 $\mathsf{Enc}(m)^a = \mathsf{Enc}(a \cdot m)$.

Our constructions can be instantiated using any semantically secure additively homomorphic encryption scheme (e.g., [12, 13, 34]). We instantiate our protocols using the homomorphic construction of Damgård et al. [12, 13] (henceforth, DGK), because it is faster [6] (and produces smaller ciphertexts) than the well known Paillier cryptosystem [34]. 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 has access to the corresponding decryption key.)

Our privacy-preserving "A" and "R" protocols rely on the comparison protocol of Erkin et al. [15]. This interactive protocol allows the client to compare two values encrypted using a homomorphic scheme, without having access to the decryption key, and without learning anything besides the output of the protocol.

Symmetric encryption. Our protocols use a semantically secure symmetric encryption scheme for protecting the template. In our implementation, we use AES in counter (CTR) mode. We use $\mathsf{E}_c(m)$ to indicate symmetric encryption performed under a key known by Client.

5.2. Protocols description

Our protocols are divided in two phases: enrollment, and verification. Enrollment is executed once per user (or when the user wants to updates her biometric profile), while verification is performed periodically (e.g., every 60–120 seconds).

In the enrollment phase, the client constructs the user's template by extracting features from keystroke data. The template is encrypted with homomorphic encryption under the server's public key, and then with symmetric encryption using a key known only to the client (e.g., a user-provided password). The resulting ciphertext is stored on the server, which cannot reconstruct the unencrypted template because it has no access to the symmetric encryption key.

The client constructs a user template $y = \{y_1, \ldots, y_n\}$. For our "A" metric protocol, the *encrypted* template is then computed as $\mathsf{E}_c(y) = \mathsf{E}_c([\![y_1]\!], \ldots, [\![y_n]\!])$, and $\mathsf{E}_c(1/y) = \mathsf{E}_c([\![1/y_1]\!], \ldots, [\![1/y_n]\!])$. For our "R" metric protocol, the encrypted template is computed as $\mathsf{E}_c(y) = \mathsf{E}_c(\langle [\![\text{rank}_{\mathtt{aa}}]\!], \mathtt{aa}\rangle, \ldots, \langle [\![\text{rank}_{\mathtt{zz}}]\!], \mathtt{zz}\rangle)$, where $\text{rank}_i \in \{1, \ldots, n\}$, and $\text{rank}_i \neq \text{rank}_j$ for all $i \neq j$.

Next, we present the verification phase for each of our privacy-preserving protocols. Our protocols for computing "A" and "R" measures are illustrated in Algorithms 1 and 2, respectively.

Privacy-preserving "A" measure. Gunetti et al. [20] define the similarity between two n-graphs occurring in two typing samples as $\max(d_1, d_2)/\min(d_1, d_2)$, where d_i is the timing associated with each n-graph. Then, the "A" distance is defined as the number of similar n-graphs divided by the total number of n-graphs. We implement this measure by using two runs of the comparison protocol of Erkin et

Algorithm 1 Our privacy-preserving "A" measure protocol

Input: Client: sample $x = (x_1, \ldots, x_n)$, Server's public key for $[\![\cdot]\!]$ and decryption key for $\mathsf{E}_c(\cdot)$; Server: encrypted template $\mathsf{E}_c(y) = \mathsf{E}_c([\![y_1]\!], \ldots, [\![y_n]\!])$, $\mathsf{E}_c(1/y) = \mathsf{E}_c([\![1/y_1]\!], \ldots, [\![1/y_n]\!])$, and decryption key for the homomorphic encryption scheme $[\![\cdot]\!]$.
Output: Server learns the "A" similarity measure between x and y.
Protocol steps:

(1) Server sends $\mathsf{E}_c(y)$ and $\mathsf{E}_c(1/y)$ to Client which decrypts it, obtaining $[\![y_1]\!], \ldots, [\![y_n]\!]$ and $[\![1/y_1]\!], \ldots, [\![1/y_n]\!]$.
(2) For $i = 1, \ldots, n$, Client and Server engage in privacy-preserving comparison of $[\![x_i]\!]$ and $[\![y_i]\!]$. Let the result of each comparison be $[\![b_i]\!]$, where $b_i \triangleq x_i > y_i$.
(3) For $i = 1, \ldots, n$, Client and Server compute $[\![d_i]\!] = [\![\max(x_i, y_i)/\min(x_i, y_i)]\!]$ as $[\![(b_i \cdot (x_i - y_i) + y_i) \cdot (b_i \cdot (1/x_i - 1/y_i) + 1/y_i)]\!]$. This requires Client and Server to jointly compute three products.
(4) Then, Client and Server engage in the comparison protocol to compute the encryption of $b'_i \triangleq d_i < t$
(5) Client computes the "A" distance as:

$$[\![d]\!] = \left[\!\!\left[\left(\sum_{i=1}^{n} b'_i \right) \right]\!\!\right] = \prod_{i=1}^{n} [\![b_i]\!]$$

(6) Client sends $[\![d]\!]$ to Server, which decrypts it and outputs d/n.

al. [15] (one to compute min and max, and the other to compare the similarity of two n-graphs with the threshold). The privacy-preserving protocol implementing "A" distance is detailed in Algorithm 1.

Privacy-preserving "R" measure. In order to implement "R" distance, we reduce it to Manhattan distance as follows. The user's template is composed of a set of digraphs, together with their rank (stored in encrypted form). During authentication, the user's sample x determines the ranking of features for the session. Client ranks the elements from the encrypted template according to x. If x and the encrypted template are ranked identically, then Client obtains the encrypted list $\bar{y} = 1, \ldots, n$. If not, some of the elements of $\bar{y}$ will not be in the correct order. Therefore, the Manhattan distance between $\bar{y}$ and $1, \ldots, n$, normalized by n, is the "R" distance. The privacy-preserving protocol implementing "R" distance is shown in Algorithm 2.

5.3. Performance analysis

The computational complexity of "R" protocols is $\mathcal{O}(n)$ for both the client and the server, where n is the number of features. Analogously, the complexity of our "A" protocol is $\mathcal{O}(n)$ for both Client and Server. For reference, the complexity of the scaled Euclidean protocol from [19] is $\mathcal{O}(n)$ for the client, and $\mathcal{O}(1)$ for the server, while the complexity of the scaled Manhattan protocol is $\mathcal{O}(n)$ for both the client and the server.

We performed experiments on a Linux system with two Intel Xeon E5420 CPUs running at 2.5 GHz. Our prototype implementation is written in C, and relies on the GNU GMP library. Table 3, summarizes the performance results of our protocol implementation with 25–400 features. These results are obtained representing each feature using 10 bits. The cost of the "R" protocol is dominated by the computation

Algorithm 2 Our privacy-preserving "R" measure protocol. Without loss of generality, the protocol is shown with the use of digraphs. It is easy to see how it can be easily extended to use arbitrary n-graphs

Input: Client: sample $x = (x_1, \ldots, x_n)$, Server's public key for $[\![\cdot]\!]$, and decryption key for $\mathsf{E}_c(\cdot)$; Server: encrypted template $\mathsf{E}_c(y) = \mathsf{E}_c(\langle[\![\mathrm{rank}_{\mathtt{aa}}]\!], \mathtt{aa}\rangle, \ldots, \langle[\![\mathrm{rank}_{\mathtt{zz}}]\!], \mathtt{zz}\rangle)$ (where $\mathrm{rank}_i \in \{1, \ldots, n\}$, and $\mathrm{rank}_i \neq \mathrm{rank}_j$ for all $i \neq j$), and decryption key for the homomorphic encryption scheme $[\![\cdot]\!]$.
Output: Server learns the "R" similarity measure between x and y.
Protocol steps:

(1) Server sends $\mathsf{E}_c(y)$ to Client, which decrypts it, obtaining $\langle[\![\mathrm{rank}_{\mathtt{aa}}]\!], \mathtt{aa}\rangle, \ldots, \langle[\![\mathrm{rank}_{\mathtt{zz}}]\!], \mathtt{zz}\rangle$.
(2) Client ranks (sorts) $[\![\mathrm{rank}_{\mathtt{aa}}]\!], \ldots, [\![\mathrm{rank}_{\mathtt{zz}}]\!]$ according to x. Let $[\![r_1]\!], \ldots, [\![r_n]\!]$ indicate the result of this process.
(3) Client and Server engage in the privacy-preserving Manhattan protocol, where $[\![y]\!] = [\![r_1]\!], \ldots, [\![r_n]\!]$, and $w = (1, \ldots, n)$. At the end of the protocol, Server learns Manhattan distance d between w and $[\![r_1]\!], \ldots, [\![r_n]\!]$.
(4) Server, outputs d as the "R" distance.

Table 3
Performance of our prototype implementation. Because the performance difference between scaled Manhattan and "R" is negligible, we report the two protocols together

	Scaled Manhattan [19]/R		Scaled Euclidean [19]		A Measure	
# of Feat.	Server	Client	Server	Client	Server	Client
25	149 ms	35 ms	$\approx$ 1 ms	22 ms	276 ms	37 ms
50	298 ms	71 ms	$\approx$ 1 ms	44 ms	551 ms	74 ms
100	595 ms	142 ms	$\approx$ 1 ms	88 ms	1102 ms	147 ms
200	1190 ms	284 ms	$\approx$ 1 ms	178 ms	2205 ms	294 ms
400	2380 ms	568 ms	$\approx$ 1 ms	355 ms	4409 ms	589 ms

of the privacy-preserving scaled Manhattan distance. Therefore, we report the performance of the two protocols combined.

Our client and server implementations are single core. This allows straightforward performance comparison with the related work. However, all protocols presented in this paper can greatly benefit from multiple cores, because most operations can be performed in parallel on different features.

5.4. Security analysis

The security of our protocols relies on the security of the underlying building blocks. In our analysis, we assume that $[\![\cdot]\!]$ is a semantically secure homomorphic encryption scheme, and that $\mathsf{E}_c(\cdot)$ is a semantically secure symmetric encryption scheme. In our instantiation, $[\![\cdot]\!]$ was implemented using the DGK encryption scheme, which was shown to be semantically secure [12, 13]. To implement $\mathsf{E}_c(\cdot)$, we used AES-CTR, which is known to be semantically secure under standard assumptions [27]. The privacy-preserving comparison protocol of Erkin et al. was shown to be secure in [15], and therefore it is not included it in our analysis. Finally, the privacy-preserving scaled Manhattan protocol used to construct our "R" protocol was proved secure in [19].

To show that our protocols are secure, next we describe how to simulate the view of each party using its inputs and outputs alone. Because these simulations are indistinguishable from the real execution of

the protocol, for semi-honest parties this implies that the protocols do not reveal any information besides the protocol output to the client and server.

Privacy-preserving "R" verifier. The security of the privacy-preserving protocol implementing the "R" verifier reduces to the security of the privacy-preserving Manhattan distance protocol used in Step 3 of Algorithm 2. In fact, because of the semantic security of $\mathsf{E}_c(\cdot)$, Server does not learn information about the *biometric* template (including which n-graphs have been include in the template) from the *encrypted* template. Further, because of the semantic security of $[\![\cdot]\!]$, Client does not learn information about n-graph ranking in the template. Because Step 3 of Algorithm 2 does not disclose any information to Client and Server besides their inputs and outputs, the protocol in Algorithm 2 is secure against honest-but-curious Client and Server.

Privacy-preserving "A" verifier. Since $\mathsf{E}_c(\cdot)$ is semantically secure, the server cannot extract any information from $[\![y]\!]$. The server's view of the protocols consists of the decryption key for $[\![\cdot]\!]$, encrypted vector $[\![y]\!]$, and ciphertext $[\![d]\!]$ from the client. (The server's view also contains the message exchanged during the comparison and multiplication protocols. We ignore these messages since the two protocols have been proven secure in [15].) The server's output is d/n. Simulator S_s provides the server with $[\![y]\!]$ and with the decryption key for $[\![\cdot]\!]$ as input. It then uses the protocol output d/n to build $[\![d]\!]$, which is sent to the server. Since $[\![d]\!]$ is properly distributed, the server cannot distinguish between the simulation and a real execution of the protocol. Therefore, the protocol is secure against an honest-but-curious server.

The client's view of the protocol consists in the server's public key, the symmetric key for $\mathsf{E}_c(\cdot)$, sample x, and encrypted template $[\![y]\!]$ and $[\![1/y]\!]$. The client has no output. Simulator S_c selects a random set of values $y'_1,\ldots,y'_n$, constructs $[\![y']\!] = \mathsf{E}_c([\![y'_1]\!],\ldots,[\![y'_n]\!])$, and $[\![1/y']\!] = \mathsf{E}_c([\![1/y'_1]\!],\ldots,[\![1/y'_n]\!])$, and sends them to the client. The semantic security of $[\![\cdot]\!]$ prevents the client from determining that $[\![y']\!]$ and $[\![1/y']\!]$ correspond to the encryption of random values. Therefore, $[\![y']\!]$ and $[\![1/y']\!]$ are 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 honest-but-curious client.

6. Authentication accuracy using application-specific templates

When authentication is performed locally on the user's device, disclosure of application context to the authentication system does not represent a privacy violation. Assuming this use case, we show that it is possible to use application-specific templates to improve authentication accuracy. Our results provide empirical evidence that leveraging application context improves authentication accuracies. We constructed application-specific templates by using all keystroke data from one application at a time. For instance, we constructed a Microsoft Word-specific template by computing the average of each feature value collected while the users were typing text in Word.

Table 4 shows the impact of application-specific matching in keystroke-based verification. We report accuracy results, expressed in terms of HTER, for templates constructed using 1,100 keystrokes. HTER is a standard accuracy measure defined as (FAR+FRR)/2, where FAR represents the percentage of vectors from an impostor incorrectly classified as vectors from the legitimate user (false accept rate), and FRR the percentage of vectors from the legitimate user incorrectly classified as impostor vectors (false reject rate) [41]. The diagonal (highlighted in bold) indicates the HTER-s obtained for application-specific matching (i.e., test vectors collected within an application are matched with the templates from the same application). The non-diagonal cells represent HTER-s obtained when the test vectors from an

Table 4

Performance of keystroke verification using fusion of A, R, SE, and SM verifiers on key hold, key press, digraph, key hold with next, key hold with previous. Templates are constructed using 1,100 keystrokes

		Testing (Session 2)			
		Word	Internet Explorer	Outlook	Mathematica
Training (Session 1)	Word	$\mathbf{0.0582\ (\sigma = 0.0442)}$	$0.2766\ (\sigma = 0.1177)$	$0.0810\ (\sigma = 0.0707)$	$0.3247\ (\sigma = 0.1451)$
	Internet Explorer	$0.1167\ (\sigma = 0.0753)$	$\mathbf{0.1375\ (\sigma = 0.0390)}$	$0.1190\ (\sigma = 0.0932)$	$0.3015\ (\sigma = 0.1306)$
	Outlook	$0.0726\ (\sigma = 0.0584)$	$0.2645\ (\sigma = 0.1053)$	$\mathbf{0.0589\ (\sigma = 0.0523)}$	$0.3499\ (\sigma = 0.1615)$
	Mathematica	$0.2434\ (\sigma = 0.1532)$	$0.3014\ (\sigma = 0.1306)$	$0.2639\ (\sigma = 0.1581)$	$\mathbf{0.1770\ (\sigma = 0.1104)}$

Table 5

Number of users with at least 1,100 keystrokes for each combination of training and testing

		Testing (Session 2)			
		Word	Internet Explorer	Outlook	Mathematica
Training (Session 1)	Word	**62**	51	58	35
	Internet Explorer	51	**51**	49	32
	Outlook	56	47	**56**	34
	Mathematica	34	31	34	**34**

application are matched against a template from a different application (cross-template matching). Our results show that the HTER-s obtained for application-specific matching are considerably lower than HTER-s obtained with cross-template matching.

On the other hand, the HTER of application-agnostic matching, i.e., the template and the test vectors were constructed using keystrokes from all applications, was 14.7%. This is therefore less accurate than application-specific matching – with the exception of Mathematica. However, further experiments show that the HTER increase associated with Mathematica can be attributed to the lower number of keystroke available in this application: as we increase the number of keystrokes used in the construction of application-specific user templates from 1,100 to 1,400, we observed a reduction in HTER to 10.6%, although with data from 26 users instead of 34. (Availability information for 1,100 keystrokes is presented in Table 5.)

7. Conclusion

In this paper, we showed that application context impacts both privacy and performance of keystroke authentication. We were able to measure application-specific variations from aggregated keystroke data, and successfully used them to identify application context with over 95% accuracy. Because the identification of application context raises significant privacy concerns when authentication is outsourced to a third-party server, we designed two new privacy-preserving protocols for continuous user authentication with keystroke data. Our protocols incur relatively small overhead (595 ms for "R" computation of distance, and 1.1 s for "A" distance, using 100 keystroke features), and disclose no information besides the outcome of the authentication process to the server.

We then determined whether application context can be used to improve authentication accuracy. Using application-specific templates we successfully reduced HTER from 14.7% (with application-agnostic matching) down to 5.8% for Microsoft Word, 5.9% for Microsoft Outlook, and 13.8% for Microsoft Internet Explorer. However, because of the low availability of keystroke samples, the HTER of Mathematica was higher than the baseline at 17.7%.

Acknowledgments

This work was supported in part by DARPA Active Authentication grant FA 8750-13-2-0274. 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.

The authors thank Colonel Ron Dodge (United States Military Academy, West Point, NY) and his team for leading the data collection effort.

References

[1] AdmitOne Security Sentry, http://www.admitonesecurity.com.
[2] S.P. Banerjee and D.L. Woodard, Biometric authentication and identification using keystroke dynamics: A survey, *Journal of Pattern Recognition Research* **7**(1) (2012), 116–139. doi:10.13176/11.427.
[3] M. Barbosa, T. Brouard, S. Cauchie and S. de Sousa, Secure biometric authentication with improved accuracy, in: *Proceedings of the 13th Australasian Conference on Information Security and Privacy*, 2008.
[4] 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: *Proceedings of the 12th ACM Workshop on Multimedia and Security*, 2010.
[5] R. Bixler and S. D'Mello, Detecting boredom and engagement during writing with keystroke analysis, task appraisals, and stable traits, in: *Proceedings of the 2013 International Conference on Intelligent User Interfaces*, ACM, 2013, pp. 225–234.
[6] M. Blanton and P. Gasti, Secure and efficient protocols for iris and fingerprint identification, in: *Proceedings of the 16th European Conference on Research in Computer Security (ESORICS)*, 2011.
[7] A. Boldyreva, N. Chenette and A. O'Neill, Order-preserving encryption revisited: Improved security analysis and alternative solutions, in: *Advances in Cryptology – CRYPTO*, 2011.
[8] L. Breiman, Random forests, *Machine learning* **45**(1) (2001), 5–32. doi:10.1023/A:1010933404324.
[9] J. Bringer, H. Chabanne, M. Izabachene, D. Pointcheval, Q. Tang and S. Zimmer, An application of the Goldwasser–Micali cryptosystem to biometric authentication, in: *Proceedings of the 12th Australasian Conference on Information Security and Privacy*, 2007.
[10] D.G. Brizan, A. Goodkind, P. Koch, K. Balagani, V.V. Phoha and A. Rosenberg, Utilizing linguistically enhanced keystroke dynamics to predict typist cognition and demographics, *International Journal of Human-Computer Studies* **82** (2015), 57–68. doi:10.1016/j.ijhcs.2015.04.005.
[11] Continuous Authentication – Behaviosec, http://www.behaviosec.com.
[12] I. Damgård, M. Geisler and M. Krøigård, Homomorphic encryption and secure comparison, *Journal of Applied Cryptology* **1**(1) (2008).
[13] I. Damgård, M. Geisler and M. Krøigård, A correction to efficient and secure comparison for on-line auctions, 2008.
[14] P. Dowland, S. Furnell and M. Papadaki, Keystroke analysis as a method of advanced user authentication and response, in: *Security in the Information Society*, Springer, 2002, pp. 215–226.
[15] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk and T. Toft, Privacy-preserving face recognition, in: *Privacy Enhancing Technology Symposium (PETS)*, 2009.
[16] M. Fairhurst, D. Costa-Abreu et al., Using keystroke dynamics for gender identification in social network environment, in: *Imaging for Crime Detection and Prevention 2011 (ICDP 2011), 4th International Conference on*, IET, 2011, pp. 1–6.
[17] O. Goldreich, *Foundations of Cryptography: Volume 2, Basic Applications*, Cambridge University Press, 2004. doi:10.1017/CBO9780511721656.
[18] A. Goodkind, D.G. Brizan and A. Rosenberg, Improvements to keystroke-based authentication by adding linguistic context, in: *Biometrics Theory, Applications and Systems (BTAS), 2015 Seventh IEEE International Conference on*, IEEE, 2015, pp. 1–6.
[19] S. Govindarajan, P. Gasti and K. Balagani, Secure privacy-preserving protocols for outsourcing continuous authentication of smartphone users with touch data, in: *Biometrics Theory, Applications and Systems (BTAS), 2013 IEEE International Conference on*, 2013.
[20] D. Gunetti and C. Picardi, Keystroke analysis of free text, *ACM Transactions on Information and System Security (TISSEC)* **8**(3) (2005), 312–347. doi:10.1145/1085126.1085129.
[21] J. Hu, D. Gingrich and A. Sentosa, A k-nearest neighbor approach for user authentication through biometric keystroke dynamics, in: *IEEE International Conference on Communications (ICC)*, IEEE, 2008, pp. 1556–1560.

[22] A. Jain, K. Nandakumar and A. Ross, Score normalization in multimodal biometric systems, *Pattern recognition* **38**(12) (2005), 2270–2285. doi:10.1016/j.patcog.2005.01.012.
[23] A. Jain, S. Prabhakar, L. Hong and S. Pankanti, Filterbank-based fingerprint matching, *IEEE Transactions on Image Processing* **9**(5) (2000).
[24] S.S. Joshi, *Naive Bayes and Similarity Based Methods for Identifying Computer Users Using Keystroke Patterns*, Dissertation, Louisiana Technical University, 2009.
[25] A. Juels and M. Sudan, A fuzzy vault scheme, in: *International Symposium on Information Theory (ISIT)*, 2002.
[26] A. Juels and M. Wattenberg, A fuzzy commitment scheme, in: *ACM Conference on Computer and Communications Security (CCS)*, 1999.
[27] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*, Chapman & Hall/CRC, 2008.
[28] K. Killourhy and R. Maxion, Comparing anomaly-detection algorithms for keystroke dynamics, in: *Proceedings of the Annual IEEE/IFIP Intl. Conference on Dependable Systems and Networks*, 2009.
[29] G. Kumar, S. Tulyakov and V. Govindaraju, Combination of symmetric hash functions for secure fingerprint matching, in: *Proceedings of the 20th Intl. Conference on Pattern Recognition*, 2010.
[30] L.I. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*, John Wiley & Sons, 2004. doi:10.1002/0471660264.
[31] D.D. Lewis, Naive (Bayes) at forty: The independence assumption in information retrieval, in: *European Conference on Machine Learning (ECML)*, Springer, 1998, pp. 4–15.
[32] H. Locklear, S. Govindarajan, Z. Sitova, A. Goodkind, D.G. Brizan, A. Rosenberg, V.V. Phoha, P. Gasti and K.S. Balagani, Continuous authentication with cognition-centric text production and revision features, in: *Biometrics (IJCB), 2014 IEEE International Joint Conference on*, IEEE, 2014, pp. 1–8.
[33] M. Osadchy, B. Pinkas, A. Jarrous and B. Moskovich, SCiFI – A system for secure face identification, in: *IEEE Symp. on Security and Privacy*, 2010.
[34] P. Paillier, Public-key cryptosystems based on composite degree residuosity classes, in: *Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, 1999.
[35] K.A. Rahman, K.S. Balagani and V.V. Phoha, Snoop-forge-replay attacks on continuous verification with keystrokes, *Information Forensics and Security, IEEE Transactions on* **8**(3) (2013), 528–541. doi:10.1109/TIFS.2013.2244091.
[36] A.-R. Sadeghi, T. Schneider and I. Wehrenberg, Efficient privacy-preserving face recognition, in: *Intl. Conference on Information Security and Cryptology*, 2009.
[37] N. Safa, R. Safavi-Naini and S. Shahandashti, Privacy-preserving implicit authentication, in: *ICT Systems Security and Privacy Protection*, 2014.
[38] B. Schoenmakers and P. Tuyls, Security with noisy data: Private biometrics, secure key storage and anti-counterfeiting, Springer-Verlag, 2007, Chap. "Computationally secure authentication with noisy data".
[39] J. Šeděnka, K. Balagani, V. Phoha and P. Gasti, Privacy-preserving population-enhanced biometric key generation from free-text keystroke dynamics, in: *Biometrics (IJCB), 2014 IEEE International Joint Conference on*, 2014.
[40] J. Sedenka, S. Govindarajan, P. Gasti and K.S. Balagani, Secure outsourced biometric authentication with performance evaluation on smartphones, *Information Forensics and Security, IEEE Transactions on* **10**(2) (2015), 384–396. doi:10.1109/TIFS.2014.2375571.
[41] C. Shen, Z. Cai, X. Guan, Y. Du and R.A. Maxion, User authentication through mouse dynamics, *IEEE Transactions on Information Forensics and Security* **8**(1) (2013), 16–30. doi:10.1109/TIFS.2012.2223677.
[42] T. Sim and R. Janakiraman, Are digraphs good for free-text keystroke dynamics?, in: *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, IEEE, 2007, pp. 1–6.
[43] Z. Sitova, J. Sedenka, Q. Yang, G. Peng, G. Zhou, P. Gasti and K. Balagani, HMOG: A new biometric modality for continuous authentication of smartphone users, in: *IEEE Transactions on Information Forensics & Security*, 2015.
[44] D.X. Song, D. Wagner and X. Tian, Timing analysis of keystrokes and timing attacks on SSH, in: *USENIX Security Symposium*, Vol. 2001, 2001.
[45] J.C. Spall, Simultaneous perturbation stochastic approximation, *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control* (2003), 176–207.
[46] U. Uludag, S. Pankanti and A. Jain, Fuzzy vault for fingerprints, in: *Audio-and Video-Based Biometric Person Authentication*, 2005.
[47] N. Yager and T. Dunstone, The biometric menagerie, *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **32**(2) (2010), 220–230. doi:10.1109/TPAMI.2008.291.
[48] K. Zhang and X. Wang, Peeping Tom in the neighborhood: Keystroke eavesdropping on multi-user systems, *USENIX Security Symposium* **20** (2009), 23.
[49] F. Zheng and G.I. Webb, Tree augmented naive Bayes, in: *Encyclopedia of Machine Learning*, Springer, 2011, pp. 990–991.