Inf Syst Front (2015) 17:1007–1015
DOI 10.1007/s10796-015-9587-9

Secure and privacy preserving data processing support for active authentication

Yan Sun[1] · Shambhu Upadhyaya[1]

Published online: 29 July 2015

Abstract Keystroke dynamics and mouse movements are effective behavioral biometric modalities for active authentication. However, very little is done on the privacy of collection and transmission of keyboard and mouse data. In this paper, we develop a rule based data sanitization scheme to detect and remove personally identifiable and other sensitive information from the collected data set. Preliminary experiments show that our scheme incurs on average 5.69 % false negative error rate and 0.64 % false positive error rate. We also develop a data transmission scheme using the Extensible Messaging and Presence Protocol (XMPP) to guarantee privacy during transmission. Using these two schemes as a basis, we develop two distinct architectures for providing secure and privacy preserving data processing support for active authentication. These architectures provide flexibility of use depending upon the application environment.

Keywords Privacy · Security · Sanitization · XMPP

1 Introduction

The standard methods to authenticate a computer/network user, which typically occur once at the initial log-in, suffer from a variety of vulnerabilities such as masquerading and potential system compromise, which can allow the attacker to steal important personal and organizational information as well as the identity of the victim. An effective solution to this one-time authentication problem is the active or continuous authentication using behavioral biometrics. Researchers have taken various approaches for the collection and use of behavioral biometrics. Keystroke dynamics and mouse movement are useful mechanisms for continuous authentication since a user at a console is typically typing or moving the mouse continuously throughout the session. The keystroke patterns are based on the timing between successive keystrokes, and the amount of time a key is depressed. Similarly, the mouse movement patterns are based on mouse movement coordinates, mouse clicks, etc. There is a large body of research on user authentication and masquerade detection in a live session by monitoring keystrokes dynamics (Monrose and Rubin 1997; Shavlik et al. 2001; Bergadano et al. 2002; Gunetti and Picardi 2005; Gupta et al. 2008), mouse movements (Goecks and Shavlik 1999; Pusara and Brodley 2004), and their combination (Ahmed and Traore 2005).

However the privacy of collection and transmission of keyboard and mouse data did not receive much attention among these works, nonetheless, it is important. When keystroke data is collected, usually through keystroke loggers, the personally identifiable information and personally sensitive information such as user name, password, social security number, etc., will be recorded together. Although the active authentication systems and tools do not use the sensitive information directly for authentication, such data collection may compromise the privacy of users during the process of active authentication. Especially, when the active authentication is deployed in a production system, there are potential risks that malicious third parties could gain illegal access to user's privacy information. Moreover, the active authentication may be carried out on the server side. In such a case, the data when sent over to a server would expose the sensitive information and an

✉ Shambhu Upadhyaya
shambhu@buffalo.edu

Yan Sun
ysun27@buffalo.edu

[1] Department of Computer Science and Engineering, State University of New York at Buffalo, Buffalo, NY 14260, USA

adversary (especially if the data is sent through a wireless channel) can intercept the communication and learn the passwords and other sensitive information. The current approaches to deal with this problem are largely ad hoc and no systematic study of this problem is available.

The objective of this paper is to conduct a systematic study of the security and privacy of the keystroke dynamics and the mouse movements approach to active authentication. We propose two approaches based on where the data anonymization is done, whether on the client side or on the server side. If the data anonymization issue is handled on the client side, then the data can be sent to the server directly without any privacy concerns. If the processing is done on the server side, the data transmission security needs to be guaranteed. We present with two architectures that use primarily commercial off-the-shelf (COTS) products and schemes to implement our ideas. We develop a script tool to remove the personally identifiable information and sensitive information. The Extensible Messaging and Presence Protocol (XMPP) (Johansson 2005) is used to guarantee security when the data needs to be transmitted to the server. The experiment conducted on real data shows the performance of the various schemes.

1.1 Summary of contributions

The main contribution of the paper is the development of two new architectures for providing privacy preserving data processing support for active authentication and its adoption as a secure authentication scheme in the real world. Specifically, we develop:

- A lightweight rule based scripting tool for removing personally identifiable information and other sensitive information from the keystroke dynamics data in behavioral biometrics based authentication systems.
- An XMPP based scheme for safely transmitting the keystroke dynamics data from the client to the server.

The above two schemes are evaluated with real test data.

1.2 Paper organization

The remainder of the paper is organized as follows. We describe the data acquisition and anonymization, the XMPP protocol and give some preliminaries in Section 2. Section 3 introduces our new rule based data sanitization scheme. Section 4 presents two systematic schemes for privacy preserving data processing and discusses their merits and drawbacks. In Section 5 we perform the experimental evaluation and discuss the results. Section 6 discusses the merits and drawbacks of our anonymization scheme. We conclude the paper and provide future directions of research in Section 7.

2 Preliminaries

2.1 Data acquisition

An active system logger from our previous work (Garg et al. 2006) is used to collect real user behavior data. The system logger has been developed using the Microsoft.NET framework and C# language on the Microsoft Windows platform. The.NET framework has been chosen due to its ease of use and the ability to seamlessly interact with various Windows components. The logger has been designed such that it is able to collect system events and all possible user activities on the system in real-time. The logger collects events such as keyboard activity, mouse movement coordinates, mouse clicks, system background processes and user run commands. Table 1 shows a sample of the data collected by the logger. As can be seen from the table, the logger collects the timestamp information along with the events such as mouse coordinates, mouse clicks, process information and keyboard statistics.

Among the data collected by the active logger, only the KeyPress activity may contain user's privacy information. So the following discussions mainly focus on the keystroke records.

2.2 Data anonymization

Before analyzing or processing the keystroke dynamics, the sensitive and privacy information contained within the data set needs to be removed, hidden or properly handled. Usually, there are two approaches to data anonymization: data masking and data sanitization.

Data masking (Radhakrishnan et al. 2005) or data obfuscation is the process of hiding original data with random characters or data. The main reason for applying masking to a data field is to protect data that is classified as personally identifiable data, personally sensitive data or commercially sensitive data, however the data must remain usable for the purposes of undertaking valid test cycles. It must also remain real and be consistent. Substitution, number and data variance and shuffling are effective methods of applying data masking and being able to preserve the authentic look and feel of the data

Table 1 A sample dataset collected by the logger

CPU Ticks	Event	Value
634564465190625000	Mouse coordinates	464,348
634564465190625000	Left Click	
634564462834375000	New process	chrome.exe
634564462895937500	KeyPress	G
634564462922187500	Process terminated	chrome.exe
634560110556562500	KeyPress	Back

records. Various approaches have been taken in this area (Radhakrishnan et al. 2005; Ravikumar et al. 2011; Ahmed and Athreya 2013). However, it may be necessary to retain the semantics of data in order to extract some biometric features during the analysis of keystroke dynamics. So data masking may not be suitable in the context of keystroke dynamics based active authentication.

Data sanitization is the process of removing sensitive information from the data set, so that it may be distributed to a broader group. When dealing with classified information, sanitization attempts to reduce the document's classification level, possibly yielding an unclassified document. Through data sanitization, personally identifiable information would be deleted directly from the keystroke dynamics data. The processed data can then be transmitted or simply analyzed under no risk of loss of privacy.

Numerous products, both open source and commercial, are available to assist in detecting and removing sensitive data in digital media. Some examples are: *Identity Finder*, *Find_SSN*, *Sensitive Number Finder* (*SENF*), *Spider*, *and MITRE Identification Scrubber Toolkit* (*MIST*). However, these products do not meet the requirements of data sanitization in our context. These products only perform full text searches for personally identifiable information in Word, Excel, PDF and other documents. But keystroke dynamics and mouse movements data set is generated in the form of single letter, punctuation, function keys and mouse coordinates and particularly, they will not work on the data set we got from the active logger. Furthermore, the detection algorithms in the above-mentioned products mainly rely on the data semantics. But sensitive information in keystroke dynamics data is not always included in paragraphs or forms. It is quite often context-independent. Therefore, we develop our own rule-based data sanitization scheme, which will be discussed in the next section.

2.3 XMPP

Extensible Messaging and Presence Protocol (XMPP) is a communications protocol for message-oriented middleware based on XML (Extensible Markup Language) (Johansson 2005). The protocol was originally named Jabber (Jabber INC 1998), and was developed by the Jabber open-source community in 1999 for near real-time, instant messaging (IM), presence information, and contact list maintenance.

XMPP provides great convenience for transmission of keystroke dynamics and mouse movement data from client side to server side. XMPP is becoming a standard for Machine-to-Machine (M2M) messaging and communication in both wired and wireless systems. It provides near real-time performance. Strong security (via *SASL* and *TLS*) has been built into the core XMPP specifications. Using XMPP, the security of keystroke dynamics data can be protected since all data will be encrypted during transmission. Moreover, unlike most instant messaging protocols, XMPP is defined in an open standard and uses an open systems approach of development and application, by which anyone may implement an XMPP service and interoperate with other organizations' implementations. Because XMPP is an open protocol, implementations can be developed using any software license; although many server, client, and library implementations are distributed as free and open-source software, numerous freeware and commercial software implementations also exist.

3 Rule based data sanitization scheme

The data sanitization scheme we developed is a simple lightweight rule-based system that can be used on devices with limited processing power. We have chosen shell script due to its ease of use and the ability to interact with various Linux components in a seamless fashion. The algorithms can also be easily implemented using other languages such as python, C, C# or Java.

The data sanitization process mainly consists of two steps: data reconstruction and sensitive information detection. For data reconstruction, we extract the keystroke data, including the keystroke letters and functions such as space, enter and backspace, from the user behavior data stream. Before the reconstruction, the backspace keystroke needs to be handled. The letter appearing before the backspace should be deleted from the data stream, because these letters are users' typographical errors. They will lead to misdetection for the next step if kept in the data stream. Then we reconstruct the data set by word segmentation based on punctuation and function keys. The next step is the detection of sensitive information in the data stream. Figure 1 shows the pseudo code of the algorithm. In this figure, S is an array that stores the original keystrokes data, R is an array that holds the segmentation results and P is an array that keeps the detected sensitive data.

For the purpose of this paper, we define sensitive information as email address, username, password, SSN, and birthday. Depending upon the nature of the typed data, the definition of sensitive information can be extended. The sensitive information detection is based on a comprehensive set of rules. The rules can be classified into two categories, structure-based and content-based. The structure-based rules capture the sensitive information appearing in a particular context within the data set. From the study we found that these sensitive information usually appear after some keywords such as mail, username, etc. For instance, after we type the email website address (mail.xxx.com) into a usually followed browser, we will enter the account name and password (it is similar with social networking sites). The sensitive information in this case, the account name and password, follows directly the keyword "mail." So we can just delete one or two words

Fig. 1 Algorithm for data sanitization

```
Algorithm
while keystroke logger is running do
    while the number of stored keystroke <= given window size do
        read current keystroke;
        store keystroke information into 2-D array S;
    end while
    for each keystroke k in array S do
        if k ==  "Return" or k == "tab" or k == "space" or k == Punctuation then
            increase array cell index by 1 of array R;
        else if k =="backspace" then
            delete one keystroke from current cell of array R;
        else
            write keystroke letter into current cell of array R;
        end if
    end for
    for each word in array R do
        apply all search rules; // see the set of rules in Table 2
        store detected sensitive information ID into array P;
    end for
    for each keystroke in array S do
        if current keystroke is within array P then
        delete current keystroke from array S;
        else
            send current keystroke information to server;
        end if
    end for
end while
```

following the keyword. This will also take care of the partial words typed if the auto-complete mode is enabled in the email applications. The content-based rules are designed to use the same methods as existing identity detection products to capture the sensitive information. For example, if a word containing the "@" symbol is detected, it will be considered as an email address or an account name and deleted. The sensitive information detection is based on a set of rules such as the above. Other similar rules are added to our script. Table 2 shows the details of all the rules and detection mechanisms used in our work. These rules appear to be simple but enough rules must be added so that the scheme has a high coverage and accuracy of sanitization. It may be noted that it is not necessary to precisely detect every single letter of sensitive data; actually deleting parts of words or a subset of letters will make the information hard to use by a malicious third party. However, if an attacker gains access to the sanitized data with supplementary knowledge about all the rules, there exists a small risk of the attacker reconstructing the sensitive data from the remaining data. So when the sanitization schemes are deployed in real systems, it is beneficial to keep the rules private.

In terms of time complexity, our rule-based algorithm stays at a very effective level. The complexity of the data reconstruction step is $O(n)$, where n is the number of keystrokes. The complexity of the detection step is $O(m)$, where m is the number of letters, $m \leq n$. So the time complexity of the algorithm is:

$$O(n) + O(m) = O(n) \tag{1}$$

4 Data processing support architectures

Based on the various methods of data collection and transmission of keystroke and mouse movement data in the literature, we propose two architectures that use mostly commercial off-the-shelf (COTS) products and schemes to provide the security and privacy support to the data.

4.1 Data anonymization handled on the client side

In this architecture, the keystroke dynamics and mouse movement records logged by the active system logger are fed into our data sanitization scheme presented in Section 3 directly on the client side. The contained personally identifiable and sensitive data are detected and removed as they are generated. The sensitive data does not leave the machine in which the keyboard data is collected. The pre-processed data can be sent to remote server in clear view through any transmission protocol without any privacy leak concerns, as it does not contain

Table 2 Detection rules

Sensitive components	Rule trigger mechanism
Social security number	Nine consecutive digits Nine digits following the format of XXX-XX-XXXX Appear following the key word “SSN”
Phone number	Ten or seven consecutive digits Ten digits following the format of (XXX) XXX-XXXX Seven digits following the format of XXX-XXXX Appear following the key word “number”
Birthday date	Six consecutive digits Six digits following the format of XX-XX-XXXX or XX/XX/XXXX or XX.XX.XXXX Appear following the key word “DOB” or the words that contain the key word “birth”
Bank account, debit card and credit card number	Sixteen consecutive digits Sixteen digits following the format of XXXX-XXXX-XXXX-XXXX Appear following the key word “number”
Email account	Contain the “@” symbol Appear following the word that contain key word “mail”
Password	Appear following email account Appear following the key word “password”
User name	Appear in the email format Appear following the key word “user”

any sensitive information. Then the behavioral biometrics analysis can be carried out on the remote server.

4.2 Data anonymization handled on the server side

In this architecture, the keystroke dynamics and mouse movement records logged by the active system logger will be encrypted and streamed to the remote server using XMPP. Since the collected date is encrypted and safely transferred through the XMPP system, there is no need to be concerned about sensitive information leak. Once the collected data arrives at the server side, it will be fed into our data sanitization scheme for further processing or archival. The biometrics analysis can be done on the “sanitized data.”

For the purpose of data transmission in this architecture, we developed an XMPP transmission scheme based on Openfire, a real-time collaboration (RTC) server licensed under the Open Source Apache License, and Smack API, an Open Source XMPP client library for instant messaging and presence. Two operating modes are implemented, the chat mode and the file mode, for different requirements. In the chat mode, the transmission scheme checks the keystroke dynamics and mouse movement log file generated from the active system logger every five seconds and sends the new changes to the remote server. In the file mode, the transmission starts at the end of the section. The whole log file is being sent to the remote server.

4.3 Comparison

In the previous two subsections, we described two data processing architectures, viz. anonymization on the client side and anonymization on the server side. Table 3 shows the merits and drawbacks of these two architectures. As can be seen from the table, both architectures have their own advantages and disadvantages. For anonymization on the client side, more computation and storage overhead are incurred for the client. For anonymization on the server side, there are more security concerns during transmission, despite encryption.

5 Experimental evaluations and results

5.1 Data sannitization scheme evaluation

In this section, we elaborate our experiments on the data sanitization scheme. The data sanitization experiment has been done on the server side, however, it can be effortlessly migrated to the client side. We collected real user keystroke dynamics and mouse movement data from 15 different volunteers using the active system logger (Garg et al. 2006). The 15 participants, ranging in keyboard proficiency, are all college students. The collection session for each participant lasts for around 45 min. The data was generated by engaging volunteer subjects. Each user was asked to complete the following tasks: checking the emails online, visiting random websites, filling some forms, opening and closing several applications and typing a random passage which contains some personal information. These tasks are set based on a normal daily work place profile. Auto-fill and usage of drop down list with pre-typed input are allowed in the tasks, however, it is not a concern because these features do not lead to any keystroke data.

The collected data is fed to the data sanitization scheme. Then the processed data is compared with the original data to see the differences and hence the accuracy of sanitization. Based on the ground truth on the original log files, the error rates of this sanitization scheme are calculated.

There are two different kinds of misdetections in our experiment. The first kind of misdetection is that the sensitive information contained in the original log files is not removed by the sanitization scheme (false negative error). The second kind is that the information deleted by our sanitization scheme is not actually sensitive information within the original log files (false positive error). The former should be considered

Table 3 Merits and limitations of the two architectures

	Merits	Limitations
Anonymization on client side	Safer, the sensitive info does not leave the machine	Extra workload for client
Anonymization on server side	More accurate but higher complexity sanitization scheme can be used	Additional computation due to encryption/ decryption Potentially lower transmission throughput Potential leaking risk during transmission

as real misdetection since it brings potential risk of privacy information leak. However, the latter just leads to mistakenly deleting some non-sensitive information from the collected data. This will not affect the behavioral biometrics analysis result because it only deletes a negligibly small percentage of the whole data set and hence its impact could be ignored.

Table 4 shows the experimental results. As can be seen from the table, the data sanitization scheme achieves an average false negative error rate of 5.69 % and an average false positive error rate of 0.64 % only with $O(n)$ time complexity. Considering only a limited number of detection rules are added to our sanitization scheme, there is a potential possibility that the accuracy could be improved by adding more rules.

5.2 Impact of data sanitization on behavioral biometrics analysis

To evaluate the impact of data sanitation on behavioral biometrics analysis, we have replicated an existing well-known study (Leggett et al. 1991). The classification and authentication algorithms in Leggett's study are applied to both original data and sanitized data in our experiment.

Table 4 Detection accuracy rate

	Size of log file (M)	False negative error rate (%)	False positive error rate (%)
User1	6.6	12.5	0.67
User2	2.4	0	0.25
User3	5.4	0	0.22
User4	2.7	9.09	0.75
User5	4.4	20	0.35
User6	4.0	0	1.29
User7	6.6	7.14	1.42
User8	4.4	6.67	1.12
User9	4.6	11.1	0.69
User10	2.5	11.8	0.99
User11	4.6	0	0.63
User12	8.2	0	0.32
User13	6.0	0	0
User14	2.7	0	0
User15	4.6	7.14	0.86
Average		5.69	0.64

The study follows the process described below. We use the same data sets discussed in pervious Section 5.1. Part of the subject's data is used to create a profile of the user. The remaining data is used as a test sample to test against the reference profile. A similarity score is calculated between a test sample and a reference profile. In the case of identity classification, the test sample is classified as coming from the owner of the reference profile that yields the highest similarity score. In case of identity authentication, a threshold is chosen such that if the similarity score of a test sample is higher than the chosen threshold, the test sample is considered originating from the owner of the reference profile, and thus accepted. To measure performance, we use error rate (the number of misclassified subjects divided by the total number of subjects) for classification and False Acceptance Rate (FAR) and False Rejection Rate (FRR) for authentication.

The di-graph latency feature is extracted for the data set and used by the algorithm. Di-graph latency is defined as the time interval between two adjacent keys in a subject's typing; for example, the word "the" consists of two di-graphs, "th" and "he." The concept can be generalized to an n-graph for n adjacent keys. The n-graph latency represents the time delay between the key down event of the first key and the key up event of the nth key.

Leggett et al. (1991) assume that the latency times for all occurrences of a di-graph in the reference profile follow a normal distribution (μ, σ). If the latency time for a di-graph in the test sample falls between the acceptance region ($[\mu - d * \sigma, \mu + d * \sigma]$), it is considered accepted; otherwise, rejected. The similarity score for the test sample is in turn defined as the ratio of the number of accepted di-graphs over the total number of di-graphs appearing in a test sample.

We implemented this algorithm in MATLAB and evaluated it using both original data and sanitized data for classification and authentication scenarios. In the experiment, we simply equally divided each subject's data into two halves, one half as reference profile and the other half as test sample. We set $d = 1$ and the acceptance threshold to 0.8. The performance results are listed in Table 5.

Theoretically, the sanitization process will not affect the behavioral biometrics analysis result because it only deletes

Table 5 Performace of classification and authentication

	Classification	Authentication	
	Error rate	FAR	FRR
Original data	6.7 % (1 out of 15)	3.81 %	8.09 %
Sanitized data	6.7 % (1 out of 15)	3.81 %	8.09 %

a negligibly small percentage of the whole data set. The experiment results, as it is shown in Table 5, fully support this judgment.

5.3 XMPP scheme evaluation

The keystroke dynamics data can be securely transmitted to the remote server using our XMPP scheme. All the messages are encrypted through the Transport Layer Security (TLS) protocol where strong security has been built into the core XMPP specifications. In order to present a clear view of the transmission process, in this section we describe a simplified version of our XMPP scheme.

Our XMPP scheme is developed based on Openfire server and Smack API. The Openfire server is a well-designed application. We only need to install it and set up some basic configurations. The client is programmed using the smack API. The simplified transmission process can be divided into two steps: creating the connection to server and sending the message.

Figure 2 shows the code for creating a connection to the server. Through the default setting of XMPP, the maximum possible security is negotiated with the server, including the TLS encryption, but the connection falls back to lower security settings, if necessary. However, in our context every message containing the keystroke dynamics needs to be encrypted. So, we reconfigure the connection so that security via TLS encryption is mandated in order to make the connection. If the server does not offer TLS or if the TLS negotiation fails, the connection to the server will fail. This way, the message security is ensured.

Figure 3 shows the code for sending the message. By using the smack API, the keystroke dynamics data can be easily sent to a remote server through the encrypted XMPP connection. However, as we discussed in Section 4, some overhead will be introduced due to the additional computation for message encryption/decryption.

```
// Create the configuration for this new
connection
ConnectionConfiguration config = new
ConnectionConfiguration(server, 5222);
config.setSecurityMode(ConnectionConfigurati
on.SecurityMode.required);
config.setCompressionEnabled(true);
config.setSASLAuthenticationEnabled(true);
Connection connection = new
XMPPConnection(config);
// Connect to the server
connection.connect();
// Log into the server
connection.login("username", "password",
"SomeResource");
```

Fig. 2 Creating a connection to server

6 Discussion

The main objective of this paper has been to provide a secure way of collecting and transferring biometrics authentication data without leaking any privacy and sensitive information. As observed before, considerable research has been done on keystroke authentication, however, not many of them considered the privacy issue and secure data transmission which are important for the practical deployment of keystroke dynamics based behavioral biometrics authentication schemes.

The personally identifiable information and personally sensitive information discussed in this paper mainly refer to the plaintext based information, such as user name, password, social security number, etc. The sensitive information that stays within the context, linguistic features such as user's writing style, will not be handled by the sanitization tool in this study. This type of privacy data is hard to be captured with a simple sanitization tool. Furthermore, in some scenarios, the hidden information may serve as core features for classification or authentication, thus they should be preserved.

A question that may arise is about the necessity of sanitization if the keystroke and mouse data are going to be encrypted during transmission. In both architectures we presented, sanitization is handled either on the client side or on the server side. When the data is sent to the server in plaintext form, obviously, privacy information should be removed before sending. Even if the data is encrypted, the sanitization

```
// Assume we have created a XMPPConnection name "connection".
String remoteServer = "serverName";
ChatManager chatmanager = connection.getChatManager();
Chat newChat =
connection.getChatManager().createChat(remoteServer, new
MessageListener() {
    public void processMessage(Chat chat, Message message)
{
        System.out.println("Received message: " + message);
    }
});
try {
    newChat.sendMessage("The Keystroke Dynamics Data");
}
catch (XMPPException e) {
    System.out.println("Error Delivering block");
}
```

Fig. 3 Sending message

process is still necessary on the server side. This is because of the possible exposure of sensitive data otherwise, during decryption. Clearly, the data need to be decrypted before being fed into the underlying machine learning algorithm(s) for the purpose of training and testing. Moreover, the processed data may be stored for a specific period of time on the server. Thus, there is a potential for privacy leak without sanitization. Moreover, when the system is deployed in a real system, the authentication process may be carried out by a third part server. Therefore, sanitization becomes necessary.

Another security concern is if the sanitized data gets intercepted during transmission in clear text. In certain scenarios, a malicious third party may apply the learning algorithms to the intercepted data and obtain the user biometric feature profile and use it to do replay attacks or synthetic forgery attacks to the system. It may bring the potential risks for the proposed architecture. According to the study in (Pusara and Brodley 2004), it is possible to construct a successful replay attack by reverse-engineering the keystroke data but it is much harder to construct a useful replay attack with mouse data. Therefore, for security consideration, the keystroke dynamic authentication may be combined with mouse movement authentication when the system is deployed in the real world.

7 Conclusion

In this paper, we addressed the data privacy issues arising during data collection and transmission for active authentication. We developed a data sanitization scheme to detect and remove personally identifiable and sensitive information from the collected data set. It incurs an average 5.69 % false negative error rate and 0.64 % false positive error rate. We also developed a transmission scheme with two different operating modes using XMPP to guarantee privacy during data transmission. Based on these schemes, we proposed two different architectures providing secure and privacy preserving data processing support for active authentication. As future work, we plan to: 1) conduct experiment on a larger dataset; 2) add more rules to the sanitization scheme to improve the detection accuracy; 3) apply these two architectures to other related scenarios and establish the generalization and applicability of our scheme.

Acknowledgments This research is supported in part by NSF Grant No. CNS: 1314803. Usual disclaimers apply. A preliminary version of this paper was presented at the 6th Secure Knowledge Management Conference at Dubai, UAE in December 2014.

References

Ahmed, A., & Traore, I. (2005). Anomaly Intrusion Detection based on Biometrics. In *Proceedings of the 2005 I.E. Workshop on Information Assurance*. West Point.

Ahmed, W., & Athreya, J. (2013). Data Masking Best Practices. *An Oracle White Paper (June 2013)*.

Bergadano, F., Gunetti, D., & Picardi, C. (2002). User authentication through keystroke dynamics. *ACM Transactions on Information and System Security*, *5*, 367–397.

Garg, A., Rahalkar, R., Upadhyaya, S., & Kwiat, K. (2006). Profiling Users in GUI Based Systems for Masquerade Detection. In *Proceedings of 7th Annual IEEE Information Assurance Workshop* (*IAW 2006*). United States Military Academy, West Point.

Goecks, J., & Shavlik, J. (1999). Automatically Labeling Web Pages Based on Normal User Actions. In *IJCAI Workshop on Machine Learning for Information Filtering*. Stockholm.

Gunetti, D., & Picardi, C. (2005). Keystroke analysis of free text. *ACM Transactions on Information and System Security (ACM TISSEC)*, *8*(3), 312–347.

Gupta, A., Asthana, A., & Gupta, N. (2008). Masquerade Detection using Typing Pattern. In *Proceedings of 2nd National Conference on Challenges and Opportunities in Information Technology* (*COIT-2008*). Mandi Gobindgarh.

Jabber Inc. (1998). Jabber.org.

Johansson, L. (2005). XMPP as MOM. *Greater NOrdic Middleware Symposium* (*GNOMIS*). Oslo: University of Stockholm.

Leggett, J., Williams, G., Usnick, M., & Longnecker, M. (1991). Dynamic identity verification via keystroke characteristics. *International Journal of Man-Machine Studies*, *35.6*(1991), 859–870.

Monrose, F., & Rubin, A. (1997). Authentication via Keystroke Dynamics. In *ACM Conference on Computer and Communications Security.* Zurich, pages 48–56.

Pusara, M., & Brodley, C. E. (2004). User re-authentication via mouse movements. In *VizSEC/DMSEC'04*: *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security.* Washington DC, pages 1–8.

Radhakrishnan, R., Kharrazi, M., & Memon, N. (2005). Data masking: a new approach for steganography? *The Journal of VLSI Signal Processing*, *41*(3), 293–303.

Ravikumar, G. K., Manjunath, T. N., Ravindra, S., & Umesh, I. M. (2011). A survey on recent trends, process and development in data masking for testing. *IJCSI*, 534.

Shavlik, J., Shavlik, M., & Fahland, M. (2001). Evaluating Software Sensors for Actively Profiling Windows 2000 Computer Users. In *Fourth International Symposium on Recent Advances in Intrusion Detection*. Davis.

Yan Sun received his MS degree in Computer Science and Engineering from the State University of New York at Buffalo, 2014. He has several publications in international conferences and his research interests are security and biometric authentication. Currently, he is a PhD candidate at the State University of New York at Buffalo.

Shambhu Upadhyaya is a professor of computer science and engineering at the State University of New York at Buffalo since 1986 where he also directs the Center of Excellence in Information Systems Assurance Research and Education (CEISARE), designated by the National Security Agency and the Department of Homeland Security. His research interests are information assurance, computer security and fault tolerant computing.