Practical Application of Web-Based Authentication using Keystroke Dynamics

Yuankai Guo,[*] Myungho Jung,[†] Junwei Shi,[‡] and Xu Wang[§]
University of Utah

I. INTRODUCTION

Password authentication has been one of the most popular method to identify. However, attackers' techniques to steal password have also been developed in many ways. Therefore, we need different methods to figure out identity other than just plain text of passwords. For example, access from different IP or MAC address can be detected as a suspicious behavior or attack. However, most of the additional authentication algorithms make users inconvenient. Keystroke dynamics can be extracted while typing password as well as used as identity verification[3]. We will show that the existing users don't have to buy an additional device as well as only a small change in the existing servers will be required to apply it.

II. RELATED WORKS

Many algorithms to detect outliers were studied and tested on performance and precision[5]. Various algorithms were used for pattern matching, such as neural network, Support Vector Machines, K-Nearest Neighbors, and Euclidean distance. For most researches, hundreds of patterns from dozens of people were collected and tested. On the other hand, we focused on comparing the performance of algorithms depending on the variation of the users. One of the defects of the result from most existing tests is that they intentionally collected data from a certain group of people. If the group only consisted of experienced users, the data would be biased. Also, the variation of patterns cannot be observed in a short period of data. We formulated a hypothesis on three changes in patterns and tested in well-controlled environment.

III. THREAT MODEL

Id and password login system is ubiquitous for identification on the internet. However, as the number of web services has dramatically increased, it becomes impossible for people to remember many different passwords and most of them are using the same password for all web sites. It would be a problem if a cracker breaks in a insecure server and steals users' passwords, the password can be used for other web services as well.

On the other hand, keystroke dynamics is one of the biometric authentication such as face, fingerprint, and iris recognition[4]. Thus, past data become naturally useless for authentication. Also, a set of patterns can be stores as a different form of structure like a neural network. It makes hard for attackers to imitate the keystroke pattern.

IV. METHOD

A. Data structure

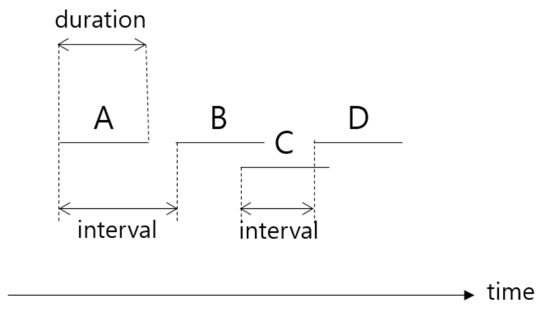

FIG. 1. Timing measurement in interval and duration

The timing vector of keystroke patterns can be measured as intervals between keys and keystroke durations[2]. Therefore, given n characters password, a pattern is represented as an array of $n+1$ numbers. We applied the structure in a slightly different way from other studies. The interval in our project was measured not from previous release time but from press time. By doing so, there is no negative value and storage can be saved by storing only unsigned data. The timing vector can be considered as a point in $n+1$ dimensional space. Then, we can compare similarity using statistical or machine learning algorithms.

The pattern needs to be normalized because the storage is limited and in order to equivalent comparison among users. First, the timing values were divided by the maximum value and stored with the maximum value. Then, the values becomes less than 1 and the data size can be controlled by floating point precision. Also, two different length of passwords cannot be evaluated with the same error bound because of different dimensional space. we solved the problem by multiplying the error bound by $\sqrt{d}$ such that d is the order of dimension.

[*] u0942867@utah.edu
[†] myungho.jung@utah.edu
[‡] sjwarthas2009@gmail.com
[§] xu724@cs.utah.edu

B. Algorithm

To filter imposter's patterns, novelty detection algorithms are required. There are many researches to detect anomaly[1]. We selected and tested three common algorithms, which are the Nearest Neighbor, Mahalanobis distance, and one class SVMs, with changing the patterns.

V. IMPLEMENTATION

A. Architecture

Our system is built based on the B/S model. We use an HTML page as the frontend and program a series of Python sciprts as the backend. The frontend collects the timing pattern from the user and send it with encryption to the backend via AJAX. The backend retrieve previously stored feature pattern from a database and match it against the input with our novelty detection algorithm. If the matching is successful, a positive result is send to the frontend, and vice versa.

A user's inputting habit may change over time. Therefore, we need to regenerate the feature pattern periodically. Every time the authentication is successful, we record the timing pattern from the user into database. And when a certain number of patterns are collected. We run the training routine of our novelty detection algorithm to generate a new feature pattern.

B. Encryption

It's not secure to transmit the password and pattern in plaintext through the network. So we need to protect the password and pattern by encryption or message digest. In our implementation, we use one handshake protocol instead of multiple handshakes for better efficiency. The format of authentication message is $KeyHash(TS|Pattern)|TS|Pattern$ such that TS is timestamp. The Key is 128bit generated from $MD5$ message digest of password. We generate $Hash(TS|Pattern)$ using $SHA-1$, which provides integrity protection for the message. We encrypt the $Hash(TS|Pattern)|TS|Pattern$ by AES algorithm. Moreover, we add a salt to for better diversity. With the timestamp and salt, it's invulnerable to rainbow attack.

How to prevent replay attack? In our implementation, the server record the timestamp. If the server receives a message with the same timestamp which already exists in the time window, the server will refuse the message. Of course, the server will also drop the message with invalid timestamp.

However, there is a drawback in the protocol. Since the client couldn't authenticate the identity of the server, it's vulnerable to in-the-middle attack. Besides, the protocol is not secure when it comes to server break-in attack. The attacker could also modify the time in the server.

VI. ANALYSIS

We created a pattern generator to test algorithms. Our tests are focused on the prediction of the variation of user's patterns and comparing performances of pattern matching methods. The error rates for pattern detection are divided into two parts which are False Acceptance Rate(FAR) and False Rejection Rate(FRR)[2]. The two error rates are calculated for each test case.

A. Environment

Test program is coded in python using Scipy, Numpy, and scikit-learn library. For each test, 1000 evenly distributed patterns of imposters are generated. Also, the same number of patterns of the Gaussian distribution for genuine users are created. If our assumption is correct, the distribution of patterns will be similar to actual patterns from a user and imposters.

Our tests are focused on the variation of the patterns over time. First, only an interval of patterns would be fluctuated. Suppose that the password is mixed with a sequence of alphabets and that of numbers like 'foo1234.' In this cases, the patterns of a word and numbers may be stable except the interval between them. Second, we assume that the user will type the password fast or slow; however, the pattern would be proportional. The third assumption is that the user may use different types of keyboards such as laptop, desktop keyboard, and touch keyboard on mobile devices. Then, we expected that the patterns will form multiple clusters.

1. Test Results

Most of the error rates are not visible because they are very close to 0. As seen in the three tests, we concluded that the performances of the three algorithms are not much different regardless of the variation of the patterns. The most accurate method among them was the nearest neighbor method, the SVMs was next, and the Mahalanobis distance showed the worst performance. With respect to computation time, the SVMs was the fastest, the nearest neighbor was next, and the Mahalanobis distance was the slowest. However, the difference in performance may come from different libraries used. The SVMs and nearest neighbor using ball tree algorithm were implemented using $scikit-learn$ library. On the other hand the Mahalanobis distance was measured using $scipy$ library.

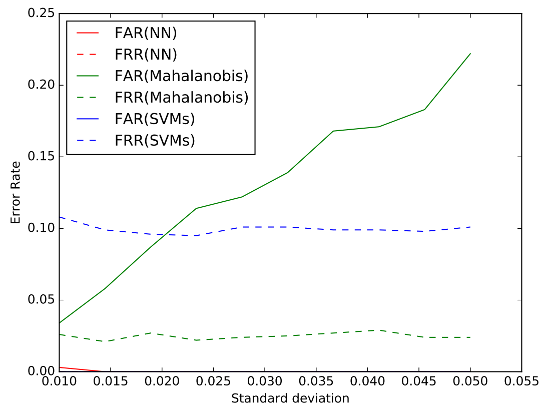

FIG. 2. Test result of patterns that only one of the intervals varies

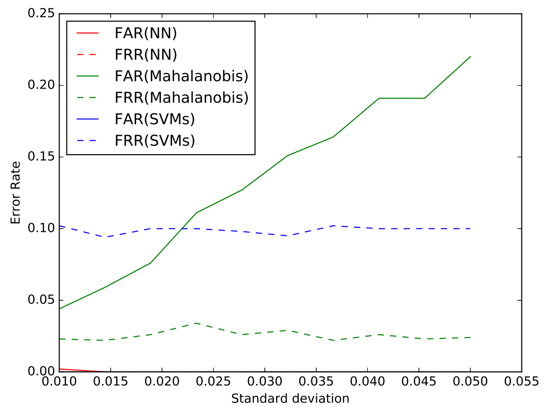

FIG. 3. Test result of proportional patterns

VII. DEFECTS

Although many researches and practical application of Keyboard Dynamics exist, there are problems in keystroke authentication. Assume that the mechanism is applied for Web-based system. Then, people can move cursors using arrow keys or mouse while typing and editing password. We just ignored the cases and assumed that the web page allows only password typed at once. Though it would make the users inconvenient, it will make the system more secure. Also, key stroke patterns may be able to be recorded by spyware with key logger. In general, we don't have to worry about the attack because the pattern is encrypted before sent. If the encryption cannot be applied on the system, it can be protected by the attack by sharing a nonce with the server and adding it to the pattern. lastly, a user's patterns change more frequently and radically than other physiological traits or behavioral characteristics, and thus it would be a problem. We suggest a solution that evaluates only the latest patterns like sliding window.

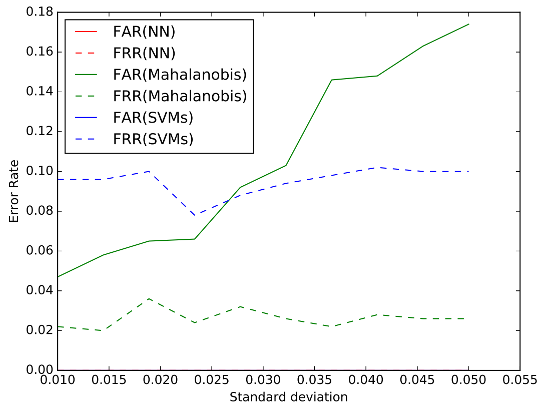

FIG. 4. Test result of patterns of multiple clusters

VIII. CONCLUSION

Through the project, we checked the algorithm is valid by demo program and also tested the performance between algorithms. We suggested that the possible web-based authentication using keystroke dynamics by implementing and testing actual login web page. Unfortunately, it is not obvious that the test data is significant compared to that from other research. However, the application developed for the experiments will be useful to test other algorithms with a myriad of normally distributed patterns and variations. It will help to create various environments for next experiments.

[1] Salil P Banerjee and Damon L Woodard. Biometric authentication and identification using keystroke dynamics: A survey. *Journal of Pattern Recognition Research*, 7(1):116–139, 2012.

[2] Sungzoon Cho, Chigeun Han, Dae Hee Han, and Hyung-Il Kim. Web-based keystroke dynamics identity verification using neural network. *Journal of organizational computing and electronic commerce*, 10(4):295–307, 2000.

[3] R Stockton Gaines, William Lisowski, S James Press, and Norman Shapiro. Authentication by keystroke timing: Some preliminary results. Technical report, DTIC Document, 1980.

[4] M Karnan, M Akila, and N Krishnaraj. Biometric personal authentication using keystroke dynamics: A review. *Applied Soft Computing*, 11(2):1565–1573, 2011.

[5] D Shanmugapriya and Ganapathi Padmavathi. A survey of biometric keystroke dynamics: Approaches, security and challenges. *arXiv preprint arXiv:0910.0817*, 2009.