Effects of Text Filtering on Authentication Performance of Keystroke Biometrics

Jiaju Huang, Daqing Hou, and Stephanie Schuckers
Electrical and Computer Engineering, Clarkson University,
Potsdam NY, USA

Shambhu Upadhyaya
Computer Science and Engineering, University at Buffalo,
Buffalo NY, USA

***Abstract*— Free text keystroke dynamics is a behavioral biometric that has the strong potential to offer unobtrusive and continuous user authentication. In free-text keystroke biometrics, users are free to type whatever they want to while still being authenticated. However, not all keystrokes from a user exhibit the same quality of stable patterns that can be used to differentiate them from others. The "unstable" keystrokes may originate from such activities as when the user is playing a computer game, or other sources of noisy or "gibberish" text. Our hypothesis is that some forms of text negatively impact keystroke dynamics-based user authentication, and thus, should be filtered out. This study investigates the impact of gibberish text on authentication performance through locating and removing gibberish text, and comparing the difference in authentication performance before and after the removal. We confirm the positive effect of text filtering on authentication performance, especially on the reduction of the false reject rate.**

Keywords— keystroke biometrics; text filtering; performance

I. INTRODUCTION

The increasing adoption of automated information systems coupled with the pervasive use of mobile devices has greatly simplified our lives, but also makes us overwhelmingly dependent on computers and digital networks. In the current digital age, user authentication technologies are critical for preventing online crimes and ensuring adequate cybersecurity.

Keystroke dynamics, a kind of behavioral biometrics [1], is especially promising and attractive for defending cybersecurity [2]. This is because unlike other biometric modalities that may be expensive to implement, keystroke dynamics is almost free. The only hardware required is the keyboard, which people already have anyway. Keystroke dynamics is also unobtrusive, so people can work unhindered as long as they pass the authentication. Lastly, instead of other one-stop authentications such as verifying the user id and password, keystroke dynamics offers continuous authentication. It continuously authenticates users as long as they are using the keyboard.

This paper presents the first study that evaluates the impact of data quality on the authentication performance of keystroke dynamics. While biometric data quality [3] has been investigated in recent years for biometric modalities such as iris [4], no groups have yet studied quality of keystrokes and their performance impact [2]. By far most previous research on keystroke dynamics either asks participants to type fixed pre-defined strings [5], or perform tasks such as writing emails [6] or answering questions [7]. These researches have not touched on the problem of noisy or gibberish keystrokes, an issue that any practical applications of keystroke dynamics must address.

In free-text based authentication, users may type whatever they want to. While it is probably safe to state that most keystrokes that a user types on a computer are natural languages, they may also come from other non-textual activities such as gaming, keyboard shortcuts, or simply mindless, random typing. In this paper, we loosely call such non-textual keystrokes "gibberish."

We believe that gibberish text, when mixed up with normal text, will have a negative impact on authentication performance. This is because different typing activities with different goals may result in different motor behaviors [2]. When typing normally, users anticipate the next few keys they are going to type, and type at a pace they are comfortable with. This process reveals the inherent typing patterns that serve as the basis for authentication. Therefore, such text is "useful" keystrokes for user authentication. The situation is quite different for gibberish keystrokes, where users do not know or care about what they are going to type next, such as gaming or random typing. Such keystrokes are therefore "useless," i.e., they have limited identifying information.

Since keystroke dynamics rely on digraph latencies and their distributions as primary features, it is reasonable to hypothesize that gibberish text may distort the distributions of digraph latencies and negatively affect the authentication performance of keystroke dynamics.

Luckily, with the exception of passwords, we observe that normally, a significant proportion of a user's keystrokes are useful, intelligible text. By filtering out the gibberish text, we are able to remove the noise and keep the useful keystrokes. In our study, we explore the types of gibberish typing, algorithms for detecting them, and improvement of authentication performance as a result of removing gibberish keystrokes.

This paper contributes the first study that evaluates the types of gibberish text in keystroke dynamics and the impact of their removal on authentication performance. A novel dataset has been collected of individuals during their normal computing behavior. For subjects who signed consent, keystrokes were recorded in an open computing lab while subjects performed their normal work (homework, email, etc.). Unlike previous datasets where there is some control of users during the free text collection, this collection was from completely uncontrolled, natural behavior. This has also raised the opportunity to study noisy keystrokes, the subject of this

978-1-5090-1138-4/16/$31.00 ©2016 IEEE

paper. The dataset and algorithms used in this paper will be made available upon request.

The rest of the paper is organized as follows. Section II presents the setup of our study. Section III defines four types of gibberish text, two kinds of filtering rules, and the amounts of gibberish text detected in our dataset. Section IV evaluates the impact of removing gibberish text on authentication performance. Section V explores the causes for the negative impact of gibberish. Lastly, Section VI concludes the paper.

II. Study Setup

In this section, we describe our free text keystroke dataset and the two baseline algorithms [6, 8] that we re-implemented in order to evaluate the difference in authentication performance before and after gibberish text is filtered from our dataset. These two algorithms are chosen because they represent the state-of-art in free text authentication [6, 8].

A. Free Text Keystroke Dataset

We gathered free text keystrokes from users who volunteered to contribute their data to support our research under an approved data collection protocol. Users were asked to work in a campus student computer laboratory, but otherwise they were free to type whatever they wanted to. The laboratory included a keystroke logger that recorded keystrokes for only those users who registered for the study. Users were able to turn on and off the logger as needed.

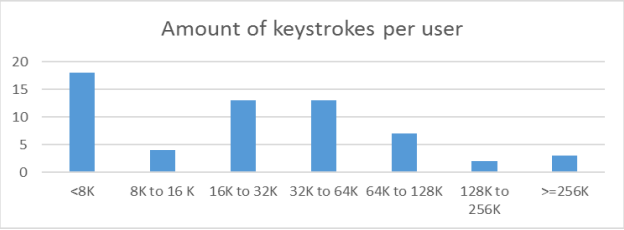


Fig. 1. Distribution of amounts of keystrokes from the 60 volunteer users

Currently, there are a total of 60 users registered. Fig. 1 depicts a histogram for the amounts of keystrokes contributed by the 60 users. On average, our users have contributed 21 hours of typing (max 76 hours) over a period of 181 days (max 550 days). We required a user to have at least 11,000 keystrokes in order to run the baseline keystroke algorithms for performance evaluation. Thirty-eight of the 60 users have contributed more than 11,000 keystrokes, but filtering will reduce the number of users possessing the required keystrokes.

B. Baseline One: Gunetti&Picardi's Algorithm

We implement Gunetti & Picardi's algorithm [6] as a baseline for evaluation because it is considered the state-of-the-art [9] in free text keystroke dynamics. Informed by a previous study on sizes of reference and test samples versus performance [10], we use 1,000 consecutive keystrokes to form a sample. The data for each user is divided into samples of 1,000 keystrokes. Each user profile comprises 10 randomly selected samples. The remaining samples are used as genuine tests. Performance metrics are calculated for the user. Random sampling is repeated 50 times, and the average of the performance metrics from all the repeats is calculated as the final performance metrics for the user, and reported.

Briefly, given a test sample X and a user A, Gunetti & Picardi's algorithm verifies whether X comes from A, as follows. More details can be found in their paper [6]:

1. For each legal user U in the system, calculate two metrics $m(U)$ and $md(U,X)$, where $m(U)$ is the average distance between all the samples in U's profile, and $md(U,X)$ is the average distance between all the samples in U's profile and the test sample X.
2. If there exists any other legal user B of the system, to whom the test sample X is closer than to the verified user A, that is, $md(B,X) \leq md(A,X)$, then reject X for user A. Otherwise, X is closest to A in the system; conduct a further test in the next step to ensure that X is sufficiently close to A's profile samples.
3. Furthermore, if test sample X is closer to the core of user A's profile samples than the average distance between themselves ($m(A)$). That is, if $md(A,X) \leq m(A)$ is true, then accept X. Otherwise, $md(A,X) > m(A)$ holds, and check the next condition.
4. Finally, accept X, if $md(A,X)$ is closer to $m(A)$ than to any other $md(B,X)$ computed by the system, i.e., $md(A,X) - m(A) < md(B,X) - md(A,X)$.

The distance between two samples S1 and S2 is measured in terms of n-graphs (n consecutive keystrokes typed by a user), using "A" measure (for *absolute*; using the difference in average n-graph latencies to determine similarity) and "R" measure (for *relative*; using the relative ordering of average n-graph latencies to determine similarity).

The "A" measure between $S1$ and $S2$ is defined in terms of the n-graphs they share and a constant value t, as follows:

$A_{t,n}$ (S1,S2) = 1 - (# of similar n-graphs shared by S1 and S2) / (total number of shared n-graphs).

A constant t is needed for defining n-graph similarity. Specifically, let $G_{S1,d1}$ and $G_{S2,d2}$ be the same n-graph occurring in typing samples S1 and S2, with durations $d1$ and $d2$, respectively. We say that $G_{S1,d1}$ and $G_{S2,d2}$ are similar if $1 < max(d1,d2)/min(d1,d2) \leq t$, where t is some constant greater than 1 (t = 1.25).

C. Baseline Two: Leggett's "Zone-of-acceptance"

The "zone-of-acceptance" algorithm by Leggett et al. [8] assumes that the latencies for all occurrences of a digraph in the reference profile follow a normal distribution $N(\mu,\sigma)$. If the average latency for a digraph in the test sample falls between the acceptance region ($[\mu - d*\sigma, \mu + d*\sigma]$), the digraph is then considered accepted; otherwise, rejected.

The similarity score for a test sample is in turn defined as the ratio of the number of accepted digraphs over the total number of digraphs appearing in the test sample. The test sample is accepted if its similarity score is greater than or

equal to a set threshold. In this study, we set both d and the acceptance threshold to 0.6, for all users.

III. Exploring Gibberish Text in Our Dataset

We have identified four types of gibberish keystrokes by manually inspecting the data. In this section, we formally define the four types of gibberish keystrokes, the two groups of algorithms that we have created for removing gibberish text, and the amounts/percentages of gibberish keystrokes that are detected from our dataset.

TABLE I. Four types of gibberish keystrokes

Gibberish type	Example
Repetition: Repeating the same characters at least 3 times.	aaaaaaaaaaaaa
Gaming: gaming patterns, any combinations of the four keys used for movement in games ('a','s','d','w') and space.	awdaswdaswdsd wsa
Distinct: Strings with too few distinct characters (moving window of 15 characters with no more than 5 distinct characters). Parameters are chosen manually.	uijiuijiujijii
Length: Long strings (greater than 20) of alphabetical letters. Motivated by observation on maximal English word length.	jkfjkjfsdjkfsdjkhfjk ds

A. Patterns of Gibberish Text

Using our dataset, we explored and identified four patterns of gibberish text: repeating characters, gaming patterns, long string with few distinct characters, long string with no white spaces or separators. Table 1 shows the definition of each case along with an example.

B. Algorithms for Filtering Gibberish Text

We have experimented with two kinds of algorithms for removing gibberish keystrokes. One is based on regular expressions, and the other on a spell checker.

Regular expressions

We use regular expressions to detect and remove four patterns of gibberish keystrokes. Before filtering, there are 38 out of the 60 users who have more than 11,000 keystrokes. These 38 users contribute a total of 2.90 million keystrokes. The Repetition filter detects 427K, or 14.7% of the total keystrokes. The Gaming filter detects 326K, or 11.2%. The Distinct filter detects 620K, or 21.4%. The Length filter detects 548K, or 18.9%. Applying the combined of the four regular expression filters detects 676K, or 23.3%. Notice that there are overlaps in the keystrokes that the four filters detect.

Spell checker

Another way of removing gibberish is to identify and accept only good textual keystrokes. We implement this by checking whether the keystrokes are at least correctable English words. Since there can be typos and other variations for an English word, we need more than a dictionary lookup. To this end, we have utilized Norvig's spell checker, which is based on statistical language models [11]. A token that is not an English word and cannot be corrected is discarded. This allows correctable typos to be accepted as normal text.

We also implement a context-based filtering, on the rationale that by chance there can be correctable words even in a completely random string. Context-based filtering prevents the correction of such tokens. Specifically, if the neighbors of the current token do not have enough correct or correctable words, we do not correct it. We use five tokens to the left and five to the right of the current token as its neighbors, and require that a token have at least seven correctable neighbors.

IV. Evaluating Authentication Performance

In this section, we compare the performance of the Gunetti & Picardi algorithm, with and without filtering, using (1) the proposed regular expressions, (2) the spell checker, and (3) their combination. We measure the authentication performance using two standard metrics:

- FAR (False Accept Rate): The ratio of impostor attacks that are falsely accepted as genuine users.
- FRR (False Reject Rate): The ratio of genuine tests that are falsely rejected as impostors.

We run multiple experiments to produce the authentication performance (FAR and FRR) by applying the various filtering rules. Specifically, for each filtering rule, we filter out the gibberish text and evaluate the algorithms using the remaining data. We test the statistical significance of the performance improvements for FAR and FRR before and after the filtering, at the p=0.05 level, using the Wilcoxon Signed Rank Test.

A. Effects of Filtering Using Regular Expressions

Fig. 2 shows the FAR before and after gibberish keystrokes are filtered using regular expressions. *Overall, FAR remains similar before and after filtering, and none of the FAR from the filtered data is statistically different from that without filtering.*

In contrast, as shown in Fig. 3, we have achieved significant improvements in FRR. With the exception of the Repetition filter, all other regular expressions produce a better average FRR than that before filtering, and combining all the regular expression filters yields the best result. Moreover, *the Distinct filter and the combined both yield results that are statistically significantly better than the FRR before filtering.*

Note that 38 users have over 11,000 keystrokes, but there are only 37 users in Fig. 2 and Fig. 3. This is because after filtering, one user (user 38) does not have enough keystrokes required for performing the evaluation, and thus, is removed. Upon closer inspection of the data, we conclude that despite our warning in the consent form, this user has contributed mostly gibberish text, possibly to receive payment faster.

B. Effects of Filtering Using Spell Checker

The regular expressions that we have employed in Section IV.A are conservative in the sense that each of them will

remove only one specific pre-defined pattern of gibberish keystrokes from the data. In contrast, the spell checker is more aggressive and capable of removing more gibberish, but may also mistakenly remove meaningful keystrokes such as foreign languages and programming codes. Therefore, these two types of filters would complement each other. To understand their relative strengths, in this section, we compare the effects of the regular expressions, the spell checker, and their combination, on authentication performance. We conclude that *the different filters positively affect FRR but have no effect on FAR, and that combining regular expressions and the spell checker yields the highest statistically significant improvement in FRR.*

We compare the FAR and FRR from four sets of experiments: vanilla text, filtering with the regular expressions, with the spell checker, and their combination. The FAR and FRR for all users are plotted in Fig. 4 and Fig. 5.

As shown in Fig. 4, FAR (False Accept Rate) for the four situations are very similar, with means of 0.47%, 0.52%, 0.51%, and 0.49%, respectively. The difference in FAR between any two of them is not statistically significant.

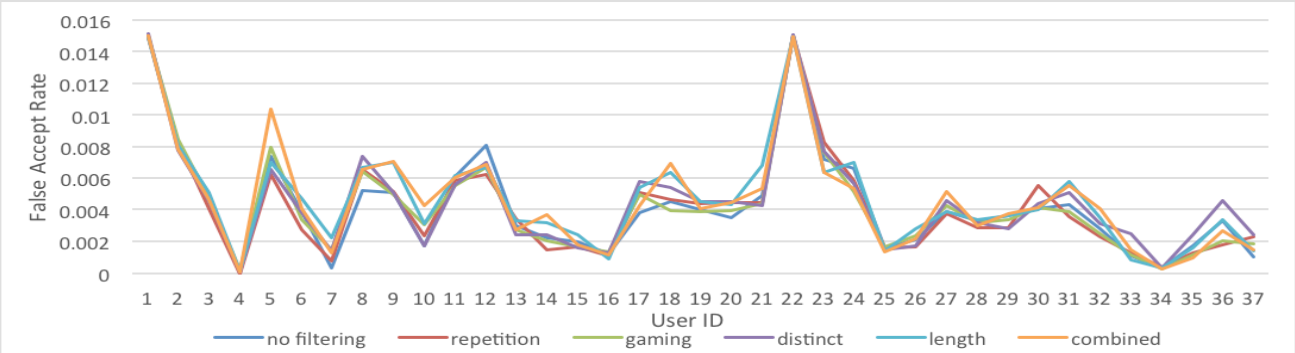


Fig. 2. FAR (False Accept Rate) before and after filtering using the regular expressions.

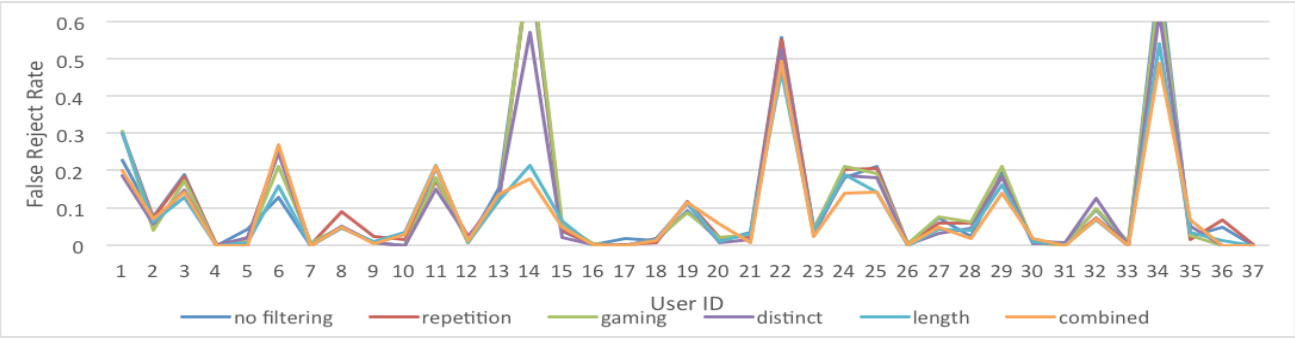


Fig. 3. FRR (False Reject Rate) before and after filtering using the regular expressions.

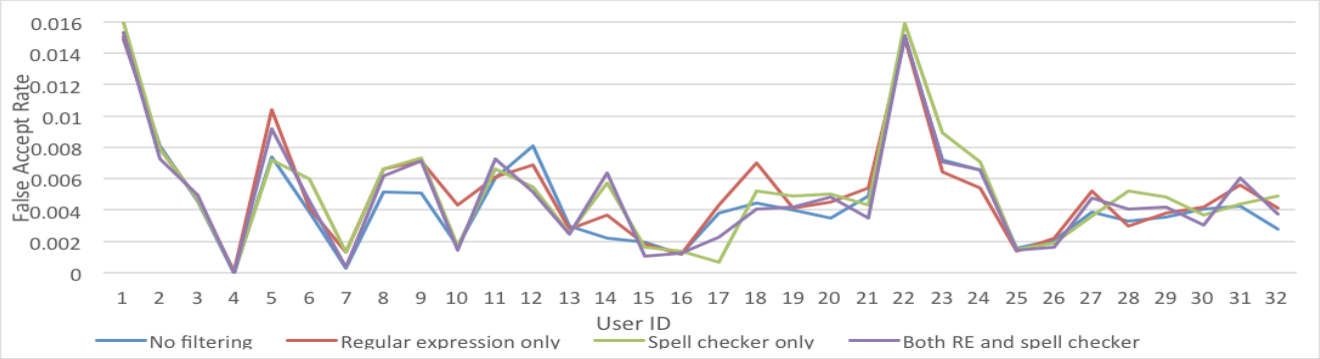


Fig. 4. False Accept Rate for different strategies of filtering gibberish text.

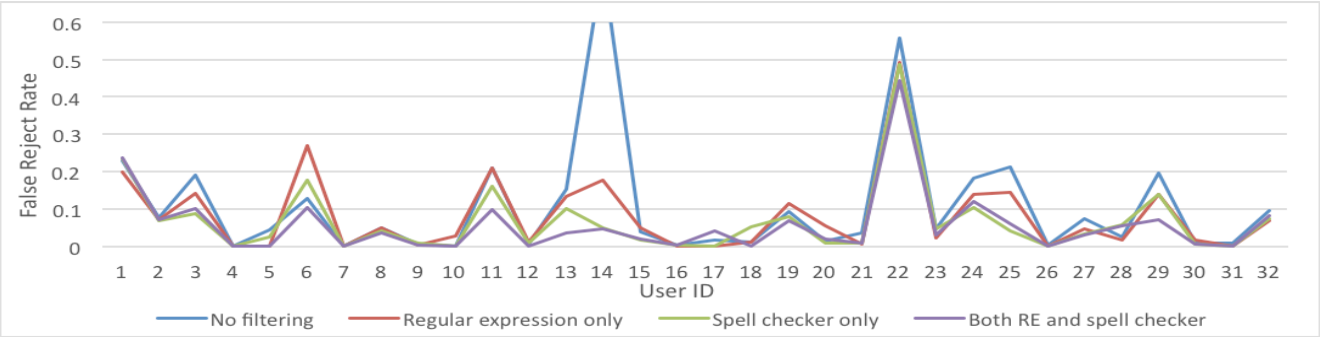
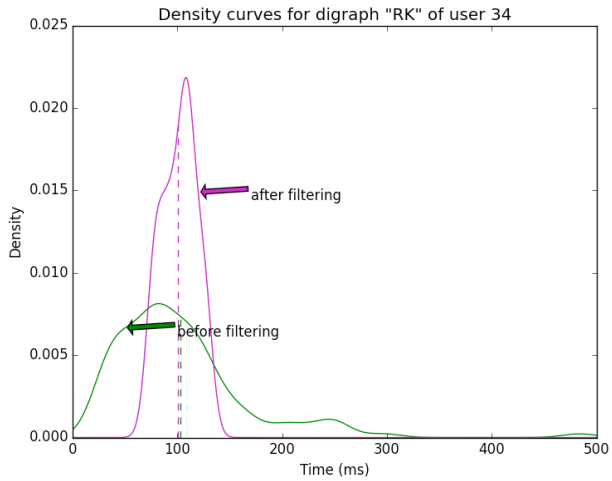


Fig. 5. False Reject Rate for different strategies of filtering gibberish text.



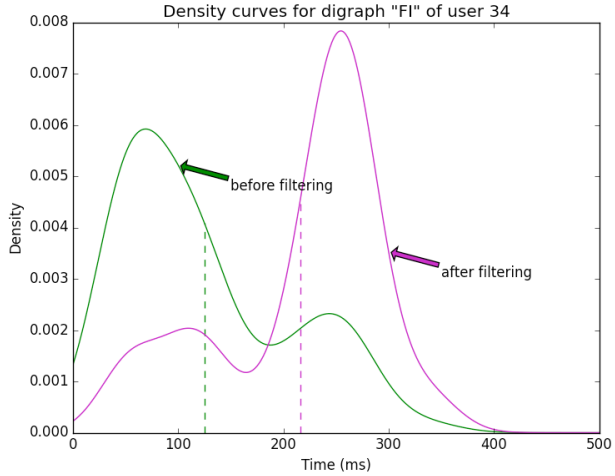


Fig. 6. Density curves of "FI" and "RK" before and after filtering. The vertical dashed lines indicate the mean digraph latencies.

Unlike FAR, however, as shown in Fig. 5, there are clearer differences in FRR (False Reject Rate). The mean FRRs for all the users in the four situations are 10.8%, 8.3%, 6.5% and 5.6%, respectively. We find that all three filtering strategies have yielded statistically better FRR than that of the vanilla text, that the spell checker improves FRR better than the regular expressions, and that combining the two yields the largest improvement. All of these differences are statistically significant.

Adding context to the spell checker (See Section III.B) does not make a significant difference in authentication performance. The performance of using a context-based spell checker and a regular spell checker are similar, with an average FAR of 0.52% and FRR 5.79% for a context based spell checker, and an FAR of 0.49% and FRR of 5.62% for the regular spell checker without context. The differences are not statistically significant.

Note that there are only 32 users shown in Fig. 4 and Fig. 5, fewer than the 37 users in Fig. 2 and Fig. 3. This is because five more users are removed from the comparison as they do not have the minimum amounts of keystrokes

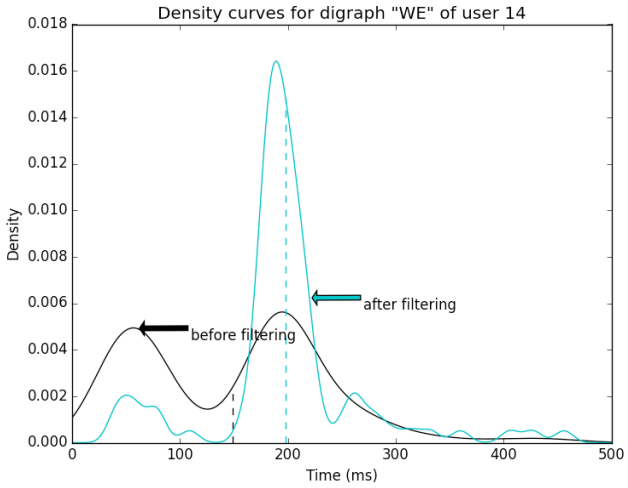


Fig. 7. Density curves of "WE" before and after filtering.

required any more after the combined filtering is applied, which are more aggressive in filtering out gibberish than the regular expressions alone.

V. Why Gibberish Text Worsens FRR?

Our evaluation in Section IV shows that gibberish text does not cause significant differences in FAR (False Accept Rate) but significantly worsens FRR (False Reject Rate). The lack of effect on FAR can be explained by the fact that different users tend to have distinctive distributions of digraph latencies regardless of gibberish text. This helps separate them apart and avoids false acceptance of impostors. To confirm this hypothesis, we have randomly selected multiple digraphs and visually compared their density curves. We conclude that these digraph distributions indeed tend to be distinct. In the rest of this section, we shall focus on investigating why gibberish text worsens FRR.

A. The Probability Density of Digraph Latencies with and without Gibberish Text

To explore the impact of gibberish keystrokes on the distribution of digraph latencies, we first identify those user who have the highest ratios of gibberish keystrokes and then compare the probability density curves for the most frequent digraphs from these users, before and after filtering. We make two observations of the impact that gibberish text has on the distribution of digraph latencies.

Our first observation is that because gibberish keystrokes are often typed faster than normal text, the mean of digraph latencies in the vanilla text (without filtering) are often smaller than those in the filtered text. For example, in Fig. 6, the mean latency for user 34's digraph "FI," before filtering, is about 120 ms, with a peak at near 80 ms; after filtering, the new mean is shifted to the right, near 210 ms, with a new high peak at around 250 ms.

Our second observation is that in most of our data, gibberish keystrokes tend to create a distribution with a wider range for digraph latencies than normal text. As shown by the example for "RK" in Fig. 6, while both

distributions before and after filtering have a mean near 100ms, the before-filtering distribution has a much wider range than the after-filtering distribution.

Most keystrokes show a combination of these two effects. That is, the distributions of digraph latencies for the original data tend to both have a smaller mean and a wider range than the filtered data. Fig. 7 depicts such an example for "WE."

Taking into consideration the above two observations, along with the fact that the baseline algorithms rely on the zone of acceptance to measure digraph similarity, it is not too hard to explain why gibberish keystrokes cause the poorer FRR. As shown in Fig. 6, after filtering of gibberish text, the digraph has a narrower distribution. For the zone of acceptance, the narrower the distribution, the larger the area of the density curve that falls in the zone, hence the better chance for a genuine test to be accepted. For example, user 34 has a similar mean and zone of acceptance for digraph "ER" before and after filtering (not shown). Before filtering, only 27% of the density curve falls within the acceptance region, which ranges between 0.8 to 1.25 times of the mean, whereas in the filtered text, the area is increased to 43%, a much higher chance for a genuine test to be accepted, hence a lower FRR.

Furthermore, recall that Gunetti & Picardi's algorithm defines the zone of acceptance as a function of the mean (for the same digraphs in the two samples, the larger mean divided by the smaller mean must be within 1.25, therefore one mean must be within 0.8 to 1.25 times of the mean of another). Since gibberish keystrokes tend to render a smaller mean for the digraph latencies, it also makes the zone of acceptance smaller, hence resulting in a larger FRR.

These two factors combined cause users with a high ratio of gibberish text to perform poorly in FRR.

B. Comparing Gunetti&Picardi's and Leggett's Algorithms

Recall that Gunetti & Picardi's algorithm is based on both the A measure and R measure for measuring similarity [6]. In Section V.A, we have analyzed the impact of gibberish text on the distribution of digraph latencies, showing that gibberish text negatively affects the A measure and FRR. However, its impact on R measure is less obvious, which measures the similarity between two samples in terms of the ranking of latencies for the shared digraphs.

To contrast, we select Leggett's zone of acceptance algorithm [8] as a second baseline algorithm for performance evaluation. Our experiment shows that the zone of acceptance algorithm is similar as Gunetti & Picardi's algorithm in terms of the effects of gibberish text. With a digraph threshold of 0.6 standard deviation and a similarity threshold of 0.6, we get an average FAR of 2.38% and FRR of 23.5% for vanilla data, and FAR of 2.68% and FRR 11.3% after filtering. The result is consistent with our earlier finding that filtering does not change FAR but significantly reduces FRR.

However, the performance improvement in FRR from filtering is much greater for the zone of acceptance algorithm [8] than the Gunetti & Picardi's algorithm [6]. This suggests that the zone of acceptance algorithm is more sensitive to gibberish keystrokes than Gunetti & Picardi's algorithm. In other words, Gunetti & Picardi's algorithm is more robust to gibberish text, which can be attributed to two unique aspects in its design: (1) The use of R measure, which measures the difference in relative ranking of digraph latencies rather than the absolute values. (2) Authenticating a user by ranking the similarity scores from all users and selecting the most similar one. Future work is needed to further discern the impact of these two aspects.

VI. CONCLUSION

Using a novel free-text keystroke dataset of our own, we find that gibberish keystrokes make up nearly a quarter (23.3%) of all the keystrokes. Moreover, addition of gibberish keystrokes has no impact on false accept rate but increases false reject rate significantly. Filtering with a spell checker yields a larger performance improvement than regular expressions, but combining both filters produces the best improvement. We explain the reasons why gibberish text worsens FRR because it changes the mean and range of the distribution of the digraph latencies. Lastly, filtering implies that a larger test sample is needed before an authentication can be attempted. Future work needs to investigate its implication on the usability of this modality.

ACKNOWLEDGMENT

This work was supported by United States NSF Awards #1314803 (Buffalo) and #1314792 (Clarkson).

REFERENCES

[1] R. V. Yampolskiy and V. Govindaraju, "Behavioural biometrics: a survey and classification," International Journal of Biometrics, vol. 1, 2008.

[2] S. P. Banerjee and D. L. Woodard, "Biometric authentication and identification using keystroke dynamics: a survey," Journal of Pattern Recognition Research, vol. 7, pp. 116-139, 2012.

[3] NIST. "Biometric quality homepage," URL:http://www.nist.gov/itl/iad/ig/bio_quality.cfm. Last accessed 10/19/2016.

[4] K. P. Hollingsworth, K. W. Bowyer, and P. J. Flynn, "The best bits in an iris code," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 31, 2009.

[5] K. S. Killourhy and R. A. Maxion, "Comparing anomaly-detection algorithms for keystroke dynamics," Dependable Systems & Networks, 2009.

[6] D. Gunetti and C. Picardi, "Keystroke analysis of free text," ACM Transactions on Information and System Security, vol. 8, pp. 312–347, 2005.

[7] E. Vural, J. Huang, D. Hou, and S. Schuckers, "Shared research dataset to support development of keystroke authentication," Biometrics (IJCB), 2014.

[8] J. Leggett and G. Williams, "Verifying identity via keystroke characterstics," International Journal of Man-Machine Studies, 1988.

[9] A. A. Ahmed and I. Traore, "Biometric recognition based on free-text keystroke dynamics," IEEE transactions on cybernetics, vol. 44, pp. 458-472, 2014.

[10] J. Huang, D. Hou, and S. Schuckers, "Effect of data size on the performance of free-text keystroke authentication," Identity, Security and Behavior Analysis (ISBA), 2015.

[11] P. Norvig. "How to write a spelling corrector," URL: http://norvig. com/spell-correct.html. Last accessed 10/19/2016.