Don't listen to my Keystroke Dynamics!

Denis Migdal, Christophe Rosenberger

Don't listen to my Keystroke Dynamics!

Denis Migdal, Christophe Rosenberger

February 2019

Abstract

Keystroke Dynamics enables the authentication or identification of users by analyzing their way of typing, e.g. when browsing the Internet. Most studies in the state of the art focus on increasing Keystroke Dynamics Systems performances. In this paper, we address the problematic of avoiding biometric authentication with keystroke dynamics in order to protect users' privacy against unwanted authentication/identification. Experimental results obtained on significant datasets show the benefits of the proposed approaches.

1 Introduction

As any biometric authentication solution, a keystroke dynamic system (KDS) is composed of two main modules: the enrollment and the verification modules. Each user must enroll himself/herself in the KDS in order to compute its biometric reference template given multiple samples (*i.e.*, several inputs of the password) acquired during the enrollment step. For each input, a sequence of timing information is captured (*i.e.*, time when each key is pressed or released) from which some features are extracted (*i.e.*, latencies and durations) and used to learn the model which characterizes each user. During a verification request, the claimant types his/her password. The system extracts the features and compares them to the biometric reference template of the claimant. If the obtained distance is below a certain threshold, the user is accepted, otherwise he/she is rejected.

First works on KD have been done in the eighties [2], although the idea of using a keyboard to automatically identify individuals has first been presented in 1975 [14]. In the preliminary report of Gaines *et al.* [2], seven secretaries typed several paragraphs of text and researchers showed that it is possible to differentiate users with their typing patterns. Since then, several studies have been done, allowing to decrease the quantity of information needed to build the biometric reference, while improving the performances [15, 11, 13, 9, 4]. However, to the knowledge of the authors, no studies has yet tried to decrease their performances in order to protect users' privacy against unwanted authentication or identification.

The article is organized as follows. Section 2 focus on possible attacks using keystroke dynamics biometric modality. Possible protection schemes are given in section 3 illustrated with different experimental results. Section 4 concludes and gives some perspectives of this study.

2 Attack

We present in this section the attack that can been done in order to identify anybody.

2.1 Attacker model

The attacker is able to execute arbitrary JavaScript code on the users' browser in order to authenticate them, using only the keyboard events' timestamps. We assumed the typed text to be fixed, s.a. a login, e-mail, or password.

The attacker is able to measure the timestamps of keyboard events she/he receives with the javacript

function `Date.now()`. Thus, modifying the events' timestamps will have no effect, as the attacker can measure them himself. However, events can be delayed, i.e. waiting some time before sending the keyboard event. As JavaScript events loop is monothreaded, any active wait is troublesome and will be easily detected by the attacker using `setInterval()`, thus requiring the delayed event to be destroyed, and recreated after a passive wait with `setTimeout()`.

The attacker has an *a priori* on the user's identity, and will be able to use any Keystroke Dynamics Systems, and to perform any pre-processing, in order to authenticate him. The way the Keystroke Dynamics is protected, and the eventual parameters of such anonymization scheme is also assumed to be known by the attacker. Thus, such parameters should be fixed for all users in order to prevent the attacker from using them to discriminate users through browser fingerprinting techniques [1].

2.2 Datasets

There exist many keystroke dynamics datasets [10]. We decided in this work to focus on fixed text datasets (i.e. where users typed the same passphrase). Datasets have been cleaned to remove incoherent data, e.g. entries in which the user did not type the asked text. This corresponds to 13% of entries in GREYC W, and less than 3 entries for other datasets.

In order to get comparable sets, only the first 45 entries per users is kept, users with less than 45 entries, and datasets with less than 45 users, are discarded. From the existing fixed-text datasets, only 3 matched our criteria. From these 3 datasets, we build 4 datasets composed of a fixed text Keystrokes for each user (one having 2 fixed text, 2 datasets are thus created). Table 1 gives the datasets used in this work.

2.3 Attack performance

As the number of collected samples during the enrollment step is usually low, many Keystroke Dynamics Systems are based on a distance. In the scope of this article, the attacker uses the Hocquet distance function [6]:
We aim at computing a distance between two templates K_A and K_B. We suppose that the template K_A is associated by μ and σ the average value of biometric samples and the standard deviation (note that $0/0$ is assumed to be 0).

$$STAT2 = 1 - \frac{1}{n}\sum_{i=1}^{n} e^{-\frac{|K_B(i)-\mu_i|}{\sigma_i}} \quad (1)$$

In the scope of this paper, the templates are composed of the gap and dwell durations for each typed key, i.e. the duration between two consecutive key press, and the time a key is pressed. The 10 first templates of each user are used for the reference template computation.

The capacity of an attacker to authenticate an user will be quantified with the maximal estimation of the Equal Error Rate (EER), which corresponds to configuration of the biometric system when FAR equals FRR. The False Acceptance Rate (FAR) describes the ratio of accepted impostor data, the False Rejection Rate (FRR) describes the ratio of falsely rejected legitimate users.

The performance of a KD Anonymization Scheme (KDAS) will thus be quantified as the minimum of the maximal estimation of the EER for each possible KDS and pre-processing. For a given KDS and pre-preprocessing, if the KDAS is not deterministic, the

Name	Text	# of users (45)	Clock resolution	EER	Source
GREYC K	greyc laboratory	104	10.0144ms	14.75%	[5]
GREYC W1	laboratoire greyc	62	1ms	14.40%	[5]
GREYC W2	sésame	46	1ms	25.39%	[3]
CMU	.tie5Roanl	51	0.2ms	19.38%	[8]

Table 1: Description of used datasets.

KDAS is tested 20 times, and the mean of the maximal estimation of the EER for each test is used. If the dataset is not indicated, the number given is the mean of the number for each of the 4 datasets used in this study.

2.4 Attacker pre-processing

The timestamp of a given event depends of the resolution and jitter of the clock used to measure it. The *resolution* is the mean time between two clocks tics, and the *jitter*, the difference between the theoretical clock tic timestamp and its real timestamp. This mean that an event occurring at a time t will have a timestamp of $\lfloor t/r \rfloor * r + j$, with r the clock resolution, and j a random noise (the jitter). Existing studies have found that the clock resolution influence KDS performances [7], and discretization might improve KDS performances [4].

Values where discretized using 1,001 different resolutions (from 0 to 1 by step of 1/1,000). As shown in Figure 1, attacker might expect slight (J $\simeq$ 0.02 for GREYC W1) or negligible (J $<$ 0.005) EER improvement by doing so. Figure 6 shows the discretization can both increase or decrease the EER depending on the resolution.

Jitter can be removed with the following formula : $t' = \lceil t/r \rfloor * r$. As shown in Figure 1, it has negligible influence on the EER (diff $<$ 0.004, and noJ $\simeq$ J), and thus does not need to be removed.

3 Protection

We propose different solutions to anonymize keytroke dynamics of users.

3.1 Costless protection

Release keyboard events can be automatically generated a constant time after the pressure event e.g. 2ms (A). As shown in figure 2, such strategy increase significantly the EER ($0.044 \leq$ A ≤ 0.117).

Users' screen typically draw a frame every 1/60 seconds. Thus, in ordinary use, the time an event occurred between two consecutive frames makes no difference to the users, i.e. any delay of an event to match the time of the next frame is *de facto* impossible to perceive for an user, and thus assumed

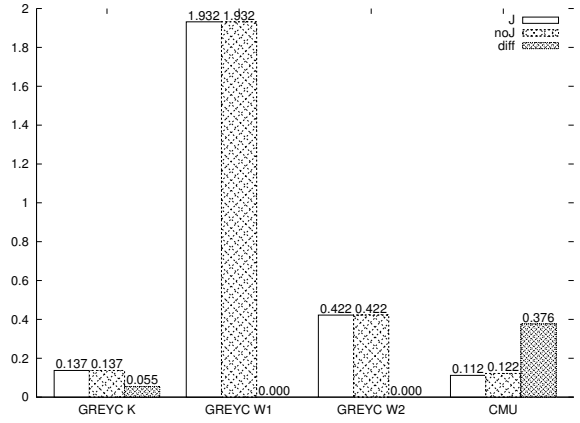

Figure 1: Maximal absolute *gains* on the EER using 1,001 different discretizations with (J) or without jitter (noJ), and maximal difference between EER with and without jitter (diff). EER values are expressed in %.

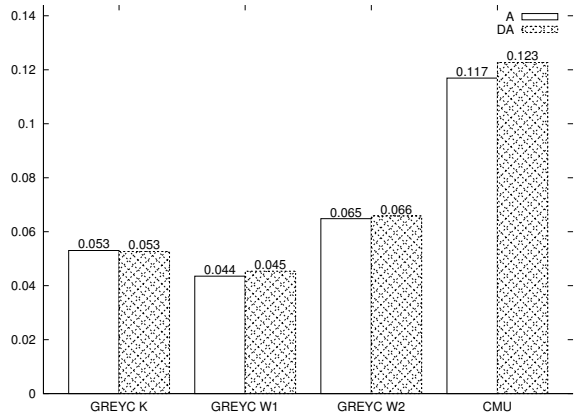

Figure 2: Minimal absolute *loss* on the EER using 1,001 different discretizations with automatic release (A) and delay then automatic release (DA).

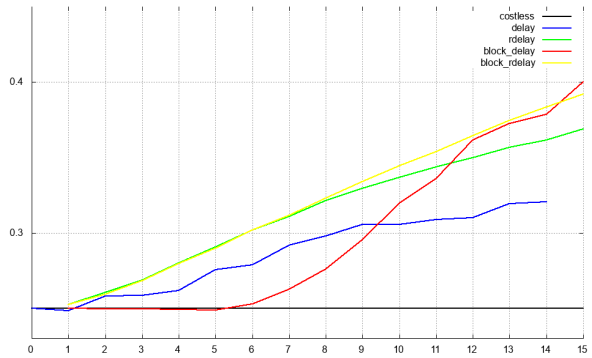

Figure 3: Minimal EER with 5 KDAS in function of their parameter N.

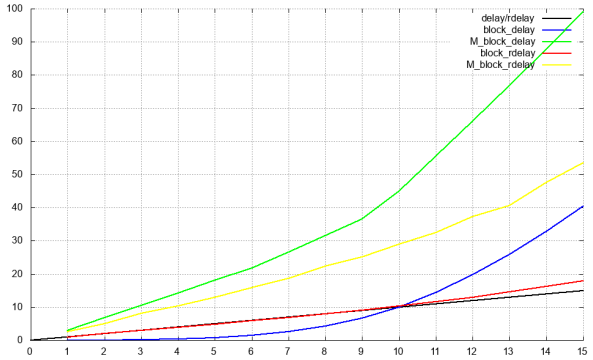

Figure 4: Mean, and maximum (prefixed with M_), of the expected latency for 5 KDAS in function of their parameter N

costless. Such operation can be trivially done thanks to `Window.requestAnimationFrame()`.

As shown in Figure 2, automatically generating release events after delaying pressure events to the next frame (DA), gives slight increase of the EER. However, such strategy is interesting as it would suppress information that could be exploited by other KDS.

In the following, non-costless KDAS, pressure events will be delayed beforehand, and release events will be automatically generated afterward.

3.2 Non-blocking protection

In order to further increase the EER values, some events have to be delayed beyond the next frame. Such delay might be perceivable by the users and thus constitute a cost in terms of usability of the KDAS. This cost, we call latency, is computed as the maximal number of frames skipped during a typing of a given text.

Non-blocking KDAS delays pressure events independently from the previous, with the only constraint to preserve the events' order. Their parameters N is the number of frames that can be skipped, and *de facto* their latency.

Two non-blocking KDAS are studied. In the first, events are discretized with a resolution of $(N+1)/60$ (delay), and in the second, events are delayed by n frames with n an uniform discrete noise $n{\sim}U(0,N)$ (rdelay). These two KDAS were tested with 15 configurations, $N \in [\![0,14]\!]$ for delay, and $N \in [\![1,15]\!]$ for rdelay. As shown in Figure fig:cost, both provide significant protection compared to the costless KDAS, however, for the same latency, rdelay seems always better than delay.

3.3 Blocking protection

In order to continue to increase the EER, events can be delayed depending on the previous event. The first blocking KDAS studied ensures that there is at least N frames between each pressure events (block_delay), the second (block_rdelay) delays them

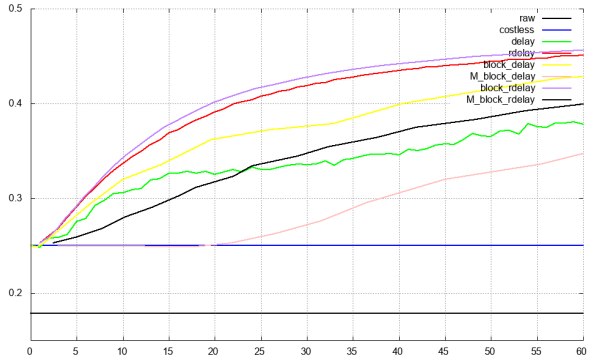

Figure 5: EER in function of latency

such as the i^{th} pressure event's delayed timestamp (t'_i) is computed from the original timestamp t_i as follows: $t'_i = max(t'_{i-1}, t_i) + U(0, N)$.

As shown in Figure 3, both blocking KDAS increase the EER faster than non-blocking KDAS. However, as shown in Figure 4, their latency quickly explodes. Thus, in order to compare fairly the KDAS between them, Figure 5 gives the EER in function of the latency.

As shown in Figure 5, blocking KDAS are, in mean, a little better than non-blocking KDAS, however, using maximal latency, non-blocking KDAS out-performs by far blocking KDAS. Moreover, if users type too fast (or N too high), blocking KDAS's latency adds up at each key pressed. When this happens, t'_i will only be computed from t'_{i-1}, i.e. every users will have the same way of typing, but at the cost of a non-ergonomic latency. Adapting N to match the user typing speed would enable browser fingerprinting attacks. This suggests that blocking KDAS should be avoided in favor of non-blocking KDAS.

4 Conclusion and perspectives

KeyboardPrivacy[12] is a Google Chrome extension that implements a blocking KDAS. Timestamp of each events is computed as follow (a is user-defined for gap and dwell) :

$$t'_i = max(t'_{i-1}, t_i) + \begin{cases} U(0,a) & \text{1 time out of 2} \\ 0 & \text{1 time out of 2} \end{cases}$$

The construction of this KDAS extension seems to be ad hoc, and could be improved using the conclusion of this study:

- use passive waits instead of active waits ;
- automatically generate release events ;
- delays pressure events to the next frame ;
- use non-blocking KDAS (rdelay) to limit the latency ;
- use fixed parameters for all users to prevent fingerprinting attacks.

This work constitutes a preliminary study on the Keystroke Dynamics Anonymisation Scheme. Greater values of N should be tested for non-blocking KDAS. Other KDS could be tested, for authentication, but also, e.g. for soft-biometrics. Attacker model could also be modified to include the knowledge of non-protected users references. Other KDAS are also possible, e.g. using non-regular discretization, using non-uniform radom laws, or by merging KDAS (e.g. merging delay and rdelay).

References

[1] Peter Eckersley. How unique is your web browser? In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 1–18. Springer, 2010.

[2] R. Gaines, W. Lisowski, S. Press, and N. Shapiro. Authentication by keystroke timing: some preliminary results. Technical Report R-2567-NSF, Rand Corporation, May 1980.

[3] R. Giot, M. El Abed, and C. Rosenberger. Web-based benchmark for keystroke dynamics biometric systems: a statistical analysis. In *Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), 2012 Eighth International Conference on*, pages 11–15. IEEE, 2012.

[4] Romain Giot, Mohamad El-Abed, Baptiste Hemery, and Christophe Rosenberger. Unconstrained keystroke dynamics authentication with shared secret. *Computers & Security*, 30(6-7):427–445, September 2011.

[5] Romain Giot, Mohamad El-Abed, and Christophe Rosenberger. Greyc keystroke: a benchmark for keystroke dynamics biometric systems. In *IEEE International Conference on Biometrics: Theory, Applications and Systems (BTAS 2009)*, pages 1–6, 2009.

[6] Sylvain Hocquet, Jean-Yves Ramel, and Hubert Cardot. User classification for keystroke dynamics authentication. In *The Sixth International Conference on Biometrics (ICB2007)*, pages 531–539, 2007.

[7] K. Killourhy and R. Maxion. The effect of clock resolution on keystroke dynamics. In *Proceedings*

of the 11th international symposium on Recent Advances in Intrusion Detection, pages 331–350. Springer, 2008.

[8] Kevin S Killourhy and Roy A Maxion. Comparing anomaly detectors for keystroke dynamics. In *Proc. of the 39th Ann. Int. Conf. on Dependable Systems and Networks*, pages 125–134, 2009.

[9] H. Lee and S. Cho. Retraining a keystroke dynamics-based authenticator with impostor patterns. *Computers & Security*, 26(4):300–310, 2007.

[10] Vinnie Monaco. Public keystroke dynamics datasets, 2018.

[11] F. Monrose and A.D. Rubin. Keystroke dynamics as a biometric for authentication. *Future Generation Computer Syststems*, 16(4):351–359, 2000.

[12] Paul Moore and Per Thorsheim. Keyboard privacy plugin, 2016.

[13] Kenneth Revett, Florin Gorunescu, Marina Gorunescu, Marius Ene, Sergio de Magalhees Tenreiro, and Henrique M. Dinis Santos. A machine learning approach to keystroke dynamics based user authentication. *International Journal of Electronic Security and Digital Forensics*, 1:55–70, 2007.

[14] RJ Spillane. Keyboard apparatus for personal identification. IBM Technical Disclosure Bulletin, April 1975.

[15] D Umphress and G. Williams. Identity verification through keyboard characteristics. *Internat. J. Man Machine Studies*, 23:263–273, 1985.

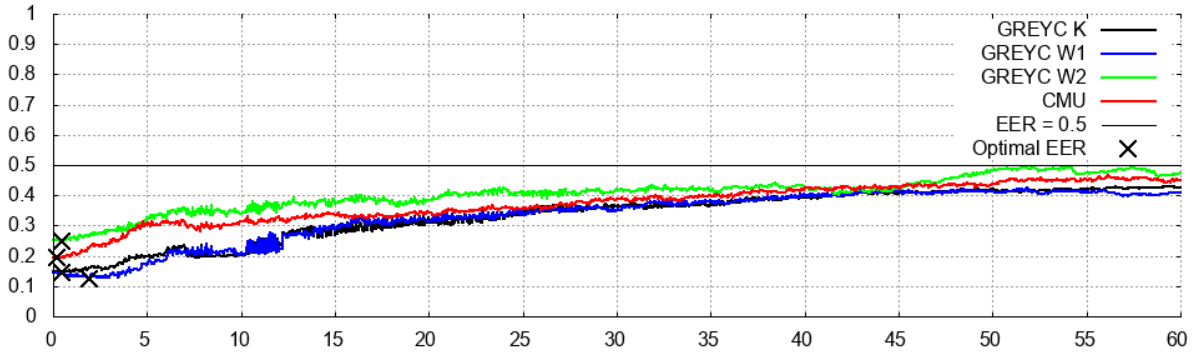

Figure 6: Attack