Spoofing key-press latencies with a generative keystroke dynamics model

John V. Monaco Md Liakat Ali Charles C. Tappert
Pace University, Pleasantville, NY 10570, USA
{jmonaco,md03901n,ctappert}@pace.edu

Abstract

This work provides strong empirical evidence for a two-state generative model of typing behavior in which the user can be in either a passive or active state. Given key-press latencies with missing key names, the model is then used to spoof the key-press latencies of a user by exploiting the scaling behavior between inter-key distance and key-press latency. Key-press latencies with missing key names can be remotely obtained over a network by observing traffic from an interactive application, such as SSH in interactive mode. The proposed generative model uses this partial information to perform a key-press-only sample-level attack on a victim's keystroke dynamics template. Results show that some users are more susceptible to this type of attack than others. For about 10% of users, the spoofed samples obtain classifier output scores of at least 50% of those obtained by authentic samples. With at least 50 observed keystrokes, the chance of success over a zero-effort attack doubles on average.

1. Introduction

Models of keystroke dynamics in the biometrics literature have predominantly been discriminative, with little attention paid to the generative mechanisms underlying typing behavior. Discriminative models are generally favored due to better asymptotic performance and efficiency [5]. Despite this, generative models have a wide range of applications and offer insight where discriminative models do not.

In this work, a generative model for keystroke dynamics is introduced and used to replicate typing behavior from eavesdropped key-press latencies with missing key names. This represents a non-zero-effort attack in contrast to the commonly used zero-effort attack to evaluate the performance of keystroke authentication systems.

Key-press times can be obtained remotely without installing a keylogger on the victim's computer by observing the network traffic generated from an interactive application, such as SSH in interactive mode [15] and Google Suggestions service [17]. A timing attack on SSH network traffic timestamps collected during password entry can provide about 1 bit of information in cracking a password. Song *et al*. verified that the key-press latencies can be reliably determined from the packet inter-arrival time of interactive SSH traffic, as the time between the actual press of a key and the creation of the packet by the kernel is negligible [15]. In [17], the key-press timestamps are masked by a buffering mechanism, but still allow for the estimation of key-press latency statistics.

With access to the victim's computer, we could use an attack such as in [12]. Observing a victim's keystrokes directly allow for typing behavior to be easily replicated. Without any knowledge of the victim's typing pattern, an attack such as [14] can be used. The latter work used an independent keystroke dynamics database to reduce the search space of typing behavior. This works well for shorter strings and assumes that the authentication system allows multiple attempts, as it relies on enumerating many possible typing behaviors.

The assumptions in this work fall between [12] and [14], and are most closely related to [17] where it was shown that a user's mean key-press latency could be determined with less than 20% error. We assume that we don't have access to the victim's computer but are able to remotely observe the network traffic generated from keystrokes in an interactive application, such as SSH in interactive mode. The authentication scenario in this work is also more stringent than [14]. We consider long-text applications and the possibility of only a single authentication attempt. This is a type of *sample-level* attack that does not require knowledge of the authentication algorithm itself. A user's typing pattern is imitated in a spoofed sample and provided to the keystroke biometric authentication system. Such a scenario is typical of those recently introduced by some Massively Open Online Course (MOOC) providers [7], where users have to

The authors would like to acknowledge the support from the National Science Foundation under Grant No. 1241585. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or the US government.

copy several sentences for the purpose of keystroke biometric verification.

Part of this work is motivated by the nature of time-stamped human events. It is well known that the inter-event times from human events generally follow heavy-tailed distributions and exhibit temporal clustering. There have been various generative mechanisms proposed that explain this phenomenon, such as a priority queue for task execution [1]. Generally speaking, it is not clear what type of distribution the inter-event times follow, and it has been disputed whether a power-law or log-normal provide better fit [16, 2]. In this work, a log-normal distribution is used to model key-press latencies, motivated by empirical observations and hypothesis testing.

Beginning in the 1980's, there has been much interest in the behavior of transcription typing and generative mechanisms underlying typing behavior. This line of research seems to be underutilized by the keystroke biometric community. Both [3] and [13] provide a treatise on typing behavior. More recently, [18] uses a general human processor model to explain some of the phenomena of transcription typing.

This work proposes a novel generative model for typing behavior. It also demonstrates how the model can be used to perform a non-zero-effort attack on a keystroke biometric system by spoofing key-press latencies. The generative model is shown to be consistent with typing behavior in long-text applications through Monte Carlo hypothesis testing. The model is then used to perform a non-zero-effort attack using key-press latencies with missing key names, which can be observed remotely. Spoofing capabilities are evaluated using an open-source algorithm with previously-published performance results. The rest of the paper is organized as follows. The experimental data is described in Section 2, the model is defined in Section 3, experimental results are provided in Section 4, and conclusions in Section 5.

2. Keystroke data

A subset of a publicly-available keystroke database[1] is analyzed in this work [9]. The database contains long-text input from a variety of users and under a variety of conditions. Users were instructed to either copy a fable or answer open-ended questions while their keystrokes were logged by a Java application. Data were collected on both desktop and laptop keyboards. This database is thought to be representative of the type of data that would be collected in an online course.

For this work, we randomly selected 129 users who answered 4 prompts. Of this dataset, each sample contained 751 ± 94 keystrokes, with 69 fable prompts and 447 essay prompts. This population was made up of 54 females and 71 males, with 53% of the users aged between 18 and 30. There were 18 left-handed users, 105 right-handed, 2 ambidextrous, and 2 unspecified. About 60% of the samples were collected on desktop keyboards, and the rest on laptops.

In this work, only the press-press latencies, or simply the key-press latencies, are utilized. This represents the data that can be captured over a network during an interactive session, such as SSH or Google instant. The i^{th} key-press latency is denoted by

$$\tau_i = t_i - t_{i-1} \tag{1}$$

where t_i is the timestamp of the i^{th} key-press event. This value is strictly greater than zero and has about a $16ms$ resolution with standard hardware on a Windows desktop [6].

3. Model

A Linguistic Buffer and Motor Control (LBMC) model for typing behavior is proposed. The model is first motivated by some empirical observations and then used to predict the empirical distribution of key-press latencies in the dataset. The model is subjected to a goodness-of-fit test through Monte Carlo hypothesis testing and then used to replicate typing behavior.

The LBMC model begins with the source text that is typed by the user. This could be text that is either copied or generated by the user in answering an open-ended question. The text consists of a sequence of words separated by the space character. Each word can be made up of any non-space characters, including punctuation and special symbols.

In the model, the process of typing long text generally requires two mechanisms. First, the words are loaded into a buffer with finite linguistic capacity. The linguistic capacity of the buffer is measured by the rarity of the words, and this capacity is fixed for any individual. The rarity of a word is its inverse frequency. The size of the buffer is the number of keys that must actually be typed, and this is taken to be the total number of characters in the buffer. Depending on the linguistic capacity, the buffer may be capable of holding several very rare words or many common words. Several long common words may have the same linguistic capacity as one or two short rare words. The resulting distribution of buffer character sizes depends heavily on the word-length frequency of the language.

After the buffer is loaded, it must be translated into a motor control program (MCP). The MCP specifies the physical actions that must occur for the buffer contents to be typed and displayed on screen. The model is summarized in Figure 1.

There are two types of delays that can occur in the LBMC model. The first type of delay occurs while the

[1] http://vmonaco.com/datasets/

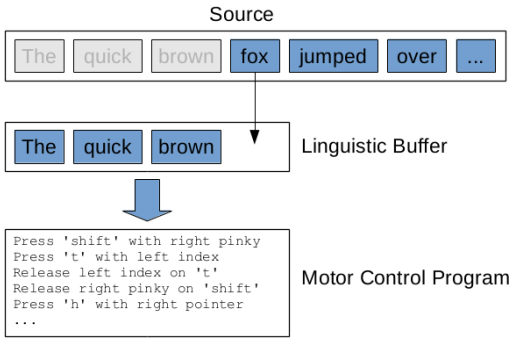


Figure 1: Linguistic Buffer and Motor Control (LBMC) model. Words are loaded into a buffer with finite linguistic capacity and then translated into a motor control program.

buffer is loaded and translated into the MCP. During this time, the user may be reading the source text or thinking to generate the source text; the user is in a passive state and relatively long key-press latencies are observed. We assume that the time to load the buffer and translate it into the MCP is proportional to the character size of the buffer and that the capacity of the buffer is limited by the rarity of the words. The second type of delay occurs during the execution of the MCP. During this time, the user is in an active state and relatively short latencies are observed. The latencies in the active state are proportional to the complexity of the MCP, as detailed below.

Many works on keystroke dynamics assume that key-press latencies are normally distributed, with [14] being an exception. In our analysis, we find that key-press latencies are right skewed, and generally follow a heavy-tailed distribution. We will show that the key-press latencies in both states can adequately be described by a log-normal distribution, given by

$$f(\tau;\mu,\sigma) = \frac{1}{\tau\sigma\sqrt{2\pi}} \exp\left(\frac{-(\ln\tau-\mu)^2}{2\sigma^2}\right) \qquad (2)$$

where μ is the log-mean and σ is the log standard deviation (SD).

3.1. Passive state

Given a source text, we can predict the distribution of key-press latencies in the passive state, within a scaling factor, by modeling the buffer character size. The linguistic size of the buffer, s_l, is a measure of the amount of rarity in the buffer. When the linguistic size reaches capacity L, the buffer is said to be full. Let $w = \frac{1}{f}$ be the linguistic size of a word, where f is the frequency of the word. The linguistic size of the buffer is given by $s_l = \sum w$. The buffer character size, s_c, is the total number of characters in the buffer. The latencies in the passive state are proportional to s_c. Using the text from the keystroke database, the buffer

Figure 2: Keyboard key locations

character sizes are simulated as follows.

1. Initialize an empty buffer with a fixed linguistic capacity
2. Add the next word to the buffer and increase the buffer's linguistic size by the inverse frequency of the next word
3. If linguistic size of the buffer is over capacity, record the character size and go back to step 1. Otherwise repeat step 2.

The distribution of buffer character sizes when $L = 0.05$ is shown in Figure 3a. The best-fit log-normal distribution is also shown and visually agrees with the predictions. We have verified that a log-normal provides a good fit for a wide range of L. The buffer character sizes also agree with a copy span of 7-40 characters that have been observed in transcription typists [18].

3.2. Active state

After the linguistic buffer is loaded and translated into a MCP, the user transitions to an active state and the MCP is executed. The key-press latencies in the active state are proportional to the complexity of movement between keys. For a touch typist, the complexity of typing two keys that are distant from each other is low. This is due to the ability to use different fingers or hands for each of the key. When two keys are close, the key-press latency will increase as a result of the immediate reuse of the finger [3]. This phenomenon has been established in transcription typists [18].

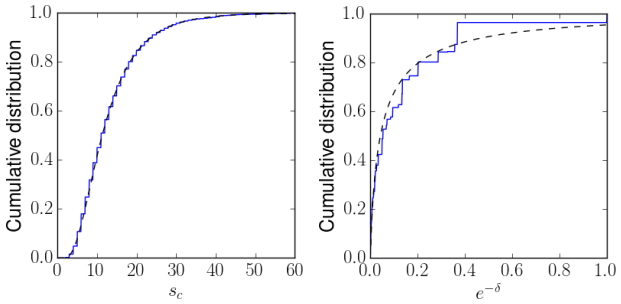
We define the key locations of a Dell USB L-100 keyboard, shown in Figure 2. Let the inter-key distances be given be as

$$\delta_i = \|k_i - k_{i-1}\| \qquad (3)$$

where k_i is the key location of the i^{th} keystroke, determined by Figure 2. We propose that the log-key-press latencies are inversely proportional to the inter-key distance, or alternatively

$$\tau_i \propto \exp -\delta_i \qquad (4)$$

This claim is backed by some research that has shown memory search and motor learning processes to be characterized



(a) Key-press latencies in the passive state.

(b) Key-press latencies in the active state.

Figure 3: Predicted key-press latencies in the passive state using buffer character size and in the active state using inter-key distance. The best fit log-normal (dashed black) is shown for both types of latency.

by the exponential function [4]. Similar to the previous section, we can predict the distribution of key-press latencies in the active state, up to a scaling factor. This can be accomplished by taking all the key distances in the source text and modeling the distribution of $e^{-\delta}$. The predicted key-press latencies are shown in Figure 3b. Again, a log-normal provides a reasonable fit.

3.3. Hidden Markov Model

The true system state is hidden from observation, as we cannot generally determine whether the user is in an active or passive state. We introduce the latent variable z_i to indicate the hidden state of the system at the i^{th} key-press, where $z_i = 0$ indicates the passive state and $z_i = 1$ indicates the active state. Let μ_j, σ_j for $j = \{0, 1\}$ indicate the log-normal emission parameters in the passive and active state. Also let a_0, a_1 be the probabilities of staying in each state and π_0,π_1 be the probabilities of beginning in each state. With the Markovian assumption, the parameters can be efficiently determined by the Baum-Welch algorithm. The model parameters consist of $\theta = \{\mu_0, \sigma_0, \mu_1, \sigma_1, \pi_0, \pi_1, a_0, a_1\}$. The Viterbi algorithm can then be used to determine which key-press latencies correspond to the passive state and active state. The proposed model is summarized in 4.

The maximum likelihood (ML) parameters for the log-normal emission have a closed form solution, given by

$$\hat{\mu}_i = \frac{\sum_{n=1}^{N} \gamma_n(j) \ln \tau_n}{\sum_{n=1}^{N} \gamma_n(j)}$$

and

$$\hat{\sigma}_i = \frac{\sum_{n=1}^{N} \gamma_n(j) (\ln \tau_n - \hat{\mu}_j)^2}{\sum_{n=1}^{N} \gamma_n(j)}$$

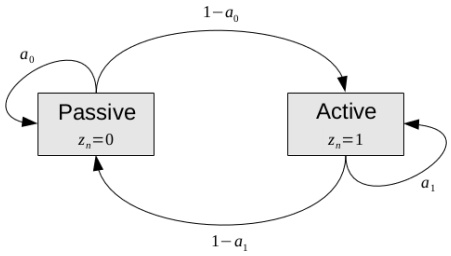


Figure 4: HMM summary

where $\gamma_n(j)$ is the posterior probability of state j, given observations up to time N.

3.4. Goodness of fit

To determine whether the proposed model is consistent with observed data, we perform a goodness-of-fit test through Monte Carlo hypothesis testing. The test proceeds as follows. For each user, we determine the ML parameters, θ_m. The area test statistic between the model and empirical distribution is determined [8], denote by A_m. The area test statistic is a compromise between the Kolmogorov-Smirnov (KS) test and Cramer-von Mises test [8], given by

$$A = \int |P_D(\tau) - P_M(\tau|\theta)| d\tau$$

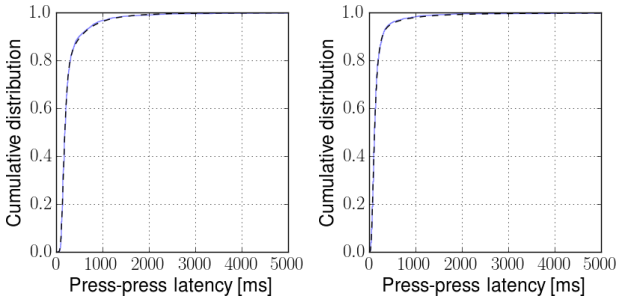
where P_D is the empirical cumulative distribution and P_M is the model cumulative distribution. The marginal density of the model is given by

$$g(\tau; \theta) = \sum_{j=0}^{1} p_j f(\tau; \mu_j, \sigma_j) \tag{5}$$

where p_j is the steady-state probability of being in state j and f is the log-normal distribution. Using the model with trained parameters θ_m, a surrogate data sample the same size as the empirical data is generated. The surrogate data is then treated similarly to the empirical data, where ML parameters θ_s are determined. The area test statistic A_s between the surrogate-data-trained model and surrogate data is computed. This process repeats until enough surrogate statistics have accumulated to reliably determine $Pr(|A_s - \langle A_s \rangle| > |A_m - \langle A_s \rangle|)$. The biased P value is given by

$$\frac{I\{|A_s - \langle A_s \rangle| > |A_m - \langle A_s \rangle|\} + 1}{m + 1} \tag{6}$$

where $I\{\cdot\}$ is the indicator function. A biased P value is used because a large number of tests are being performed and there is a possibility of increased Type I error using an unbiased estimator [11]. The model-predicted and empirical distribution of key-press latencies for two random users



(a) $P = 0.37$ (b) $P = 0.35$

Figure 5: Empirical (solid line) and predicted (dashed line) distribution of key-press latencies for two random users. The predicted distributions match the empirical distributions very closely.

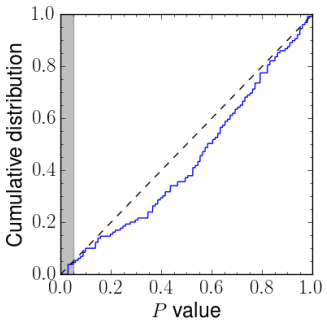


Figure 6: Distribution of P values in testing the null hypothesis that the model is consistent with the data. The shaded region indicates the 5% significance level, where the null hypothesis was rejected. If the data was actually generated by the model, the P values would follow a uniform distribution given by the dashed line.

are shown in Figure 5. The model predictions visually agree with the empirical distribution.

The null hypothesis (that the model is consistent with the data) is tested for each user in the database for a total of 129 tests. As each test requires fitting 101 models (1 empirical and 100 surrogate samples), this process is time consuming. The null hypothesis failed to be rejected for 123 out of 129 users at the 5% significance level, demonstrating that the proposed model cannot be rejected as a possible explanation of typing behavior. The distribution of P values is shown in Figure 6. The dashed line indicates a uniform distribution.

3.5. Empirical patterns

Ultimately, we would like to generate key-press latencies that correspond to a user's typing behavior. If we are given some key-press latencies with missing key names, such as those observed over a network from an interactive application, we can exploit the key-press latency scaling predicted by Equation 4. We assume that we have some predefined text that we must generate key-press latencies for. This could be, for example, a prompt that the user has to copy for identity verification in an online course.

First, the HMM defined in the previous section is used to identify the active-state latencies in the observed data. Only the active state is of interest since this is where the inter-key distance key-press latency scaling occurs. The active state is itself composed of a mixture of log-normal distributions for each unique key-distance. Let μ_δ and σ_δ be the log-mean and log-SD for key-press latencies with key-distance δ. Then, as predicted by Equation 4, we have

$$\mu_\delta = C_\mu \delta + b_\mu \qquad \sigma_\delta = C_\sigma \delta + b_\sigma \tag{7}$$

To verify this relationship and determine C_μ, b_μ, C_σ, and b_σ, we can perform a least squares linear regression for each user in the database. In doing so, we consider only the latencies between letter keys when omitting the top 10 digrams of each user, as the model does not account for highly practiced sequences such as "th" and "he".

The result for two users is shown in Figure 7. The fast typist is likely a touch-typist and shows strong negative correlation between inter-key distance and log-key-press latency, as predicted by Equation 4. The slow typist shows the opposite behavior, as the log-latency increases with key distance. This user is likely a hunt-and-pecker, spending more time locating keys that are far apart. The log-SD increases with key distance for both users, but more so for the slow typist.

Based on these observations, we let the relationship between key-distance and latency be a function of typing speed in the active state. That is, we want to predict $C_\mu$and C_σ from the active state parameter μ_1. We anticipate that C_μ will be negative for touch-typists and positive for hunt-and-peck users. Let

$$C_\mu = m_\mu \mu_1 + b_\mu, \qquad C_\sigma = m_\sigma \mu_1 + b_\sigma \tag{8}$$

where m_μ, b_μ, m_σ, and b_σ are hyper-parameters that can be used to predict C_μ and C_σ. We can empirically determine the hyper-parameters from C_μ and C_σ for each user in the database. In Figure 8, C_μ and C_σ are plotted against the active state parameter μ_1 for each user. A least squares linear regression is again performed to determine m_μ, b_μ, m_σ, and b_σ. In the experimental results, these values are obtained on an independent database, as this requires knowledge of the key names.

3.6. Reproducing keystrokes

Given some predefined text, we can approximate typing behavior if we know the active state parameters μ_1 and σ_1. The active state parameters can be determined from observed key-press latencies with unknown key names. The

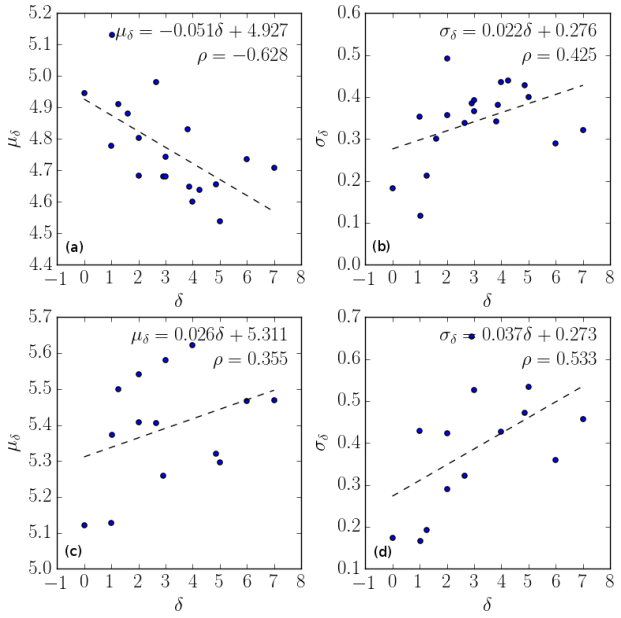


Figure 7: Inter-key distance vs. key-press latency in two users. The fast typist (a,b) shows strong negative correlation between inter-key distance and key-press latency (a). The slow typist (c,d) shows a weak positive correlation (c). Both users show positive correlation between key-distance and log-SD latency (b,d).

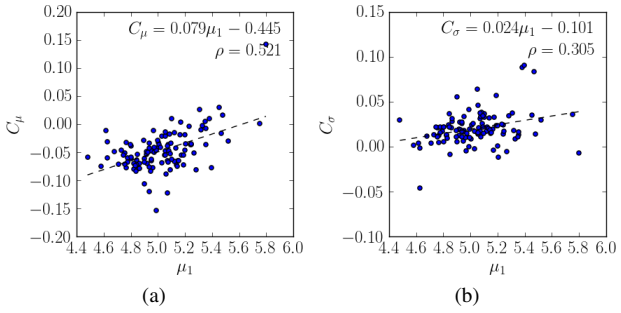


Figure 8: Hyper-parameters m_μ, b_μ, m_σ, and b_σ are used to determine $C_\mu$and C_σ. (a) shows C_μ as a function of typing speed. Typists in the upper left are fast hunt-and-peckers, upper right: slow hunt-and-peckers, lower left: fast touch-typists, and lower right: slow touch-typists. The log-mean latency of a fast typists generally decreases with increased inter-key distance. (b) shows a weak positive relationship between C_σ and typing speed.

key distance vs. log key-press latency relationship seen in Section 3.5 can then be reproduced to create seemingly-natural key-press latencies. To this effect, we scale the parameters of the log-normal distribution based on the key distances in the text we are trying to simulate typing behavior for.

Given observed key-press latencies with unknown key names, first determine the ML parameters θ for the 2-state HMM defined in Section 3.3, where μ_1 and σ_1 are the log-mean and log-SD of the active state. The scaling parameters C_μ and C_σ are then determined using Equation 8 with hyper-parameters obtained using an independent dataset that is also publicly available [10].

Let δ_i be the key distance between keys i and $i+1$ that must be pressed to reproduce the predefined text. Also let μ_δ and σ_δ be the log-mean and log-SD of latencies generated between keys with the distance δ. The following relationships hold, according to the scaling behavior defined in Equation 7,

$$\mu_{\delta_i} - \mu_{\delta_j} = \frac{C_\mu}{\delta_i - \delta_j}, \qquad \sigma_{\delta_i} - \sigma_{\delta_j} = \frac{C_\sigma}{\delta_i - \delta_j} \tag{9}$$

With n unique key distances and Equation 9, we get $2n-2$ unique equations. The reproduced latencies should also have the same first two log moments as the observed latencies in the active state. Let μ_s and σ_s be the log-mean and log-SD of all the spoofed key-press latencies. To ensure that the log-mean and log-SD of the spoofed latencies are the same as the active state, the following relationships must also hold

$$\begin{aligned} \mu_s &= \mu_1 = \sum w_\delta \mu_\delta \\ \sigma_s^2 &= \sigma_1^2 = \sum w_\delta((\mu_\delta - \mu_1)^2 + \sigma_\delta^2) \end{aligned} \tag{10}$$

where w_δ is the normalized frequency of key-distance δ in the text. Given n unique key distances in the text, together with Equations 9 and 10, we have $2n$ equations with $2n$ unknowns. This system of equations can be solved numerically for μ_δ and σ_δ. The resulting μ_δ and σ_δ can then be used to generate log-normal random variates for each key distance in the sequence. The resulting latencies will have log-mean μ_1 and log-SD σ_1, with scaling behavior C_μ and C_σ. The process for replicating key-press latencies is summarized here.

1. Using an independent dataset, determine hyper parameters $m_\mu, b_\mu, m_\sigma, b_\sigma$.
2. Observe the victim's key press timestamps with missing key names
3. Compute the maximum likelihood parameters θ for the 2-state HMM defined in 3.3
4. Determine C_μ and C_σ using Equation 8 and active state parameters μ_1 and σ_1
5. Using C_μ and C_σ, the active state parameters μ_1 and σ_1, and key-distances δ_i of a predefined text, solve Equations 9 and 10 to determine the distribution parameters μ_δ and σ_δ for each unique key distance.

6. Use μ_δ and σ_δ to generate log-normal random variates for each key distance in the text. The spoofed latencies will have log-mean μ_1 and log-SD σ_1 with key-press latency vs. key-distance scaling behavior C_μ and C_σ.

4. Experimental results

We simulate the scenario of spoofing keystrokes using data captured remotely with missing key names. Thus, only the key-press timestamps are known to the attacker. The 2-state HMM is first trained on the observed latencies and used to identify the active-state latencies. A spoofed sample is then generated to mimic the victim's typing pattern during authentication, as described in Section 3.6. The ability to replicate key-press latencies is evaluated by a well-established authentication algorithm [9]. Since the proposed model can only simulate key-press latencies, we omit the features in [9] that rely on the key-release times, such as duration and release-press latencies. Spoofing key durations is left for future work. We are left with 70 key-press latency features, which consist of the mean and SD latency of various key groups and digrams. The system uses a complex fallback hierarchy to account for missing data and an aggressive outlier removal algorithm. Source code for this classifier is publicly available[2].

The authentication and spoofing procedure is as follows. For each query sample, we use the key-press latencies with missing key names to generate several spoofed query samples of the same length. The spoofed query samples use the same text as the authentic query sample. This does not invalidate the procedure because the authentication system does not measure the linguistic style of the user in any way. Using the authentic query sample and the spoofed samples, we then obtain the authentic and non-zero-effort authentication decisions, respectively. Zero-effort authentication decisions are obtained in the usual way, by authenticating the query samples from other users. The reason for generating several spoofed query samples is to obtain the asymptotic error rates, as an unlimited number of spoofed samples can be generated and used in practice.

Results are obtained by a stratified 4-fold cross validation. In each fold, the labeled reference set is used to train the dichotomy classifier from [9] with key-press latency features only. The query set is then used to obtain the zero-effort and non-zero-effort error rates. As described above, and with 129 users with 4 samples each, a total of $66,564$ zero-effort authentications performed. This includes $129 \times 4 = 516$ mated pairs and $129 \times 128 \times 4 = 66,048$ zero-effort non-mated pairs. We generate 10 spoofed samples for each query sample, thus $129 \times 4 \times 10 = 5160$ non-zero-effort attacks are performed. The ROC curve is obtained for both the zero-effort and non-zero-effort authentications. The relative increase in equal error rate (EER) and area under curve (AUC) are reported to measure the effectiveness of the attack. Additionally, we compare the classifier output scores of the authentic samples to the spoofed samples. For each sample, let the r score be the ratio of the spoofed classifier output score to the authentic classifier output score. When $r = 1$ the sample is perfectly reproduced, and when $r = 0$ the spoofed sample is completely ineffective.

[2] https://bitbucket.org/vmonaco/dichotomy-classifier

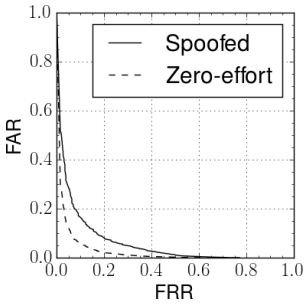


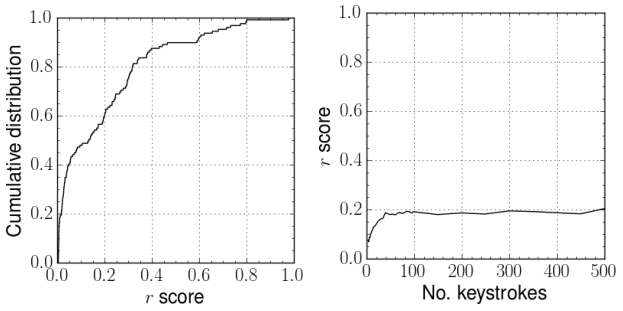
Figure 9: Zero-effort and spoofed ROC curves

We first obtain results using full query samples for observation. In this scenario, the attacker has the opportunity to observe ~ 751 keystrokes from the victim. The zero-effort and spoofed ROC curves are shown in Figure 9. The zero-effort EER is 7.5% and the spoofed EER is 12.9%, an increase of 73.4%. The zero-effort AUC is 0.028 and the spoofed AUC is 0.061, an increase of 118%. The average r score is 0.195. The distribution of r scores is shown in Figure 10a where for about 10% of users, the spoofed samples obtain classifier output scores of at least 50% of those obtained by the authentic samples.

Next, we obtain results as a function of the number of observed keystrokes. Having the opportunity to observe ~ 751 keystrokes is an unlikely scenario. More realistic is being able to observe a couple dozen keystrokes from the victim. We simulate this scenario by limiting the number of keystrokes observed from the query sample. The observed keystrokes range from 5 to 500. The relative increases in EER and AUC over zero-effort results are shown in Figure 11 and the r scores in Figure 10b. At 50 observed keystrokes, the spoofed samples obtain classifier output scores about 20% of the authentic samples and double the AUC.

5. Conclusions

This work proposed a generative model for keystroke dynamics and used that model to spoof the key-press latencies of a user after having observed latencies with missing key names. The model failed to be rejected as an explanation of typing behavior through statistical hypothesis testing. The negative correlation predicted by the model between inter-



(a) r score distribution (b) r score vs. number of observed keystrokes

Figure 10: Distribution of r scores for the full sample observation (a) and as a function of number of observed keystrokes (b).

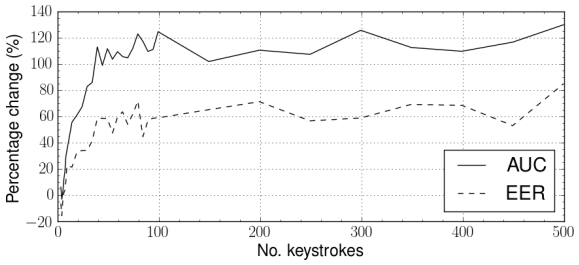


Figure 11: Relative increase in EER and AUC as a function of the number of observed keystrokes.

key distance and key-press latency was empirically verified and shown to increase with typing speed. This relationship allows key-press latencies to be spoofed by predicting the latencies that should occur in a predefined text. The proposed method was shown to require only 50 keystrokes to double the AUC, and is capable of obtaining a classifier output score 20% of an authentic sample, on average.

A model for key-release times (i.e. key durations) is left for future work, as this work dealt only with key-press latencies. The proposed model in this work may also be used to recover the key names from key-press latencies, which was the focus of [15]. Doing so may offer stronger spoofing capabilities and perhaps motivate the development of jamming devices to prevent information leakage through key-press timings.

References

[1] A.-L. Barabasi. The origin of bursts and heavy tails in human dynamics. *Nature*, 435(7039):207–211, 2005.

[2] A.-L. Barabási, K.-I. Goh, and A. Vazquez. Reply to comment on "the origin of bursts and heavy tails in human dynamics". *arXiv preprint physics/0511186*, 2005.

[3] D. R. Gentner. Keystroke timing in transcription typing. In *Cognitive aspects of skilled typewriting*, pages 95–120. Springer, 1983.

[4] A. Heathcote, S. Brown, and D. Mewhort. The power law repealed: The case for an exponential law of practice. *Psychonomic Bulletin & Review*, 7(2):185–207, 2000.

[5] A. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Advances in neural information processing systems*, 14:841, 2002.

[6] K. Killourhy and R. Maxion. The effect of clock resolution on keystroke dynamics. In *Recent Advances in Intrusion Detection*, pages 331–350. Springer, 2008.

[7] A. Maas, C. Heather, C. T. Do, R. Brandman, D. Koller, and A. Ng. Offering verified credentials in massive open online courses: Moocs and technology to advance learning and learning research (ubiquity symposium). *Ubiquity*, 2014(May):2, 2014.

[8] R. D. Malmgren, D. B. Stouffer, A. E. Motter, and L. A. Amaral. A poissonian explanation for heavy tails in e-mail communication. *Proceedings of the National Academy of Sciences*, 105(47):18153–18158, 2008.

[9] J. V. Monaco, N. Bakelman, S.-H. Cha, and C. C. Tappert. Developing a keystroke biometric system for continual authentication of computer users. In *Intelligence and Security Informatics Conference (EISIC), 2012 European*, pages 210–216. IEEE, 2012.

[10] J. V. Monaco, J. C. Stewart, S.-H. Cha, and C. C. Tappert. Behavioral biometric verification of student identity in online course assessment and authentication of authors in literary works. In *Biometrics: Theory, Applications and Systems (BTAS), 2013 IEEE Sixth International Conference on*, pages 1–8. IEEE, 2013.

[11] B. Phipson and G. K. Smyth. Permutation p-values should never be zero: calculating exact p-values when permutations are randomly drawn. *Statistical applications in genetics and molecular biology*, 9(1), 2010.

[12] K. A. Rahman, K. S. Balagani, and V. V. Phoha. Snoop-forge-replay attacks on continuous verification with keystrokes. *IEEE Transactions on Information Forensics and Security*, 8(3):528–541, 2013.

[13] T. A. Salthouse. Perceptual, cognitive, and motoric aspects of transcription typing. *Psychological bulletin*, 99(3):303, 1986.

[14] A. Serwadda and V. V. Phoha. Examining a large keystroke biometrics dataset for statistical-attack openings. *ACM Transactions on Information and System Security (TISSEC)*, 16(2):8, 2013.

[15] D. X. Song, D. Wagner, and X. Tian. Timing analysis of keystrokes and timing attacks on ssh. In *USENIX Security Symposium*, volume 2001, 2001.

[16] D. Stouffer, R. Malmgren, and L. Amaral. Comments on "the origin of bursts and heavy tails in human dynamics". *arXiv preprint physics/0510216*, 2005.

[17] C. M. Tey, P. Gupta, D. Gao, and Y. Zhang. Keystroke timing analysis of on-the-fly web apps. In *Applied Cryptography and Network Security*, pages 405–413. Springer, 2013.

[18] C. Wu and Y. Liu. Queuing network modeling of transcription typing. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 15(1):6, 2008.