keystroke dynamics

A Password Hardening Scheme

December 8, 2015

table of contents

introduction

intro

Scheme that uses biometrics gathered from durations of keystrokes and latencies between consecutive key presses to help further strengthen a login system

If pwd_a is the password used for the account a, $hpwd_a$ is denoted as this *hardened* password. $hpwd_a$ will only be generated if pwd_a is typed in the password field and the challenger's typing patterns are similar to the the ones displayed in previous successful logins

$hpwd_a$ **should remain stable** after each successful login, so that it can be used for longer term purposes (e.g. symmetric-key encryption)

features

Let a feature be denoted by ϕ, and let m be the number of features in pwd_a. In our work, $m = 2 * len(pwd_a) - 1$, where $len(pwd_a) = 10$. There were $len(pwd_a)$ durations and $len(pwd_a) - 1$ latencies. We did not gather such values

For each ϕ_i, let t_i be a fixed system-wide threshold parameter; also, μ_{ai} and σ_{ai} are, respectively, the mean and std. deviation of the successful login attempts for user a and feature ϕ_i. After a certain minimum number of successful logins h, if $|\mu_{ai} - t_i| > k\sigma_{ai}$, where $k \in \mathbb{R}^+$ is a parameter of the system, then ϕ_i is a *distinguishing feature* for the account at that point

If ϕ_i is a distinguishing feature, the user measures either consistently above or below the threshold

security goals

It should be at least as difficult to offline search for $hpwd_a$ as it is for pwd_a. In particular, the scheme should hopefully "protect" the password even if pwd_a was poorly chosen

In fact, the strength of the method increases with the number of distinguishing features, to the point where if there's none, then the effort to find both pieces is equivalent. The factor of improvement with each distinguishing feature is by around[1] 2^m

The scheme should obviously help prevent attacks where another party learns the user's password. It has been shown mimicking one's typing patterns is a strenuous task

[1] In reality, our implementation further improved that number with salting techniques applied to pwd_a.

scheme overview

secret sharing scheme

Shamir's secret sharing divides a secret S into n parts, $S_1, \ldots, S_n$. Knowledge of at least k S_i pieces is required to obtain S. Knowledge of fewer than k pieces grants no information whatsoever about S

Choose randomly a prime number $P > S$ and $a_1, \ldots, a_{k-1}$, $0 <= a_i < P$, and let $a_0 = S$. Build $f(x) = a_0 + a_1 x + \cdots + a_{k-1} x^{k-1}$. Obtain n points out of it — e.g., $(i, f(i) \mod q)$, $1 \leq i \leq n$ and distribute those n parts

Any subset containing k parts can be used to interpolate the polynomial[2]

[2]Remember, 2 points define a line, 3 points form a parabola etc. It takes k points to form a polynomial of degree $k - 1$.

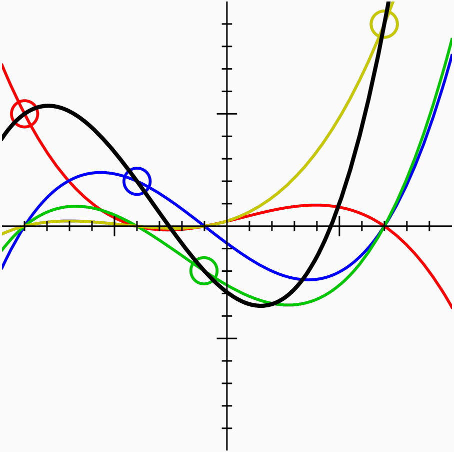

Figure 1: Multiple polynomials being interpolated by the Lagrange method.

initialisation

1. A ~ 160 bit prime number q is chosen, and should remain the same for all enrolments/login attempts
2. A random polynomial $f_a \in \mathbb{Z}_q[x]$ of degree $m-1$ is chosen such that $f_a(0) = hpwd_a$
3. Let $G(pwd_a, x)$ be a cryptographically secure pseudo-random one-way hash function. We used `sha-256`, concatenating pwd_a, x and another deterministic variable
4. An *instruction table* is generated, containing entries of the form $< i, \alpha_{ai}, \beta_{ai} >$ for each feature ϕ_i:

$$\begin{aligned} \alpha_{ai} &= y_{ai}^0 \cdot G(pwd_a, 2i) \mod q \\ \beta_{ai} &= y_{ai}^1 \cdot G(pwd_a, 2i+1) \mod q \end{aligned} \tag{1}$$

initialisation

4. All $2m$ values y^0_{ai} and y^1_{ai}, $1 \leq i \leq m$ are initially chosen such that all the points $\{(2i, y^0_{ai}), (2i+1, y^1_{ai})\}_{1 \leq i \leq m}$ lie on f_a
5. An encrypted, constant-size *history file* is created, containing measurements for the last successful h login attempts to account a. Each row in fact contains all the features, distinguishing or not, used to login on that instance. Enough redundancy is added so that it becomes easy to tell when its decryption has happened successfully. It initially has all values set to 0, and then encrypted symmetrically using $hpwd_a$ as the key. The file is padded so that its size yields no information on the amount of successful logins

log in

Let pwd^* be the password entered by the challenger. The log in proceeds as follows:

4. For each ϕ_i, pwd^* is used to "decrypt" the instruction table:

$$(x_i, y_i) = \begin{cases} (2i, \frac{\alpha_{ai}}{G(pwd_a, 2i) \mod q}) & : \text{if } \phi_i < t_i \\ (2i+1, \frac{\beta_{ai}}{G(pwd_a, 2i+1) \mod q}) & : \text{if } \phi_i \geq t_i \end{cases}$$

5. The m just-obtained points are used in a Lagrange interpolation to compute the term independent of the polynomial. $hpwd^*$ is obtained
6. The login program makes an attempt to decrypt the history file. If it fails, the login sequence ends here. Otherwise, the history table is updated with the most recent feature array. μ_{ai} and σ_{ai} are computed
7. A new f_a is generated, such that $f_a(0) = hpwd^*$

log in

Let pwd^* be the password entered by the challenger. The log in proceeds as follows:

8. For each distinguishing feature, new random values y^0_{ai} and y^1_{ai} are generated, still subject to some constraints. For non-distinguishing features, $y^0_{ai} = f_a(2i)$ and $y^1_{ai} = f_a(2i+1)$ are set
9. The instruction table is completely overridden with the one generated by the new values

implementation

Done in Kotlin

Two modes: interactive and benchmark

Interactive: user types credentials, feature array read from file

Benchmark: series of tests with data coming from a dataset — metrics on the performance and quality of results gathered

interactive mode

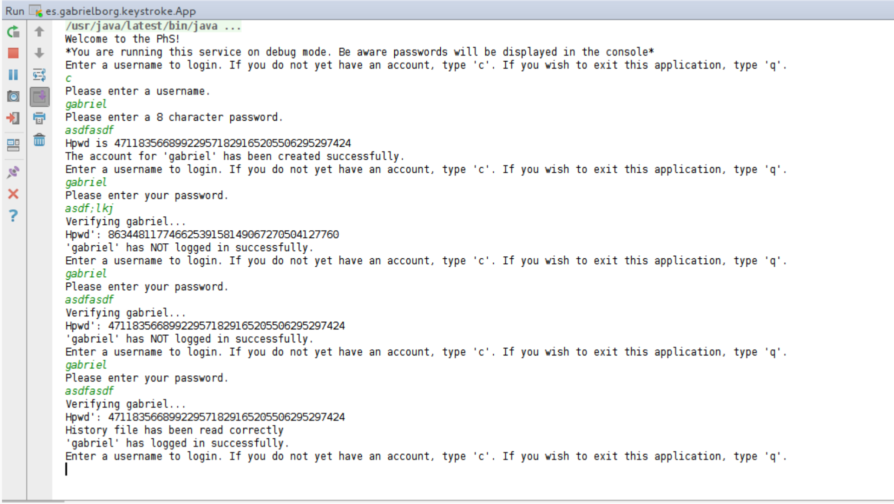

Figure 2: System running in interactive mode — enrolment and login attempts.

benchmark mode

Dataset comes from Killourhy et al.: data from 51 typists entering the password `.tie5Roanl` 400 times each

Data trimmed down to conform to our experiment: duration and ”key down-key down” latency used

Procedure for each typist:

- Enrolment + 50 authentic logins to consolidate enrolment (have system learn about features). Those logins don't show up in the results because they always go through (system doesn't have enough information yet)
- Next 700 login attempts come, alternatively: *a)* from the genuine user; *b)* from another of the 50 users (each other user contributes with 7 login attempts)

results

Dataset comes from Killourhy et al.: data from 51 typists entering the password `.tie5Roanl` 400 times each

Data trimmed down to conform to our experiment: duration and "key down-key down" latency used

Procedure for each typist:

- Enrolment + 50 authentic logins to consolidate enrolment (have system learn about features). Those logins don't show up in the results because they always go through (system doesn't have enough information yet)
- Next 700 login attempts come, alternatively: *a)* from the genuine user; *b)* from another of the 50 users (each other user contributes with 7 login attempts)

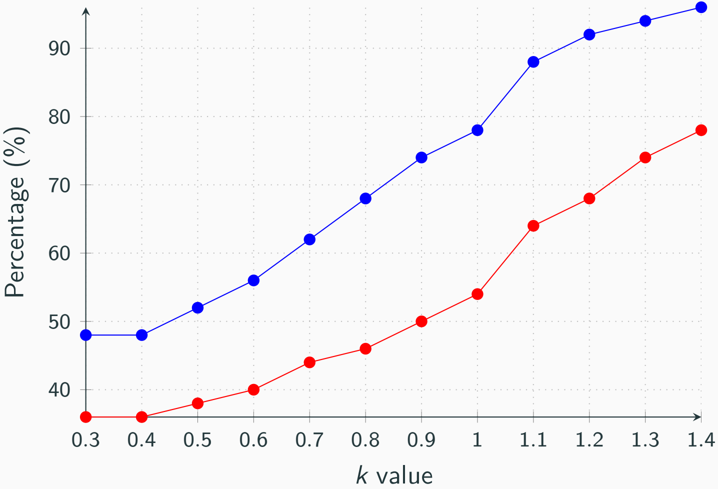

Figure 3: *Blue* curve: true positive rate. *Red* curve: false positive rate

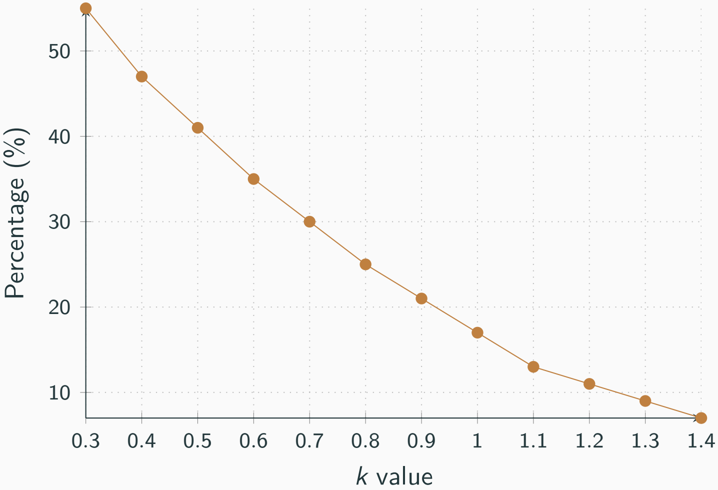

Figure 4: Percentage of distinguishing features over k value

analysis I

As a reminder, k is a fixed parameter that helps determine whether a feature is distinguishing. The higher the k, the more strict the comparison

We have chosen to plot *sensitivity* across *fall-out* in Figure 3 because that clearly illustrates the trade-off that happens when k is increased

As k increases, Figure 4 tells us distinguishing features get rarer. This means the system gradually relies less on the biometric aspect of the scheme, and more on the actual password

Both rates in Figure 3 follow k because the less the system relies on biometric data, the more alike the login attempts are — whether they're genuine or not

analysis II

The results are strikingly similar to the ones found in the paper. However, we did extend the search to a broader range of k and report results on unauthorised logins

The scheme isn't focused on completely zeroing out false positives (the security mainly comes from the entropy of the password itself). We believe a reasonable value of k for a secure system would be 0.5. More than half the users would log in at first try, and an attacker possessing the password would have a lower than 35% chance of logging in

We consider the work to have been a success: the system was completely implemented, and test results were very consistent with the expected

conclusion

Questions?

reference

Monrose, Fabian, Michael K. Reiter, and Susanne Wetzel. "Password hardening based on keystroke dynamics." *International Journal of Information Security* 1.2 (2002): 69-83. Available at `http://cs.unc.edu/~fabian/papers/acm.ccs6.pdf`.

Kevin S. Killourhy and Roy A. Maxion. "Comparing Anomaly Detectors for Keystroke Dynamics," in Proceedings of the 39th Annual International Conference on Dependable Systems and Networks (DSN-2009), pages 125-134, Estoril, Lisbon, Portugal, June 29-July 2, 2009. *IEEE Computer Society Press*, Los Alamitos, California, 2009.