AnonKeystroke

CPEN 442 Project Report

Claire Huang, Dylan Painter, Rushil Punchhi, Ying Qi Wen

Abstract

More and more websites have started to use one's keyboard typing pattern as a method of authentication and any website can collect such data from users. AnonKeystroke is a google chrome extension that renders identifying users across websites using keyboard biometrics more difficult by adding systematic delays to key down time while still allowing the website to authenticate the user. The prototype was evaluated extensively based upon functionality as well as usability. This design helps protect user's biometrics so it can be more safely and conveniently used as a form of authentication now and in the future.

Introduction

Every individual has a unique manner and rhythm when typing on a keyboard. This typing data consists of detailed timing information such as how long each key is pressed and the delays between pressing keys. This information, called keystroke biometrics, can be utilized for personal identification similar to a fingerprint. The problem arises when this identification data can be harvested by any website to track, identify, deanonymize, gain access to users credentials, or impersonate users across different websites.

The first significance of this problem is privacy. Being able to disguise keyboard biometrics helps protect the privacy of individuals and this can prevent untrustworthy websites from collecting such data, selling it to third parties, and allowing other websites to identify the user. The second significance of this problem is that keystroke biometrics were proposed as an alternative authentication method to traditional password authentication methods [1], despite functioning similarly to using the same password for all authentication purposes. Since users have one keystroke biometric that is associated with them, they are using this same keystroke biometric for the authentication of all their accounts, which can result in user impersonation or unauthorized access to a user's account. Companies such as KeyTrac [2] and TypingDNA [5] were already providing keyboard biometrics identification technology to companies and websites across the globe. One hidden danger is that if any of these companies or websites suffer from a data breach where unencrypted keystroke biometric info is stolen, then other accounts of the user which uses keystroke biometrics for authentication are at risk of being compromised. However, unlike traditional passwords, there is no easy way to change the way a user types, as it is a natural behaviour. Another danger is that since a user's keystroke information can be viewed by any website with an HTML input field, it can be easily collected by any website. Over time, this can create a detailed profile of your keystroke biometrics. Therefore, protecting user keystroke biometrics is analogous to protecting login information and

preventing identity theft.

Our design alters the typing pattern of the user. More specifically, the biometric that it changes is the key down time (time between pressing the key and releasing it). The system we have designed is a Google Chrome extension that modifies the user's key down time in real time. The top level of the system is a Google Chrome extension that runs code to determine the user's key down time, then adds an according delay to it so that the key down time sent to the website is different from that of the original key down time. The Chrome extension in the proposed design runs a machine learning model on a local server that outputs random target key down times that have the same distribution as a fake user. In order to prevent cross site tracking, the chrome extension will mimic the key down time distribution of different virtual or "anonymous" users for different websites.

Multiple tools that obfuscate the user's keyboard biometrics already existed. KeyboardPrivacy [3] and Kloak [6] are applications that obscure the keystroke bioinformatics of the user by randomizing the delays to a person's keystroke data before sending it to the webpage. A different approach was a protocol [7] between the client and carrier (network service provider) that securely authenticates the keystroke biometric between them using an encrypted messaging channel and sends the authentication result back to the application that requested authentication.

In order to test our design, we used two websites, TypingDNA and KeyTrac. These websites had free demos for their keystroke biometric authenticators, allowing us to run tests to see if our design was able to fool it. We ran 3 tests that evaluated whether our design was able to successfully obscure the original keystroke biometric, use the GAN generated keystrokes for authentication and to ensure that only the user can authenticate themselves. All tests were successful excluding mixed results for the obscuring original keystroke test, which we will discuss in detail in the Results section. Overall, we concluded that our design was able to help protect users from websites seeking to collect their biometrics or to use it to track them across websites.

Our design allows for keyboard biometrics to be used more securely, as the user's biometric data would not be leaked when using it, and in a way that is convenient and easy for the users. Compared to other solutions, AnonKeystroke can be used for authentication, automates the creation and management of different keystroke profiles for individual websites and provides a way to reset keystroke profiles in case a keystroke is compromised.

Related Work

While there are existing solutions, such as KeyboardPrivacy, Kloak and the authentication protocol that were mentioned in the introduction, none of them were as convenient and secure as AnonKeystroke. To illustrate this, we devised six criteria that our design was able to meet, that are partially or fully unmet by the existing solutions.

1. **Conceals the user's original keystroke biometric**
   As the core problem, all solutions can accomplish this. While the protocol solution provides more secure concealment of the user's original keystroke profile, AnonKeystroke trades more secure

concealment (in comparison to the authentication protocol) for usability.

2. **Allows authentication**
   AnonKeystroke manages different keystroke profiles for each website that can be used for authentication.

3. **Ease of creating and managing different keystroke profiles**
   None of the other proposed solutions are able to reliably and efficiently create different keystroke profiles that can be used for authentication. While KeyboardPrivacy supplies the option to modify some delay parameters, users must change them manually for all websites. AnonKeystroke automates the process of creating different keystroke profiles and associates them to a particular website for the user.

4. **Human-like keystroke biometrics**
   One unique aspect of AnonKeystroke is that it generates keystroke profiles that are similar to a real human's keystroke biometric. Solutions that add random delays could possibly flag the user as inauthentic, and could lead to denial-of-service. For instance, the paper "Blog or block: Detecting blog bots through behavioral biometrics" by Zi Chu et al. suggests an approach to detecting bots with the usage of keystroke biometrics [4].

5. **Resetting keystroke profiles**
   AnonKeystroke offers a way to generate a new keystroke biometric in case your old keystroke biometric was compromised.

6. **Chrome extension available**
   As Google Chrome is the most popular browser, holding approximately 65% of the browser market share worldwide [8], having AnonKeystroke take the form of a Google Chrome extension makes it more accessible to the majority of users. Meanwhile, the protocol solution requires all websites, clients and carriers to implement this protocol, which is difficult to achieve.

	AnonKeystroke	KeyboardPrivacy	Authentication Protocol
Conceals original keystroke	**Yes**	**Yes**	**Yes**
Allows authentication	**Yes**	**Yes**	**Yes**
Different keystroke profiles	**Yes**	**No**	**No**
Human-like keystrokes	**Yes**	**No**	**No**
Resetting keystroke profiles	**Yes**	**No**	**No**
Chrome extension	**Yes**	**Yes**	**No**

Table 1.0: Our design vs current solutions

Adversary Model

The potential adversaries considered were those who were able to collect another user's keystroke biometrics over the internet without the user's authorization.

Objectives: To obtain access to another user's secured account to either steal data and resources or to impersonate the user.
Methods: Capture the user's keystroke biometric from one website and use that same keystroke biometric to authenticate themselves on a different website, or to sell to another group.
Capabilities: The adversary would need the skills and knowledge to either create a webpage or the ability to create an application that can intercept and record the key press timings of a user.
Funding Level: The adversary would only need access to a computer. However, getting people to use the websites would require more funding, through scamming people, or tricking them to enter their website.
Outsider vs Insider: The adversary would be an outsider who has incentives to gain access to another user's account.

System Design

The idea behind the design was to create multiple keystroke biometrics that seem like they belong to multiple different or "anonymous users" but actually belong to the same person. Each website is associated with a different keystroke biometric, so even if one keystroke is compromised, the user's other account on other websites cannot be traced back to them, providing anonymity.

AnonKeystroke was a Google Chrome extension that was locally hosted which added a specific delay time before key presses were sent to the webpage. It serves as a tool that modifies the user's keystroke biometrics to avoid the leak of their original bioinformatics data. The Chrome extension would need an algorithm to generate target key down time, and the algorithms needed to fulfill three requirements. First, the generated target key down time should have an entirely different distribution from the user's key down time, thereby providing anonymity. Second, to prevent cross site tracking, the algorithm needed not to be the same on all websites. Third, the user would still need to be able to be authenticated by any website that uses keyboard biometrics to authenticate users, therefore the distribution of the target key down time needed to stay consistent for each website.

To do so, our team proposed to use a generative adversarial network (GAN) trained using real keystroke biometrics data to generate key down timings whose distribution mimics that of a real human user. While more powerful machine learning architectures exist, a GAN is able to accurately generate new data with the same distribution as the training set [9]. Since GANs are mostly used to generate images, it is more than powerful enough to generate an array of key press timings. This array of key press delay timings will be referred to as keystroke profiles.

Upon receiving a user's key press, the signal data would go through the AnonKeystroke extension, where the key up and key down timings which control dwell time were modified to add one of the stored GAN generated keystroke profiles on top of the original keystroke biometrics. These modified key presses were

then sent to the webpage that the user is interacting with. Since AnonKeystroke's modification of key presses is completed locally, it is not feasible for an adversary to obtain access to the original keystroke biometric through the webpage.

To associate a particular keystroke profile to a specific website, Chrome's Storage API was used to store key-value pairs of the website's hostname and the GAN generated keystroke profiles, which were synced to the user's Google account. Although local storage was a more secure alternative, the tradeoff between only being able to authenticate on one device and security was too large. As a prominent company, Google was trustworthy and had stated that "synced data is always protected by encryption when it's in transit" [10], thus AnonKeystroke relied on the security of the user's Google account to store the GAN keystroke timings.

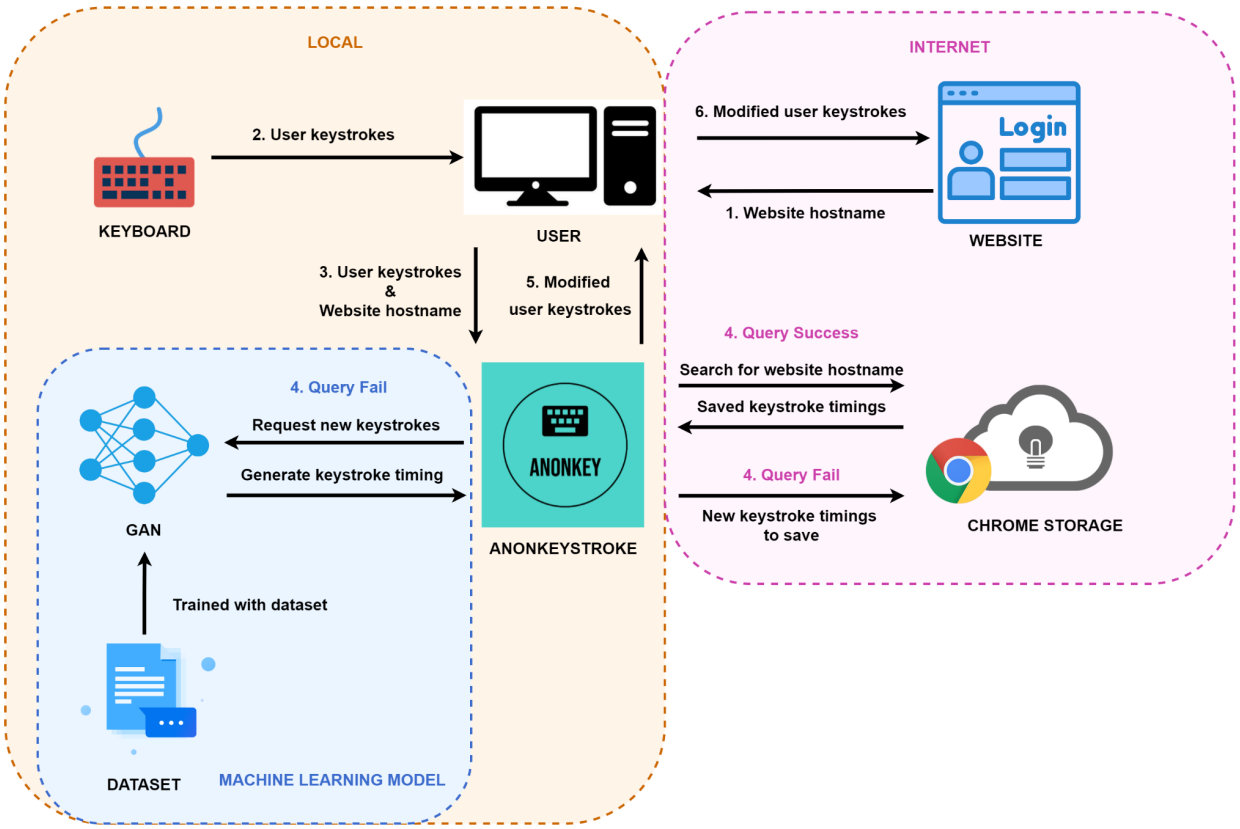

Figure 1.0: Overall architecture of the system design for AnonKeystroke

Security Principles

The following principles of designing secure systems have been employed.

User Buy In & Least Surprise: AnonKeystroke used a simple UI for convenient and fast modification of user keystroke biometrics. The UI consisted of a switch button that was on by default, but users could click to turn the extension off or on, which made the interface simple to understand and use. According to users outlined in section 6.2 (Evaluation Results), users did not notice much difference while using AnonKeystroke compared to regular typing. These factors ensured that users would not bypass our application.

Sufficient Work Factor: Although there was some tolerance for differing keystroke timings due to the nature of variances in human typing, adversaries would still need to find a range of specific keystroke biometrics that would authenticate them. Each key timing was chosen randomly from a continuum of possible values and there were currently many keys that were being used, leading to a massive number of possible combinations for the adversary to test.

Open Design: Our design did not rely on secret designs, attacker ignorance or security by obscurity. Our GAN took an input that is randomly generated to create a new keystroke biometric. Since it was random and local, adversaries would not be able to obtain this newly generated keystroke. While the storage was located online, the Chrome Storage Sync guarantees that "synced data is always protected by encryption when it's in transit" [10]. In addition, even if an adversary managed to obtain a keystroke for a website, they would still need the original user's keystroke biometrics. Thus, our components relied on randomness, encryption and the necessity of the user's biometrics to protect against adversaries, rather than a secret design.

Prototype Design

Our prototype is located at: https://github.com/RushP7/AnonKey.

To create a GAN, we researched several keystroke biometric datasets but decided upon using the Dataset for Keystroke Dynamics and Mouse Movements, which the Center for Unified Biometrics and Sensors at the University of Buffalo [15], which was provided to our team upon our request. The dataset provided a more complete collection of keystroke timings for all common keys compared to other options. This dataset contained the key up time and key down time for 148 users while they performed the tasks of answering written questions and transcribing a script. To collect the dwell time for the key presses, we ran a script that computed "key up - key down" for every letter pair of timings. To ensure the accuracy of the neural network, only the data of the keys that were pressed sufficiently many times in the data set were collected. This includes all letter keys and the space key. After we passed the script, we obtained 27 arrays, one for each key, such that each array contained the key down time of each time that key was pressed.

Having obtained the training data, they were then reformed into 1024 tensors of length 27, where each entry of a tensor represented one key down time for one specific key. The distribution of the length 27 tensors in the 27-dimensional space represents the distribution of the key down time of one specific user in the data set. The GAN was trained to, upon each call, produce a randomly generated tensor of length 27 such that the distribution of the randomly generated tensor mimicked this distribution. The final output of the GAN is an array of length 27 which contains a delay time for each key from “a” to “z‘ and the spacebar. This training process referenced a GAN tutorial [16], and the neural network was developed using the PyTorch framework before converted to a TensorFlow model that was compatible with a local server.

Our AnonKeystroke prototype referenced the source code for the KeyboardPrivacy Chrome extension [11], which collects the user’s keystroke biometrics data and alters the keystroke dwell readings before it reaches the DOM for that webpage in real time. The extension uses Jquery to read and modify DOM elements. A Jquery event listener is set up to run every time a text input element is updated. It then uses its own synchronous sleep function to cause the system to sleep for the amount of time specified by the GAN generated keystroke profile for the inputted key. Only after the sleep period has ended is the input actually sent to the DOM.

A unique keystroke profile was generated and stored for each new website visited. This was done through Chrome Sync Storage, which was associated with the user’s Google account. The storage was set up to contain the key:value pairs for each website’s hostname and their associated keystroke profile. Upon visiting a new website, AnonKeystroke would receive the website’s hostname and search for its matching key-value pair in the Chrome Sync Storage. If there was a match, the keystroke profile associated with that key was used. If the hostname did not match, AnonKeystroke would query the GAN model to generate a new keystroke profile and would save them as a key-value pair on Chrome Sync Storage.

The “Regenerate” button in the extension popup UI was used to create a new keystroke profile for the website the user is currently on. Essentially, the storage data for that website was cleared and a new keystroke profile was created, saved, and loaded for that website.

Evaluation Methodology

To verify that our solution successfully identifies the user’s keystroke biometrics as an “anonymous user” instead of the original user, we ran tests using two different authentication biometrics demos from the websites KeyTrac [12] and TypingDNA [13]. TypingDNA provides products that are used by well known brands such as Microsoft and iOS [14], which makes it a suitable testing site as it uses technology that is being used in industry. KeyTrac evaluates the user’s keystroke biometrics through a series of typed sentences while TypingDNA requires an email and password along with keystroke biometrics. Both websites have an enrollment and authentication phase, which allows us to input a new keystroke to test and test its authentication with another keystroke respectively.

To test if the demos and KeyboardPrivacy work on these two websites as expected, we ran some base tests (located in Appendix). All base tests succeeded except for KeyboardPrivacy failing to differentiate between the original user’s and KeyboardPrivacy keystrokes while using TypingDNA. We then ran 3 tests

on both websites that evaluated whether our design was able to successfully obscure the original keystroke biometric, use the GAN generated keystrokes for authentication and to ensure that only the user can authenticate themselves.

Test ID	User	Phase	AnonKeystroke	Biometric
Test 1	Same	Enrollment	Disabled	Original
		Authentication	Enabled	GAN generated
Test 2	Same	Enrollment	Enabled	GAN generated
		Authentication	Enabled	GAN generated
Test 3	Different	Enrollment	Enabled	GAN generated
		Authentication	Enabled	GAN generated

Table 2.0: Table of test parameters

One key point that we were concerned about was the speed of the extension, corresponding to whether the user would be able to feel a difference in typing and its response when using the extension. For example, if it felt like the visual response on the website is lagging behind the input of the keyboard. After concluding the tests above, we asked the users to answer two questions using a Likert scale. The first question was “How do you feel about using the AnonKeystroke extension?” where 1 represents “I didn’t like it” and 5 represents “I like it”. The second question was “How noticeable was the typing delay?” where 1 represents “Very noticeable” and 5 represents “Didn’t notice it”.

Evaluation Results

Test ID	Evaluation Results
Test 1	Consistently obtained close to 0-10% match rate on KeyTrac. On TypingDNA, authentication succeeded (match rate is $> 90\%$).
Test 2	Consistently obtained close to 90-100% match rate on KeyTrac. On TypingDNA, authentication always succeeded (match rate is $> 90\%$).
Test 3	Consistently obtained close to 0-10% match rate on KeyTrac. On TypingDNA, authentication always failed.
Satisfaction	All users gave a 4 or 5.
Typing Delay	All users gave a 4 or 5.

Table 3.0: Table of test results

Discussion of Evaluation Results

Test ID	Expected Results
Test 1	This test proves that our extension can generate a keystroke biometric that cannot be traced back to the user successfully. Thus, the user should not be verified. While the results were as expected for KeyTrac, TypingDNA should not have succeeded. However, this was also a problem that we noticed for KeyboardPrivacy, so it may be possible that the specific implementation of TypingDNA does not work well with our extension. One hypothesis was that unlike KeyTrac, TypingDNA does not refresh the page upon turning the extension on or off, which does not refresh the keystroke profiles.
Test 2	The result is that the user should be successfully verified. This test proves that our extension can successfully generate a keystroke biometric that can be used to authenticate a user successfully, without using the user's original biometric keystroke. This implies that our extension generated keystroke biometric can be used to replace the user's original biometric keystroke for certain authentication services.
Test 3	The result is that the user should not be verified. This test proves that even if an adversary somehow obtained a user's GAN generated timings, they would not be able to authenticate themselves without the user's original keystroke biometrics.
Satisfaction	Users seem to like the design that was designed to automate the entire keystroke generation and modification without much user interference. This will make it likely that they will actually use this solution, instead of bypassing it.
Typing Delay	Most users did not notice much delay. This will make it likely that they will actually use this solution, instead of bypassing it.

Table 4.0: Table of expected test results

Discussion of Results

While the results of our prototype were promising, there were several limitations with our solution. The most important was that since AnonKeystroke failed Test 1, more research had to be done to pinpoint the exact reason why both our design and KeyboardPrivacy's design both failed to obscure the user's original keystrokes on TypingDNA specifically, and what can be done to get our extension working successfully.

Some other limitations included the dataset for our GAN. Due to the specific dataset that we used not containing sufficient data to convincingly train for special characters or numbers outside of alphabetical letters and the spacebar, our extension did not fully support these keys. Currently, for these special characters we added the same delay as the spacebar. However, these special characters and numbers were often used for passwords and emails, which was why we should implement proper support for them in the

future. To improve this, we could obtain a larger dataset for the GAN. More data for the GAN would also make the GAN more accurate in mimicking human users. In order to improve the accuracy rate of GAN generated keystroke profiles, we could also incorporate additional keystroke biometrics such as error rate and gap time into our GAN.

We would also increase the security of the stored key-value pairs of website hostname and GAN keystroke profiles by encrypting them so we do not have to rely on the security of Chrome Storage Sync.

Despite the limitations, AnonKeystroke offers several advantages such concealing the original keystroke biometric, allowing authentication to occur, automates the creation and management of different keystroke profiles, generated human-like keystrokes biometrics, provides the option to reset keystroke profiles and is easily usable as a Google Chrome extension. These were also the 6 core principles behind our design. Our design idea of disguising keystroke biometrics by pretending to be an “anonymous” user allows our design to protect users against adversaries that might attempt to steal the user’s keystroke biometrics.

Conclusion

Through our prototype and evaluations, AnonKeystroke proved itself to be a promising tool that could help the user obscure their original keystroke biometrics in a way that is convenient for the users. Compared to other solutions, Anonkeystroke’s advantages were apparent: it could be used for authentication and minimized the impact to usability through automating the creation and management of different keystroke profiles for individual websites.

While more research needs to be done before this Google Chrome extension can be released for use, such as pinpointing why it worked for certain implementations of keystroke authentication websites and collecting more data for the GAN, the design idea of disguising keystroke biometrics by pretending to be an “anonymous” user is a promising one.

References

[1] Robert Moskovitch, Clint Feher, Arik Messerman, Niklas Kirschnick, Tarik Mustafić, Ahmet Camtepe, Bernhard Löhlein, Ulrich Heister, Sebastian Möller, Lior Rokach and Yuval Elovici. 2009. Identity theft, computers and behavioral biometrics. 2009 IEEE International Conference on Intelligence and Security Informatics (Jun. 2009), 155-160. DOI: 10.1109/ISI.2009.5137288

[2] GmbH, T.M.S. (no date) Keyboard biometrics, KeyTrac. Available at: https://www.keytrac.net/ (Accessed: October 11, 2022).

[3] Paul Moore (2015) Behavioral profiling: The password you can't change., Paul Moore. Paul Moore. Available at: https://paul.reviews/behavioral-profiling-the-password-you-cant-change/ (Accessed: October 11, 2022).

[4] Chu, Z. et al. (2012) Blog or block: Detecting blog bots through behavioral biometrics, Computer Networks. Elsevier. Available at: https://www.sciencedirect.com/science/article/pii/S1389128612003593 (Accessed: October 11, 2022).

[5] TypingDNA (no date) *Built with TypingDNA*, *TypingDNA*. Available at: https://www.typingdna.com/built-with-typingdna (Accessed: December 13, 2022).

[6] Vmonaco (no date) *Vmonaco/Kloak: Keystroke-level online anonymization kernel: Obfuscates typing behavior at the device level.*, *GitHub*. Available at: https://github.com/vmonaco/kloak (Accessed: December 13, 2022).

[7] Halunen, K. and Vallivaara, V. (2016) "Secure, usable and privacy-friendly user authentication from Keystroke Dynamics," *Secure IT Systems*, pp. 256–268. Available at: https://doi.org/10.1007/978-3-319-47560-8_16.

[8] *Browser, OS, search engine including mobile usage share* (no date) *StatCounter Global Stats*. Available at: https://gs.statcounter.com/ (Accessed: December 13, 2022).

[9] *Background: What is a generative model? | machine learning | google developers* (no date) *Google*. Google. Available at: https://developers.google.com/machine-learning/gan/generative (Accessed: December 13, 2022).

[10] *Get your bookmarks, passwords & more on all your devices - computer* (no date) *Google Chrome Help*. Google. Available at: https://support.google.com/chrome/answer/165139?visit_id=638060644351685092-4095070217&rd=1#zippy=%2Cchange-the-google-account-where-you-save-info (Accessed: December 13, 2022).

[11] *Keyboard Privacy* (no date) *Google*. Google. Available at: https://chrome.google.com/webstore/detail/keyboard-privacy/aoeboeflhhnobfjkafamelopfeojdohk (Accessed: December 13, 2022).

[12] GmbH, W. (no date) *Live test our biometrics solution*, *KeyTrac*. Available at: https://www.keytrac.net/en/tryout (Accessed: December 13, 2022).

[13] *Typing biometrics authentication API* (no date) *TypingDNA*. Available at: https://www.typingdna.com/authentication-api.html#demo (Accessed: December 13, 2022).

[14] *Built with TypingDNA* (no date) *TypingDNA*. Available at: https://www.typingdna.com/built-with-typingdna (Accessed: December 13, 2022).

[15] *Datasets* (2022) *Center for Unified Biometrics and Sensors - University at Buffalo*. Available at: https://www.buffalo.edu/cubs/research/datasets.html (Accessed: December 13, 2022).

[16] Raevskiy, M. (2020) *Write your first generative adversarial network model on pytorch*, *Medium*. Dev Genius. Available at: https://blog.devgenius.io/write-your-first-generative-adversarial-network-model-on-pytorch-7dc0c7c892c7 (Accessed: December 13, 2022).

Appendix

Base Tests:

Test 1 (No extension, same user): The user will input their original keystroke biometrics into the demo. For the authentication step, the same user will use their original keystroke biometrics. The result is that the user should be successfully verified. This test proves that the 2 demo websites can authenticate the same user keystroke biometric successfully.

Test 2 (No extension, different users): One user will input their original keystroke biometrics into the demo. For the authentication step, a different user will use their original keystroke biometrics. The result is that the user should not be verified. This test proves that the 2 demo websites can tell the difference between 2 different user keystroke biometrics successfully.

Test 3 (KeyboardPrivacy, same user): Our user will input their original keystroke biometrics into the demo with KeyboardPrivacy activated. For the authentication step, the same user will use their original keystroke biometrics with KeyboardPrivacy activated. The result is that the user should not be verified. This test proves that the 2 demo websites can tell the difference between randomized keystroke biometrics successfully.