12-2018

Typing-Proof: Usable, secure and low-cost two-factor authentication based on keystroke timings

Ximming LIU
Singapore Management University, xmliu.2015@phdis.smu.edu.sg

Yingjiu LI
Singapore Management University, yjli@smu.edu.sg

Robert H. DENG
Singapore Management University, robertdeng@smu.edu.sg

Typing-Proof: Usable, Secure and Low-Cost Two-Factor Authentication Based on Keystroke Timings

Ximing Liu, Yingjiu Li, Robert H. Deng
School of Information Systems, Singapore Management University, Singapore
{xmliu.2015,yjli,robertdeng}@smu.edu.sg

ABSTRACT

Two-factor authentication (2FA) systems provide another layer of protection to users' accounts beyond password. Traditional hardware token based 2FA and software token based 2FA are not burdenless to users since they require users to read, remember, and type a one-time code in the process, and incur high costs in deployments or operations. Recent 2FA mechanisms such as Sound-Proof, reduce or eliminate users' interactions for the proof of the second factor; however, they are not designed to be used in certain settings (e.g., quiet environments or PCs without built-in microphones), and they are not secure in the presence of certain attacks (e.g., sound-danger attack and co-located attack).

To address these problems, we propose Typing-Proof, a usable, secure and low-cost two-factor authentication mechanism. Typing-Proof is similar to software token based 2FA in a sense that it uses password as the first factor and uses a registered phone to prove the second factor. During the second-factor authentication procedure, it requires a user to type any random code on a login computer and authenticates the user by comparing the keystroke timing sequence of the random code recorded by the login computer with the sounds of typing random code recorded by the user's registered phone. Typing-Proof can be reliably used in any settings and requires zero user-phone interaction in the most cases. It is practically secure and immune to the existing attacks to recent 2FA mechanisms. In addition, Typing-Proof enables significant cost savings for both service providers and users.

ACM Reference Format:
Ximing Liu, Yingjiu Li, Robert H. Deng. 2018. Typing-Proof: Usable, Secure and Low-Cost Two-Factor Authentication Based on Keystroke Timings. In *2018 Annual Computer Security Applications Conference (ACSAC '18), December 3–7, 2018, San Juan, PR, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3274694.3274699

1 INTRODUCTION

Two-factor authentication (2FA) systems are pervasively used for protecting login attempts and online transac-tions. They require users to provide two separate pieces of credentials for user authentication. The first factor (credential) is typically a knowledge factor, where passwords or PINs serve as something that only legitimate users should know. The second factor (credential) is typically a possession factor, where hardware tokens (e.g., ID cards, USB tokens, and wireless tags) or software tokens (e.g., smart-phones or smart-watches) serve as something that only legitimate users should possess.

Hardware token based 2FA introduces extra burden to users since they typically require a user to carry and interact with a hardware token. A one-time code displayed on the hardware token should be submitted by the user to a server for user authentication. Besides its usability issue, a service provider must manufacture a number of hardware tokens and distribute them to all customers, which is expensive (e.g., $60 per token [2]) if the customer base is large. A hardware token usually has a lifetime around 3 years; therefore a service provider needs to distribute new tokens to each customer every 3 years, which also adds to the costs.

In recent years, due to the pervasive use of phones, SMS-based 2FA becomes more popular. After a user's first factor is verified by a service provider, a verification code is sent to the user via SMS. The user needs to use this code to prove the possession of the second authentication factor. This solution relaxes the requirement on additional hardware but still requires users to interact with their phones so as to read and input verification codes during authentication processes. In addition, a service provider bears a significant cost for sending verification codes via SMS to users' phones to complete all authentication sessions in daily operations.

To eliminate the user-phone interactions, Karapanos et al. proposed Sound-Proof [23] recently which enables a server to verify a user's second factor by matching two pieces of ambient sounds recorded respectively by the user's phone and by the browser in a login computer during a short period of time (5 seconds in [23]) right after the server verifies the user's first factor (i.e., username and password submitted to the server via the browser in the login computer) for each authentication session. However, it has usability limitations such that it is not designed to be used in quiet environments and it cannot work when the login computer has not been equipped with a microphone or the browser in the login computer does not support audio recording. In addition, this solution is vulnerable to certain practical attacks, including sound-danger attack [43] and co-located attack [23].

In this paper, we propose Typing-Proof, a usable, secure and low-cost two-factor authentication system. In Typing-Proof, the second factor is the proximity of a user's phone to the computer being used to log in to an authentication server. A user needs to place his/her registered phone near the login computer. After the user passes the first-factor authentication using a browser in the login computer, he/she is required to type a random code by the user's choice (i.e., a sequence of any keys) on the computer's keyboard. During the typing, the browser in the login computer records the keystroke

ACSAC '18, December 3–7, 2018, San Juan, PR, USA

ACM ISBN 978-1-4503-6569-7/18/12…$15.00
https://doi.org/10.1145/3274694.3274699

timing sequence (i.e., a sequence of all keystrokes' timestamps) by JavaScript and the user's phone records the keystroke sound. After finishing the typing, the keystroke timing sequence is sent to the user's phone via the server. Then the user's phone compares the keystroke timing sequence with the recorded keystroke sound, and approves the second factor if they "match", meaning that the registered phone is near the computer. If this second-factor authentication fails, Typing-Proof provides a backup solution where the user's phone displays the random code through an application. The user checks whether it matches the code typed and displayed on the browser, and presses an "Approve" or "Deny" button accordingly.

Typing-Proof is user-friendly. It requires no user-phone interactions in most cases, and one-button press in the backup case. Typing-Proof works in any environment, even in a noisy place. It can be easily deployed since it is compatible with all major browsers, login computers, and smartphones, and does not require any additional plug-ins or external hardware to be used.

Typing-Proof is practically secure. In particular, Typing-Proof is more secure than Sound-Proof since it is immune to sound-danger attack and co-located attack. In sound-danger attack, an attacker deliberately makes a victim's registered phone to produce particular sounds. However, it is difficult to simulate keystroke sound on a victim's side remotely and such simulation can be easily blocked in Typing-Proof. In co-located attack, an attacker logins to a victim's account using 2FA in the same environment with the victim. It is still difficult for the victim's phone to capture the attacker's keystroke sound in Typing-Proof, except that the distance between the attacker's login computer and the victim's phone is sufficiently short (e.g., within 100cm), which may raise the victim's awareness.

Typing-Proof incurs significant lower costs compared to other solutions, including Sound-Proof, hardware token based 2FA, and SMS-based 2FA. In particular, Typing-Proof lowers the charges of data transfer compared to Sound-Proof. Only a keystroke timing sequence and a random code (around 250 bytes) need to be sent from a user's login computer to the user's phone in Typing-Proof during the second-factor authentication; this is smaller in size than the audio signal transmitted in Sound-Proof. For hardware token based 2FA and SMS-based 2FA, they do not involve any data transfer cost, but they cost much more on manufacturing hardware tokens and sending short messages, respectively.

We have implemented a prototype of Typing-Proof for Android devices. Compared to password-only authentication mechanisms, Typing-Proof takes around 4.3 seconds longer for completing user authentication on average. This additional time is not only substantially shorter than the time overhead of 2FA mechanisms based on verification codes (roughly 10.4 seconds longer than the password-only solution) but also shorter than Sound-Proof mechanisms (around 5 seconds longer than the password-only solution). A user study we conducted shows that users prefer Typing-Proof over both SMS-based 2FA [19] and Sound-Proof [23].

2 ASSUMPTIONS AND GOALS

System Model. Our two-factor authentication system requires two devices – a computer and a phone on the user side, and an authentication server on the server side (hereinafter referred as the "server"). The computer is used to login to the user's account through a web browser application (hereinafter referred as the "login computer"). The phone is installed with a "Typing-Proof" application which is bound to the user's account (hereinafter referred as the "registered phone"). Note that the login computer and the registered phone can be the same physical device when a user logs in from the browser on his/her registered phone.

During an authentication procedure, a user points his browser to the server's webpage and enters his/her username and password. The server verifies the user's credential and challenges the user to prove the second authentication factor.

Threat Model. We assume that an adversary has obtained a victim's username and password. This assump-tion is reasonable since password database suffers from various cyber-attacks. Many companies, including Dropbox [24], LinkedIn [21], Yahoo [34], are targets of password database leakage recently. An adversary is successful in impersonating a victim user if the adversary is able to convince the server that he/she also holds the second authentication factor of the victim.

We further assume that the adversary cannot compromise the victim's registered phone. In other words, the victim's registered phone is trusted by the server. This assumption is also shared by other two-factor authentication systems based on software tokens.

We do not consider Man-In-The-Middle attack. Client-web authentication cannot fully prevent such attacks even if web applications employ HTTPS communications [22]. We leave out active phishing attack where attackers trick users to visit phishing websites and relay stolen credentials to legitimate websites in real-time. Such attacks can be defended using anti-phishing technologies [17, 36].

Design Goals of 2FA Mechanism.

- *Usability.* A 2FA mechanism should be easy to learn and efficient to use. In most cases, users should not be asked to interact with their phones. In particular, a registered phone (i.e., software token) should work well even when the phone is locked or the authentication application in it runs in its background. In addition, the second authentication factor should require no memory demand for users.
- *Security.* A 2FA mechanism should be secure under a general threat model shared by other 2FA mechanisms. It should be resilient to guessing. The second factor should be independent with the first factor. This implies that the leak of any single factor should not affect the security of the other factor.
- *Low-cost.* A 2FA mechanism should not consume too much computing resources, especially for the authentication applications installed on registered phones. The total costs using 2FA, including the costs at the server's end (e.g., SMS fee and data transfer cost) and at the user's end (e.g., data transfer cost), should be low or negligible.

3 RELATED WORK

In this section, we review two traditional 2FA mechanisms, including hardware token and SMS-based software token, as well as several recent 2FA proposals which incur less user-phone interactions. We show that why these solutions fail to satisfy our design goals.

3.1 Traditional 2FA

Hardware Token. Hardware token based 2FA is a widely deployed 2FA solution in practice (e.g., in financial industry). It requires users to carry and use hardware tokens for authentication. During an authentication session, a hardware token is used to generate an authentication code at fixed time intervals (usually 60 seconds) according to a built-in clock and a factory-encoded random key (known as "seed"). A user reads the authentication code from the hardware token and inputs it to a login computer after the user inputs the first factor.

Hardware token based 2FA requires users to interact with their hardware tokens, read and remember authentication codes temporarily before input them on login computers. It also requires a service provider to manufacture a number of hardware tokens and distribute them to all customers. The cost of tokens is considerably high (e.g., $60 per token [2]), which is usually bore by service providers. In addition, a hardware token usually has a limited lifetime of around 3 years, which implies that service providers should distribute new tokens to each customer every 3 years. In contrast, our solution does not need any additional hardware except users' smartphones and it does not require users to interact with their smartphones in most cases.

SMS-based Software Token. Due to the pervasive use of phones, SMS-based software token is becoming more popular in recent years. After a user inputs the first factor on a login computer which sends it to the corresponding server, the server sends a verification code to the user's registered phone via SMS. The user reads the verification code from the registered phone and inputs this code to the login computer to complete an authentication session. This solution does not require any additional hardware but it still requires the user to interact with his/her phone, temporarily remember a verification code, and manually inputs the code on the login computer. In this solution, the service provider bears the cost for sending verification codes via SMS to users' phones. In comparison, our solution releases users from user-phone interactions in most cases and the data transfer cost required in our solution is much cheaper than sending SMS.

3.2 2FA with Less User-Phone Interactions

Sound-Proof. Sound-Proof is a recent 2FA solution proposed to eliminate user-phone interactions and lower the cost [23]. After a user inputs the first factor, both login computer and registered phone begin to record background sounds simultaneously; then, the login computer sends the recorded audio data to the registered phone via server; a Sound-Proof application installed in the registered phone compares whether the two pieces of background sounds are similar, and determines if the login computer and the phone are located in the same environment, and thus decides whether the login attempt is legitimate or fraudulent.

Sound-Proof has a limitation that it rejects the login attempt if the average power of any recorded audio sample is below certain threshold in order to prevent an impersonation attack in the case that a victim's environment is quiet (e.g., while the victim is sleeping). This lowers its usability since it is common for a user to login to his/her accounts in a quiet place (e.g., home, office, and library). The sound introduced by user's typing would not make Sound-Proof work since the average power of keystroke sound is around 30dB as we measured while the threshold for sound recording is set to 40dB in Sound-Proof [23]. Sound-Proof suggests users make certain noise (by, e.g., clearing throat, knocking on the table) in quiet environments; however, it may be awkward for some users. It cannot work either if the login computer is not equipped with a built-in microphone since it cannot record the background sounds. We notice that most desktops are not equipped with microphones. In such cases, Sound-Proof demands additional hardware (i.e., external microphone) which may not be always convenient. Compare to Sound-Proof, our solution can be reliably used in any environment and is compatible with major browsers and PCs without requiring any external hardware.

From a security point of view, Sound-Proof is vulnerable to certain practical attacks. Zhang et al. [43] proposed a *sound-danger attack* where an attacker may deliberately make a victim's registered phone to produce previously known sounds (e.g., making a phone call or VoIP call, sending an SMS, and triggering an app-based notification) remotely at the time of an attack. Therefore, the attacker can make the same ringtone on his/her side at the same time to bypass the second-factor authentication since both ambient sounds of the victim and of the attacker are the same ringtone in such case. Another potential attack is *co-located attack* [23] where an attacker and a victim stay in the same environment (e.g., in the same café). The ambient sounds of the victim and of the attacker are obviously the same so that the attacker can bypass the second-factor authentication. Compare to this work, typing random code in our solution can be fully controlled by users, which provides adjustable security, and makes it immune to co-located attacks and sound-danger attacks.

One-Button Authentication. One-button authentication requires a user to install an application on user's smartphone and bind the application to the user's account. Whenever a login attempt occurs on a user's account, the user is notified via the application and prompted to approve or reject the request. For certain one-button authentication applications, users can approve login requests with notifications without even opening the applications. This solution makes two-factor authentication more user-friendly than SMS-based 2FA. It has been adopted by several enterprises, including Microsoft Authenticator [28], Blizzard Entertainment [9], Duo Security [14], LastPass [31], and Futurea [16].

However, most one-button authentication systems are not secure against *synchronized login attack*. If an attacker and a victim login to the victim's account at the same time, the victim cannot distinguish which login request sent to his/her registered phone is legitimate, and he/she may mis-approve the login request sent from the attacker. Although some one-button authentication applications display IP addresses of login computers along with authentication requests, it is still difficult for the users who have no knowledge about the IP addresses to distinguish which login request is legitimate. Furthermore, an attacker may forge an IP address if he/she knows the victim's IP address. In contrast, our solution enhances the existing one-button authentication systems in a special case in which it requires a user to check whether the random code displayed on his/her phone is the same as the one displayed on the login computer.

Short-Range Communication. Short-range communications, such as Bluetooth, WiFi, or NFC, are also widely adopted to support two-factor authentication.

An authentication service provider – SAASPASS [32] leverages on location-based iBeacon Bluetooth Low Energy (BLE) technology to authenticate users via Bluetooth communications between their registered phones and nearby login computers. Similarly, another 2FA proposal, PhoneAuth [13], sets up unpaired Bluetooth communications between a login computer and user's phone via Bluetooth using a new challenge-response protocol. However, these solutions may not be always applicable since most browsers (e.g., Firefox, Internet Explorer, and Safari [26]) do not support Bluetooth APIs. In addition, these solutions are not secure if adversaries set up Bluetooth connections to victims' phones to bypass 2FA.

Instead of using Bluetooth, Shirvanian et al. [33] proposed using WiFi communications between login computer and user's phone for 2FA. However, this solution works only when both devices are connected to the same network.

As NFC is widely embedded into today's commodity smartphones, Facebook [15] introduced a physical NFC security key that allows users to login to their accounts on their smartphones via NFC. This solution makes hardware token based two-factor authentication process faster. Instead of reading an authentication code from a hardware token and inputting it to a login computer, a user just taps a NFC security key against his/her smartphone so as to complete an authentication session. However, this solution requires additional hardware and its cost is of similar concern as in the case of hardware token based 2FA.

4 TYPING-PROOF

In this section, we introduce Typing-Proof in detail. Our solution uses password as the first factor and the proximity of a user's registered phone to a login computer as the second factor. The proximity of the two devices is determined by comparing the keystroke timing sequence recorded by the login computer for the user's typing of a random code on the computer with the keystroke sound recorded by the user's registered phone which is placed closed to the login computer. We analyze that our approach is usable, secure and low-cost.

4.1 Enrollment and Login

Similar to other 2FA mechanisms based on software tokens, Typing-Proof requires a Typing-Proof application to be installed on a user's smartphone as a software token and be bound to his/her account on the server. This is a one-time operation, which can be carried out using similar existing techniques to enroll software tokens (e.g., [19, 23]).

Figure 1 shows an overview of the login procedure, in which a user places his/her registered phone near a login computer and uses the login computer to login to a server. In the login procedure, a user points the browser to the URL of the server on a login computer and enters his/her username and password. If both username and password are correct, the server sends two separate indicators to the login computer and the user's registered phone, respectively for activating the second authentication process. Upon receiving an indicator, the browser pops up an input box for the user to type a random code by the user's choice. At almost the same time, the registered phone receives another indicator and starts recording audio through its embedded microphone. During the user's typing,

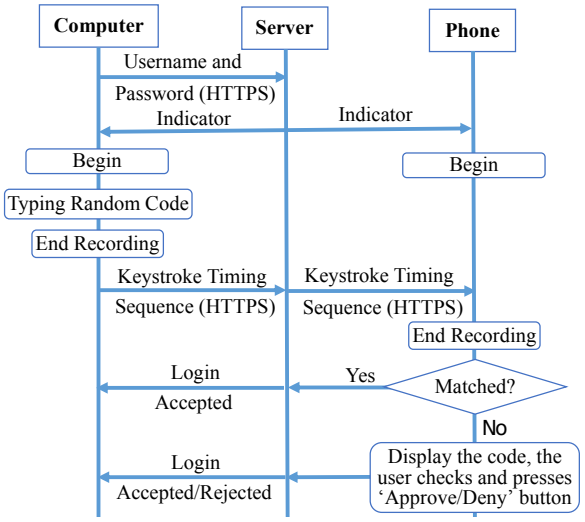
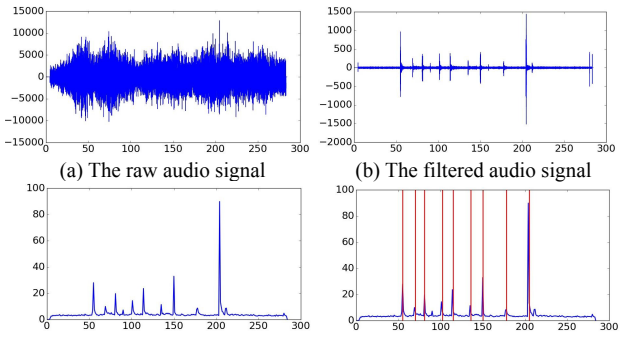


Figure 1: The overview of Typing-Proof two-factor authentication login procedure.

the browser records a timing sequence of the user's keystrokes using JavaScript. After finishing typing, the browser stops recording and sends the random code as well as the keystroke timing sequence to the registered phone through the server. When receiving the random code and the keystroke timing sequence, the registered phone stops recording and compares the keystroke timing sequence with the recorded audio signal for the second-factor authentication. In particular, it calculates a similarity score between the two. If and only if the similarity score is above a threshold τ_{sim}, the Typing-Proof application in the registered phone concludes that it is close to the login computer and informs the server that this login attempt is legitimate. Note that this second-factor authentication process is automatically carried out without any user-phone interactions.

When a user logins to his/her account using Typing-Proof in an abnormal environment, such as the keyboard is soundless, or the environment is too noisy, the automatic second-factor authentication may fail. Typing-Proof provides a backup solution where the registered phone displays the random code and 'Approve/Deny' buttons. The user presses an 'Approve' and 'Deny' button to manually accept or reject the login attempt after checking whether the random code displayed on the registered phone is the same as the one typed and displayed on the login computer. Our approach is specific to our proposed scheme and securer than the existing one-button authentication solutions described in Section 3.2 since it is immune to the synchronized login attack: the user can easily identify his/her login request by checking the random code displayed on the registered phone.

For the security reason that an attacker may infer the password from keystroke timing information [35, 42], the keystroke timing sequence is recorded from typing a random code instead of from password entry. In addition, all browser-server and phone-server communications over the Internet are transmitted via HTTPS and the server does not need to store any keystroke information (i.e., keystroke timing sequence, random code, and keystroke audio sample).



(a) The raw audio signal (b) The filtered audio signal

(c) The energy level signal (d) The energy level signal with corresponding keystroke timing vertical

Figure 2: Example of an audio signal which is recorded in a café.

4.2 Similarity Score

The Typing-Proof application on a registered phone computes a similarity score between a keystroke timing sequence and a piece of audio signal in three main steps including noise reduction, energy level extraction, and cross-correlation.

Noise Reduction. The environment where a user conducts his/her authentication may have various kinds of noise, such as other users' typing on their computers, people's talking, and background music. A user's keystroke sound using Typing-Proof may be covered by such noise. We observe that the keystroke sound mainly lies in the frequencies higher than 15,000Hz. Therefore, we utilize a high pass filter to remove the noise below. Figure 2(a) and Figure 2(b) show a raw audio sample recorded in a Starbucks café and the corresponding filtered audio sample. We have evaluated this step over a number of samples, and it turns out that keystroke signals can be correctly 'sanitized' in most cases.

Energy Level Extraction. Similar to previous study [3, 8, 38, 44], we observe that the acoustic signal of one keystroke usually involves three peaks: touch peak, hit peak, and release peak. According to our experiments, when a registered phone is placed more than 50cm away from the keyboard of a login computer or the environment is too noisy, the touch peak and the release peak may become inconspicuous while the hit peak remains clear. We thus use the hit peak to serve as a landmark of a keystroke. To highlight the hit peak, we transform the signal sequence into energy levels using time windows. Particularly, we calculate the energy levels of a keystroke sound using windowed discrete Fourier transform (DFT) and take the sum of all FFT coefficients as its energy. Figure 2(c) shows the energy-level signal corresponding to the filtered audio signal that is shown in Figure 2(b).

Cross-correlation. We choose cross-correlation as our similarity metrics. Cross-correlation is a standard measure of similarity between two time series. We use x to denote the energy-level signal converted from the audio signal recorded by a registered phone and use k to denote the keystroke timing sequence recorded by the login computer. First, we transform the keystroke timing sequence k into a pulse sequence y:

$$y[t] = \begin{cases} 0 & \text{if } t \text{ is not a element in } k \\ 1 & \text{if } t \text{ is a element in } k \end{cases} \quad (1)$$

where t ranges from 0 to the length of the energy-level signal x. Given two time series x and y, we then let:

$$CC_{x,y}(l) = \sum_{i=0}^{n-1} x[i] \cdot y[i-l] \quad (2)$$

This is a sliding dot-product of the two time series, where y is shifted by l samples over x. To accommodate different amplitudes of the two signals, the cross correlation is normalized as:

$$CC'_{x,y}(l) = \frac{CC_{x,y}(l)}{\sqrt{CC_{x,x}(0) \cdot CC_{y,y}(0)}} \quad (3)$$

where $CC_{x,x}(0)$ and $CC_{y,y}(0)$ is the auto-correlation. The cross-correlation is maximized at the offset l where the two time series are most similar. We define $max_l(CC'_{x,y})$ to be the similarity score between the keystroke timing sequence and the audio signal where l is bounded between 0 and t_{max}. Figure 2(d) shows a plot of the keystroke timing sequence and the audio signal where the two time series are matched best. The red vertical lines in the Figure 2(d) denote the timestamps of all keystrokes.

4.3 Usability Analysis

Typing-Proof requires users to place their phones near the login computer but it does not require users to interact with their phones in most cases. Users need not take any action to launch the Typing-Proof application on their registered phones before they conduct authentications. Even the Typing-Proof application is running in the background, or the registered phone is locked, the Typing-Proof application can still be activated to record keystroke sound in the second-factor authentication process as long as the registered phone is connected to a network.

The usability of Typing-Proof is slightly lower than the user authentication with password only. In most cases, Typing-Proof requires no user-phone interactions for 2FA. It takes 4.3 seconds on average without user-phone interactions, and it takes additional 7.1 seconds on average for using the backup solution in case it is triggered.

The usability of Typing-Proof is significantly higher than Sound-Proof. First, Sound-Proof is not designed to work in a quiet environment while Typing-Proof works well in various environments. Second, Sound-Proof requires a login computer to equip with a microphone for audio recording while Typing-Proof does not require any additional hardware. Third, many browsers (e.g., Internet Explorer and Safari [27]) do not support audio recording. In comparison, Typing-Proof can work with all major browsers since they all support keydown event API. On the other hand, we acknowledge that Sound-Proof is convenient to use since it does not require random typing.

The usability of Typing-Proof is also significantly higher than hardware token based 2FA and SMS-based 2FA. Typing-Proof does not require users to remember anything. In the backup solution, users need to check whether the random code displayed on the registered phone is the same as the one typed and displayed on the login computer. According to a quantitative usability analysis framework [41] shown in Appendix A, the cognitive workload of this comparison can be calculated by $0.4077 \cdot \lceil x/4 \rceil = 0.101925 \cdot x$ (seconds), where x is the length of random code. The cognitive

workload is about 1.02 seconds for $x = 10$. However, for hardware token based 2FA and SMS-based 2FA, a user needs to memorize the verification code (in most case, the length of the code is 6) temporarily and then inputs it into the browser on a login computer. The memory demand in these solutions can be calculated by $\lceil 6/4 \rceil / 29.6\% = 6.76$ (seconds) when the length of a verification code is 6 [41]. Therefore, Typing-Proof takes a shorter time for cognitive operations than hardware token based 2FA and SMS-based 2FA.

4.4 Cost Analysis

The costs for Typing-Proof stem from data transfer. During the second-factor authentication of Typing-Proof, a random code and its corresponding keystroke timing sequence (around 250 bytes) are sent from a login computer to a registered phone via server. In comparison, Sound-Proof transmits a piece of audio signal (around 250k bytes) whose data size is about 1000 times larger than that in Typing-Proof. Thus, Typing-Proof costs significant less than Sound-Proof for data transfer. For hardware token based 2FA and SMS-based 2FA, they do not involve any data transfer cost (e.g., \$0.09 per GB [4]), but they cost more on manufacturing hardware tokens (e.g., \$60 per token[1] [2]) and sending short messages (e.g., \$0.00645 per SMS [5]), respectively.

5 EVALUATION

In this section, we conduct an experiment to examine the effectiveness of Typing-Proof. We implement Typing-Proof on a prototype which consists of three components, including a web server, a web client, and a mobile client. Please refer to Appendix B for our prototype implementation, including time synchronization between web client and mobile client. We use our prototype to collect a large number of keystroke timing sequences and their corresponding audio samples. Following the similarity score calculation algorithm described in Section 4.2, we find the threshold of the similarity score that leads to the best results in terms of false rejection rate (FRR) and false acceptance rate (FAR).The performance evaluations of Typing-Proof are conducted in different settings.

5.1 Data Collection

Two volunteers recruited from our university logged in their accounts using Google Chrome, Internet Explorer, and Microsoft Edge[2] over 2 weeks. At each login, the volunteers are required to type at least 5 characters for the second-factor authentication. A login computer records keystroke timing sequence and a registered phone records audio through its microphone. This pair of data samples was stored for post-processing. The login attempts were conducted in various settings:

Environment: Different environment settings were used in our experiments, including a one-person office which is quiet, a research lab where many users sitting surrounding the user type on their own computers at the same time, and a Starbucks café with people's talking and background music. Figure 3 provides an illustration of the three environments tested in our experiment. Note that the cubicle in the research lab environment was surrounded by other 5-8 cubicles with a distance of around 1.5 meter between two cubicles next to each other.

Figure 3: An illustration of the three different environments tested in our experiment: One-person Office, Research Lab, Starbucks café.

Table 1: The number of the login attempts per volunteer for each combination of settings.

	One-person Office	Research Lab	Café
Desktop Keyboard	50S, 50M, 50L[1]	50S, 50M, 50L[1]	
Laptop Keyboard	50S, 50M, 50L[1]	50S, 50M, 50L[1]	50S[2]
Software Keyboard	50	50	50

[1] 50S, 50M, and 50L refer to collecting 50 login attempts per volunteer in the setting of short, medium, and long phone-keyboard distance, respectively.
[2] The table in the Starbucks is small so that we only consider the cases where users place registered phones 20cm away from keyboards.

Phone Position: In our experiment, the volunteers place their registered phones at various distances from the corresponding login computers, including a short distance (i.e., 20cm), a medium distance (i.e., 50cm), and a long distance (i.e., 100cm). Here, the phone-keyboard distance is measured from the center of a registered phone to the center of a login computer's keyboard.

Keyboard Model: We tested three different keyboard models for volunteers to use Typing-Proof on login computers, including a standard QWERTY keyboard (Acer PR1101U), a laptop keyboard on Mac Book Pro 13", and a software keyboard on Google Nexus 5x. In particular, the software keyboard setting refers to the scenario where a login computer and a registered phone are the same device (i.e., the registered phone). In our experiment, we used Google Keyboard-English (US) for the input, with the keypress vibration turned on and the keypress sound turned off. Therefore, for the software keyboard on a smartphone, we record the sound of keypress vibration instead of directly recording the sound of touching on the screen which is too slight to be recorded.

We collected 50 login attempts per volunteer for each combination of settings, totaling 1600 login attempts (1600 keystroke timing sequences and 1600 audio samples). Table 1 shows the structure of our dataset.

5.2 Parameters Configuration

The collected data is used to discover the best threshold τ_{sim} for comparing keystroke timing sequences with keystroke sound. The best threshold is selected according to FRR and FAR, where FRR measures the proportion of legitimate logins which are falsely rejected by the server, and FAR measures the proportion

[1] The price of hardware token may drop significantly if a large number of hardware tokens are purchased.
[2] We used Google Chrome, Internet Explorer, and Microsoft Edge since they are currently most popular browsers [30]. We also test Typing-Proof with other browsers during our user study and experience similar performance.

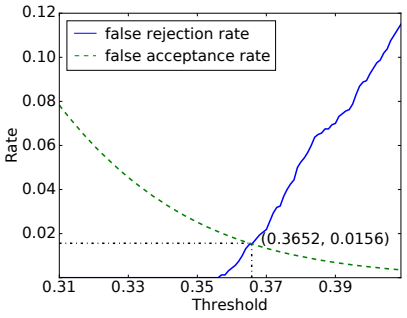


Figure 4: False rejection rate and false acceptance rate as a function of threshold τ_{sim}.

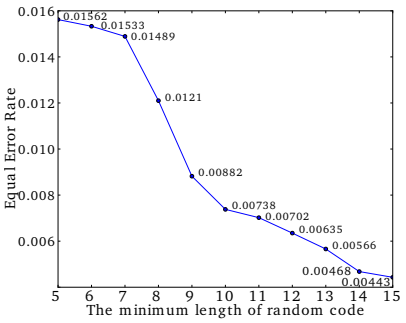


Figure 5: The relationship between ERR and the minimum length of random code.

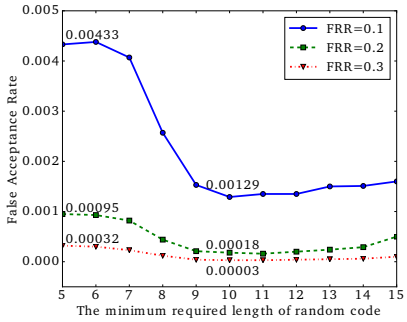


Figure 6: The relationship between FAR and the minimum length of random code when FRR is fixed.

Table 2: FRR and FAR when usability and security have different weights.

	FRR	FAR	Threshold
$\alpha = 0.1, \beta = 0.9$	0.07625	0.00603	0.394
$\alpha = 0.2, \beta = 0.8$	0.02188	0.01336	0.370
$\alpha = 0.3, \beta = 0.7$	0.00125	0.01961	0.358
$\alpha = 0.4, \beta = 0.6$	0.00125	0.01961	0.358
$\alpha = 0.5, \beta = 0.5$	0.00125	0.01961	0.358
$\alpha = 0.6, \beta = 0.4$	0.00000	0.02091	0.356
$\alpha = 0.7, \beta = 0.3$	0.00000	0.02091	0.356
$\alpha = 0.8, \beta = 0.2$	0.00000	0.02091	0.356
$\alpha = 0.9, \beta = 0.1$	0.00000	0.02091	0.356

of fraudulent logins which are falsely accepted by the server. Note that in Typing-Proof, the use of backup solution ensures that FRR is negligible assuming that users do not make any mistakes using the backup solution[3]. Therefore, we use "FRR" to denote how frequent the backup solution is activated in the following evaluations. We set t_{max} to 200ms since this is the highest clock difference experienced while testing our synchronization protocol (see Section B). Using Typing-Proof, a volunteer/user is authenticated if and only if the similarity score is greater than the threshold τ_{sim} and $l < t_{max}$, where l is the offset where the two time series are most similar.

To compute FAR, we use the following strategy. For each audio sample recorded from one of the volunteers (acting as the victim), we use all the keystroke timing sequences recorded from the other volunteer as the attacker's samples. We then switch the roles of the two volunteers and repeat the above process. Since the length of the victim's audio sample and the duration of the attacker's keystroke timing sequence are mostly different, we cut the longer sample/sequence according to the shorter one for similarity comparison. The total number of comparisons is 800*800*2=1,280,000 in our experiment.

[3]No mistake was observed in our experiments for users to compare two random codes displayed on a browser and on a registered phone, using the backup solution. The cognitive workload of comparing two random codes in Typing-Proof is significantly lower than remembering of a code displayed on a hardware token or phone and typing it on a browser [41] as it is required by hardware token based 2FA and SMS-based 2FA.

Figure 4(a) plots FRR curve and FAR curve as a function of threshold τ_{sim} when the backup solution of Typing-Proof is not used. The threshold τ_{sim} for similarity score can be determined based on the Equal Error Rate (ERR). ERR is the rate at which FRR and FAR are equal. The value of ERR is derived from the crossing point of FRR and FAR, which is 0.015625. We have $\tau_{sim} = 0.365235$ at this point.

The threshold for similarity score can also be computed when usability and security are weighted differently by the service provider. In particular, we compute the threshold that minimizes $f = \alpha \cdot FRR + \beta \cdot FAR$, for $\alpha \in [0.1, ..., 0.9]$ and $\beta = 1 - \alpha$. Table 2 provides FRR and FAR when usability and security have different weights. In Typing-Proof, we value security higher than usability since we have the backup solution which reduces FRR to almost zero. The FAR can be reduced to 0.006 if we set $\alpha = 0.1, \beta = 0.9$.

We observe that FAR is highly correlated with the length of random code. If the length of random code is longer, the keystroke patterns of different users are more diverse so that an attacker has a lower probability to bypass the second-factor authentication. Figure 5(a) shows the relationship between ERR and the minimum length l_{min} of random code. Figure 6(a) further shows the relationship between FAR and l_{min} for fixed FRRs. The FAR can be reduced significantly if users are required to type longer random codes for the second-factor authentication. For example, if it is acceptable to resort to the backup solution by 30% of chance, the FAR is 0.003% for 10-digit or longer random codes.

In the following evaluations, we provide the performance of Typing-Proof under two configurations: a general configuration and a recommended configuration. The general configuration is set according to ERR: $t_{max} = 200ms$, $l_{min} = 5$, and $\tau_{sim} = 0.365235$. In practice, we recommend service providers to value security higher than usability and require users to type at least 10 characters for the second-factor. Our recommended configuration sets $\alpha = 0.1, \beta = 0.9$, $t_{max} = 200ms$, $l_{min} = 10$, and $\tau_{sim} = 0.37$.

5.3 False Rejection Rate

We evaluate the impacts of settings, including different environments, phone positions, and keyboard models, to FRR if the backup solution is not in use. The results are shown in Figure 7. The overall FRR is 0.015625 in the general configuration and 0.01847 in the

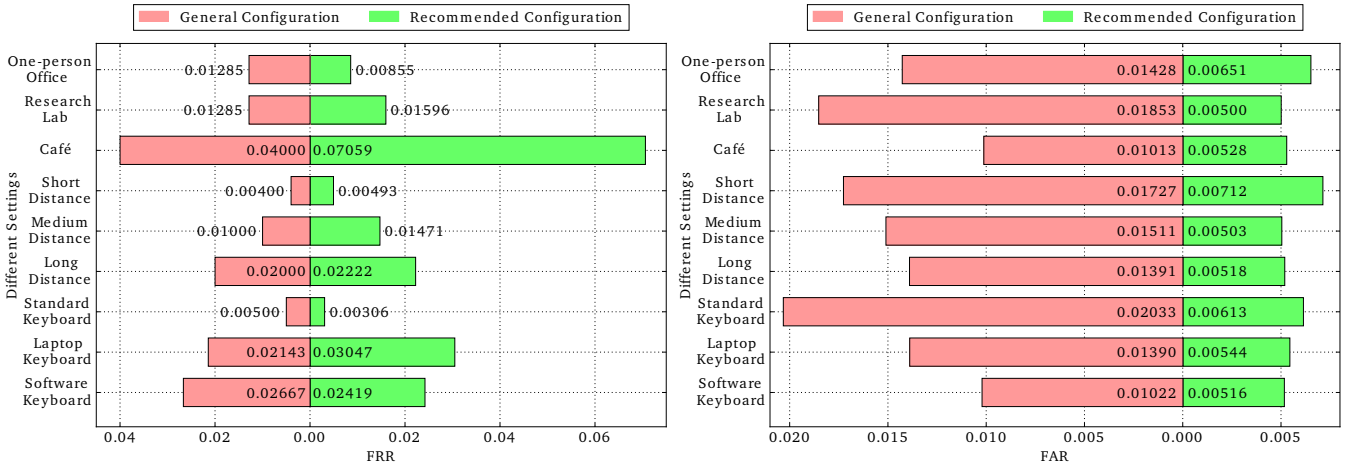


Figure 7: Impacts of different environments, phone positions, keyboard models to FRR and FAR in different configurations.

recommended configuration. This implies that the frequency of Typing-Proof resorting to the backup solution is relatively low since users do not need to interact with their registered phones in most cases (i.e., over 98%). As a comparison, the FRR due to mistyped passwords is around 0.04 [23, 25].

Typing-proof performs equally well in one-person office and research lab, which implies that the sounds of many users' typings at the same time do not affect the performance of Typing-Proof. In terms of phone positions, Typing-Proof performs best when the registered phone is placed 20cm away from the keyboard of a login computer. The performance of the long distance (i.e., 100cm) is 5 times worse than that of the short distance (i.e., 20cm). As for keyboard models, the standard QWERTY keyboard performs the best since this kind of keyboards produces loudest and clearest keystroke sound while laptop keyboard and software keyboard perform much worse. However, even in the worst case, the FRR is around 3% which is still low for resorting to the backup solution.

5.4 False Acceptance Rate

We further evaluate FAR in Figure 7. The overall FAR is 0.015625 in the general configuration and 0.00569 in the recommended configuration. In comparison, the FAR of Sound-Proof is 0.00200. It is worth noting that the FAR of Typing-Proof is measured in the worst case scenario where a victim is typing at the time of an attack. In practice, if the victim is not typing or his/her phone is put away from the victim's keyboard at the time of the attack, the attack may easily fail unless the attacker's keyboard is close enough to the victim's phone. We argue that the FAR of Typing-Proof is small enough for protecting not-so-sensitive user accounts such as those in online social networks. While for highly sensitive user accounts such as those in financial services, FAR of Typing-Proof can be further reduced (e.g., 0.003%) by setting higher FRR (e.g., 30%) as shown in Figure 6(a).

With regard to FAR in different settings, we observe that the setting which leads to lower FRR would make FAR higher. In particular, Typing-Proof has a lower probability to be attacked when a victim stays in a noisier environment, places his/her registered phone farther away from the keyboard, and uses sound-lighter keyboard.

6 SECURITY ANALYSIS

Remote Attack. A remote attack requires an attacker to obtain a victim' username and password, pass the first-factor authentication, submit a keystroke timing sequence as the second factor on a login computer. The keystroke timing sequence is then sent to a victim's registered phone for similarity comparison. It also requires the victim to type at the same time so that the keystroke timing sequence x submitted by the attacker and the keystroke sound y recorded by the victim's registered phone are highly correlated within certain time lag t_{max}, that is, $max_l(CC'_{x,y}) > \tau_{sim}$ with $l < t_{max}$.

The security of Typing-Proof against remote attack stems from the attacker's inability to know whether the victim is typing and guess what the victim is typing at the time of the attack. We bound the time lag l between 0 and t_{max} to enhance the security of Typing-Proof.

Known Typed-Text Attack. A known typed-text attack is that a remote adversary could correctly guess what a victim is typing or predict what a victim will type at some point in time and submit the same typing sequence at the same time. Note that during such attack, the victim might be typing certain meaningful text rather than random code. However, it is still difficult to bypass Typing-Proof because the keystroke patterns of typing a same code are typically different for different users [7, 29]. To prove this, we conduct an experiment to evaluate the success rate of known typed-text attacks.

In this experiment, we select 25 frequently-used 5-letter words (e.g., 'there', 'would', 'about', and etc.) and 25 frequently-used words or phrases with no less than 10 characters (e.g., 'thanks so much', 'for instance', and etc.). One volunteer (acting as the victim) is required to type these words or phrases using Typing-Proof. The audio samples of his typing are collected. Another 9 volunteers (acting as attackers) are asked to type the same words or phrases as the victim has typed. Each word or phrase is typed for 5 times per volunteer and the corresponding keystroke timing sequences are recorded. In total, $50 * 5 * 9 = 2250$ attacking cases are generated.

We calculate the similarities between an attacker's keystroke timing sequences and the victim's audio samples for typing a same word or phrase. All similarities are lower than the thresholds selected in the Section 5.2 (i.e., $\tau_{sim} = 0.365235$ for the general configuration and $\tau_{sim} = 0.37$ for the recommended configuration), which implies that all known typed-text attacks failed. It is observed that if the minimum required length of random code is longer, the similarities in the attacking cases are lower. In particular, the average similarity of attacking a 5-letter word and attacking a word or a phrase no shorter than 10 characters are 0.17750, 0.12895, respectively. This indicates that service providers may set the minimum required length of random code to 10 or more so as to provide better protection against known-typed text attacks.

Co-located Attack. In a co-located attack, an attacker logins to a victim's account and types a random code in the same environment where the victim stays. Typing-Proof can withstand such attack since it is difficult for the victim's registered phone to capture the attacker's keystroke sound unless the victim's registered phone is very close to the attacker's keyboard. We conduct an additional experiment to evaluate the success rate of such co-located attack.

In this experiment, we assume that a victim and an attacker are located in a same environment and there is a certain distance between the attacker's keyboard and the victim's registered phone. In particular, in the one-person office and research lab environments, we set the distance between the attacker's keyboard and the victim's registered phone as 150cm and 200cm, respectively. In the café environment, we let the attacker sit at the same table with the victim (phone-keyboard distance is around 50cm), and the attacker sit at the next table to the victim (phone-keyboard distance is around 100cm). We also consider the cases where the attacker and the victim typing at the same time and the cases where the victim is not typing at the time of co-located attack in each environment and for each phone position. In each test case, we run the attack 50 times and calculate the similarity between each audio sample collected by the victim's registered phone and the corresponding keystroke timing sequence generated by the attacker.

Our results show that the success rate of co-located attack is 0.00667 (4 out of 600 cases). In particular, 3 successful cases occur when the attacker sits at the same table with the victim in a Starbucks, and the rest successful case occurs when the attacker sits 150cm away from the victim in the one-person office. The victim is not typing at the time of attack in all four successful attacking cases. In order to launch a successful attack, the attacker needs to sit very close to the victim's registered phone and ensures that the victim himself/herself does not make any keystroke sound. However, this is likely to raise the victim's suspicion anyway. In suspicious cases, awared users can simply move their registered phones farther away (e.g., larger than 150cm) from the keyboards used by suspicious attackers.

Relay Attack. A relay attack is that an attacker obtains a victim' username and password, passes the first-factor authentication, records the keystroke sound of his/her typing of a random code, and plays the keystroke sound near a victim's registered phone with a speaker in real time. In a legitimate authentication, the real keystrokes are produced from different positions on a keyboard unless a user types repeated keys as random code. In a relay attack, however, the sounds played from a speaker come from the same source. Such relay attack can be detected by analyzing the time-difference-of-arrival of keystroke sound and determining whether the sound source is moving. This approach was first proposed by Zhang et al. for voice liveness detection [43]. It requires that the registered phone is equipped with two microphones, which is met by most popular smartphones (the smartphone products of Samsung, Apple, Huawei, and Xiaomi whose total market share is more than 62.32% [37] are equipped with at least two microphones). Although an attacker may deliberately choose to type repetitive "random" code like "aaaaa" in a relay attack, we recommend users not to type repetitive codes and if such codes are sent to the registered phones, Typing-Proof is switched to the backup solution.

Sound-Danger Attack. A sound-danger attack [43] is that an attacker deliberately makes a victim's registered phone to produce previously known sounds (see Section 3.2). In order to launch a successful sound-danger attack against Typing-Proof, an attacker needs to trigger the victim's registered phone to produce the keystroke sound that matches the timing sequences of the attacker's typing for 2FA.

While an attacker may use ringtone, notification tones or notification vibrations to simulate keystroke sound on the victim's phone, such attack can be detected by checking the signatures of keystroke sound which are different from the signatures of triggered tones/vibrations [38]. Another countermeasure is to temporarily disable the function of a Typing-Proof application on a registered phone whenever a notification is received. Android platform provides Class `NotificationListenerService` [20] to monitor whether any new notification is received.

Alternatively, an attacker may choose a random code, craft an audio signal which contains the keystroke sound for typing this code, hide the audio signal into a video or audio recording, and trick a victim to play the recording (e.g., through a manipulated website or YouTube video). At the same time when the victim plays the recording, the attacker submits the chosen random code to the victim's account. Since Typing-Proof uses only the frequency higher than 15000Hz from an audio sample, it is even easier to hide the high-frequency part of the keystroke sound, which is inaudible to human. However, such attacks can still be detected by analyzing the time-difference-of-arrival of keystroke sound, which is the same as the countermeasure of relay attacks.

7 USER STUDY

An IRB-approved user study was conducted to evaluate the usability of Typing-Proof and to compare it with the usability of Sound-Proof and SMS-based 2FA.

7.1 Procedure

Our user study involved 25 participants, including 16 males and 9 females with ages from 21 to 27. All participants were students or staff in a university. All participants were informed that no personal information is collected and that the survey in the user study is anonymous.

The user study took place in a classroom. Most participants used their own laptops and their own Android smartphones for 2FA logins. For the participants who did not have any Android phones, we provided our test-phones for them. All devices were connected to

the Internet through WiFi. We set up a server on a desktop, Acer Veriton M4630G running Windows 7, and created a website that integrates Typing-Proof, Sound-Proof, and SMS-based 2FA. All participants were required to install our application on their phones which supports all three authentication mechanisms.

During the user study, each participant was asked to log in to the server using all three mechanisms in random order and for several times. After using all 2FA mechanisms, participants were required to fill in a survey. The survey includes three parts: demographic information, System Usability Scale (SUS) [11], and a post-test questionnaire which covers various aspects of 2FA mechanisms that are not covered by the SUS.

7.2 Usability

Survey Results. All participants had ever used 2FA for online banking and 88% of them had the experience of using 2FA for online payment. Only 28% and 24% of them respectively ever used 2FA in Google Services (e.g., Gmail) and Apple Services (e.g., iCloud). Our experience is that many finance-related services, like online banking or online payment, enforce users to use 2FA, while other services, such as Gmail or iCloud, make 2FA optional, in which case many users choose to opt out.

The System Usability Scale (SUS) is widely used to assess the usability of IT systems [6]. Its score ranges from 0 to 100, where a higher score indicates better usability. Appendix C reports the items of SUS. The mean SUS scores for Typing-Proof, Sound-Proof, and SMS-based 2FA are 81.7 (±14.68), 69.2 (±18.02), 73.4 (±12.62), respectively. The result shows that the usability of Typing-Proof is obviously better than the other two mechanisms. One interesting observation is that the mean SUS score of Sound-Proof is a little bit lower than that of SMS-based 2FA while the standard deviation of Sound-Proof's SUS score is larger than that of SMS-based 2FA. One potential reason is that some participants' browsers do not support audio recording, which contributes to the low scores on the usability evaluation of Sound-Proof.

The post-test questionnaire is similarly designed as that in [23], aiming to collect information on the perceived quickness of the three mechanisms (2FA-quick for short in Figure 8) and participants' willingness to adopt them (2FA-mandatory in mandatory setting, and 2FA-optional in optional setting for short in Figure 8). It also inquires of participants whether they feel comfortable using the mechanisms in different environments, including use @ home, use @ workplace, use @ café, use @ library in Figure 8. The full post-test questionnaire is listed in Appendix D. Figure 8 summarizes the participants' answers on 5-point Likert-scales in a radar chart plot. In general, participants show the strongest willingness to adopt Typing-Proof. Most participants evaluated that both Typing-Proof and Sound-Proof are much quicker than SMS-based 2FA. Similar to [23], our results show that participants tend not to use SMS-based 2FA if it is optional, while for Typing-Proof and Sound-Proof, the difference in users' acceptance between mandatory setting and optional setting is much less significant. More than 88% of participants evaluated that Typing-Proof is suitable to be used at home, at their workplace, while fewer participants would use Typing-Proof in a public place (i.e., at a café or library). As for SMS-based 2FA, participants shared a similar willingness in various scenarios.

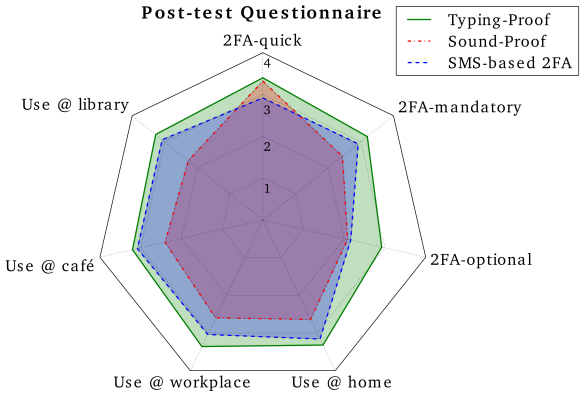


Figure 8: Answer to the post-test questionnaire.

Login Time. The login time we measured in our user study is from the start of the second-factor authentication (i.e., after username and password is verified), to the moment when the login attempt is accepted. We did not witness any login failure in our experiment. Therefore, the login time of Typing-Proof does not include the potential time of one-button authentication in the backup solution, which takes 7.1 seconds on average as measured separately in our experiments. The averaged login time for Typing-Proof is 4.3 seconds while it is 5.0 seconds for Sound-Proof and 10.4 seconds for SMS-based 2FA. Among the three mechanisms, Typing-Proof is the most favorable in terms of login time.

8 DISCUSSION

Adjustable Security. When Typing-Proof is used, a longer random code makes it more difficult for an attacker to launch a successful attack. On the other hand, typing longer random codes will lower the usability of Typing-Proof. Therefore, users may take different typing strategies according to their needs. For important authentications in a less secure environment, users may choose to type longer random codes, while for less important authentications in a more secure environment, users may choose to type shorter random codes.

Transaction Authentication. Transaction authentication is another important application of 2FA. Transaction authentication may suffer from Man-in-the-Mobile attack and Man-in-the-Browser attack [1] in which an attacker may change the content of a transaction such as destination account number and transaction amount. To solve this problem in Typing-Proof, user's transaction details should be sent to user's registered phones via the server. Users should use the backup solution of Typing-Proof and check it out before pressing the 'Approve' button for transaction authentication.

CAPTCHA. Many existing authentication systems involve the use of CAPTCHA. Typing-Proof can be easily used when CAPTCHA is involved. Instead of typing random codes, users may type CAPTCHA codes for 2FA. The overall user experience of Typing-Proof is same with CAPTCHA-based password authentication if the backup solution is not triggered.

Keyboard Protector. Some users may place keyboard protectors on their keyboards in order to prevent dust entry or liquids. Commercial keyboard protectors made of silicone can effectively reduce keystroke sound. It may lead to high FRR when users

login to their accounts using Typing-Proof. In the light of this, we recommend users to take the keyboard protector off when they use Typing-Proof (otherwise they need to resort to the backup solution).

Combined with other mechanisms. Typing-Proof can be combined with other 2FA mechanisms, including Sound-Proof, hardware token based 2FA, and SMS-based 2FA. Furthermore, Typing-Proof and Sound-Proof can work simultaneously to make authentications more usable and secure. In particular, if a user logins to his/her account in a quiet environment where Sound-Proof does not work, the server can rely on Typing-Proof for 2FA. On the other hand, if the user uses 2FA in a noisy environment, Sound-Proof activated for a better decision.

Alternative Devices. Currently, Typing-Proof uses a smartphone as a software token. It is straightforward to replace it with other smart devices such as smartwatch for 2FA. Compared to using smartphone, the use of smartwatch in Typing-Proof may further lower FRR since the distance between the keyboard of a login computer and the smartwatch of a user who logins to his/her account wearing the smartwatch should be shorter than the short distance (i.e., 20cm) that is used in our experiments.

Comparative Analysis. The framework of Bonneau et al. [10] can be used to compare Typing-Proof, Sound-Proof and SMS-based 2FA in terms of usability, deployability, and security. Table 3 in the Appendix E shows the comparison. In general, both Typing-Proof and Sound-Proof achieve better usability than SMS-based 2FA. The deployability of Typing-Proof is better than Sound-Proof since Sound-Proof may not be exactly browser-compatible. We observed several cases where participants' browsers did not support audio recording in our user study. As for the security aspect of the comparison, Typing-Proof is better than Sound-Proof.

9 CONCLUSION

This paper presents Typing-Proof, a usable, secure, and low-cost two-factor authentication mechanism. Typing-Proof does not require a user to interact with his/her phone in most cases and does not have any memory demand. It can be used in any environment and is compatible with major browsers, PCs, and phones without requiring any additional plug-ins or hardware. Typing-Proof is secure against practical attacks, including remote attack, sound-danger attack, co-located attack, and relay attack. Compared to hardware token based 2FA, SMS-based 2FA and Sound-Proof, Typing-Proof enables significant cost saving for both service providers and users. This brings in high commercial potential which may foster large-scale adoptions.

REFERENCES

[1] Manal Adham, Amir Azodi, Yvo Desmedt, and Ioannis Karaolis. 2013. How to attack two-factor authentication internet banking. In *International Conference on Financial Cryptography and Data Security*. Springer, 322–328.
[2] Aladdin. 2018. Two-Factor Authentication – The Real Cost of Ownership. https://mpa.co.nz/media/4410/twofactorauthenticationtherealcostofownership.pdf. (2018).
[3] Dmitri Asonov and Rakesh Agrawal. 2004. Keyboard acoustic emanations. In *IEEE Proceedings of the 2004 Symposium on Security and Privacy*. 3–11.
[4] AWS. 2018. Amazon EC2 Pricing. https://aws.amazon.com/cn/ec2/pricing/on-demand. (2018).
[5] AWS. 2018. Worldwide SMS Pricing. https://aws.amazon.com/cn/sns/sms-pricing. (2018).
[6] Aaron Bangor, Philip T Kortum, and James T Miller. 2008. An empirical evaluation of the system usability scale. *Intl. Journal of Human–Computer Interaction* 24, 6 (2008), 574–594.
[7] Francesco Bergadano, Daniele Gunetti, and Claudia Picardi. 2002. User authentication through keystroke dynamics. *ACM Transactions on Information and System Security (TISSEC)* 5, 4 (2002), 367–397.
[8] Yigael Berger, Avishai Wool, and Arie Yeredor. 2006. Dictionary attacks using keyboard acoustic emanations. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*. ACM, 245–254.
[9] Blizzard. 2015. Introducing The One-Button Authenticator. http://us.battle.net/heroes/en/blog/20152210/introducing-the-one-button-authenticator-6-16-2016. (2015).
[10] Joseph Bonneau, Cormac Herley, Paul C Van Oorschot, and Frank Stajano. 2012. The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. In *Security and Privacy (SP), 2012 IEEE Symposium on*. IEEE, 553–567.
[11] John Brooke et al. 1996. SUS-A quick and dirty usability scale. *Usability evaluation in industry* 189, 194 (1996), 4–7.
[12] Nelson Cowan. 2001. The Magical Number 4 in Short-term Memory: A Reconsideration of Mental Storage Capacity. *Behavioral and Brain Sciences* 24 (2001), 87–114.
[13] Alexei Czeskis, Michael Dietz, Tadayoshi Kohno, Dan Wallach, and Dirk Balfanz. 2012. Strengthening user authentication through opportunistic cryptographic identity assertions. In *Proceedings of the 2012 ACM conference on Computer and Communications Security*. ACM, 404–414.
[14] Duo-Security. 2018. DUO PUSH: Quickly Verify Your Identity. https://duo.com/product/trusted-users/two-factor-authentication/authentication-methods/duo-push. (2018).
[15] Facebook. 2017. Security Key for safer logins with a touch. https://www.facebook.com/notes/facebook-security/security-key-for-safer-logins-with-a-touch/10154125089265766. (2017).
[16] Futurea. 2018. Futurae Authentication Suite. https://futurae.com/product/. (2018).
[17] Sujata Garera, Niels Provos, Monica Chew, and Aviel D Rubin. 2007. A framework for detection and measurement of phishing attacks. In *Proceedings of the 2007 ACM workshop on Recurring malcode*. ACM, 1–8.
[18] GOOGLE. 2018. Firebase Cloud Messaging. https://firebase.google.com/docs/cloud-messaging. (2018).
[19] GOOGLE. 2018. Google 2-Step Verification. https://www.google.com/landing/2step. (2018).
[20] Google. 2018. NotificationListenerService. https://developer.android.com/reference/android/service/notification/NotificationListenerService.html. (2018).
[21] Robert Hackett. 2016. LinkedIn Lost 167 Million Account Credentials in Data Breach. http://fortune.com/2016/05/18/linkedin-data-breach-email-password. (2016).
[22] Nikolaos Karapanos and Srdjan Capkun. 2014. On the Effective Prevention of TLS Man-In-The-Middle Attacks in Web Applications.. In *Proceedings of the 23th Conference on USENIX Security Symposium*.
[23] Nikolaos Karapanos, Claudio Marforio, Claudio Soriente, and Srdjan Capkun. 2015. Sound-Proof: Usable Two-Factor Authentication Based on Ambient Sound.. In *Proceedings of the 24th Conference on USENIX Security Symposium*. 483–498.
[24] Swati Khandelwal. 2018. Download: 68 Million Hacked Dropbox Accounts are Just a Click Away! https://thehackernews.com/2016/10/dropbox-password-hack.html. (2018).
[25] Manu Kumar, Tal Garfinkel, Dan Boneh, and Terry Winograd. 2007. Reducing shoulder-surfing by using gaze-based password entry. In *Proceedings of the 3rd symposium on Usable privacy and security*. ACM, 13–19.
[26] MDN. 2018. Bluetooth.requestDevice(). https://developer.mozilla.org/en-US/docs/Web/API/Bluetooth/requestDevice. (2018).
[27] MDN. 2018. MediaDevices.getUserMedia(). https://developer.mozilla.org/zh-CN/docs/Web/API/MediaDevices/getUserMedia. (2018).
[28] Microsoft. 2018. One easy-to-use app for all your multi-factor authentication needs. https://dirteam.com/sander/2016/08/15/microsoft-authenticator-one-easy-to-use-app-for-all-your-multi-factor-authentication-needs/. (2018).
[29] Fabian Monrose and Aviel D Rubin. 2000. Keystroke dynamics as a biometric for authentication. *Future Generation Computer Systems* 16, 4 (2000), 351–359.
[30] NetApplications. 2018. Browser Market Share. https://www.netmarketshare.com/browser-market-share.aspx. (2018).
[31] Eric Ravenscraft. 2018. LastPass Authenticator Now Has a One-Button Approval Option. https://lifehacker.com/lastpass-authenticator-now-has-a-one-button-approval-op-1785138823. (2018).
[32] SAASPASS. 2018. Two-factor Authentication with Proximity Uses iBeacon Bluetooth Low Energy (BLE) to Authenticate Users Instantly. https://saaspass.com/technologies/proximity-instant-login-two-factor-authentication-beacon.html. (2018).
[33] Maliheh Shirvanian, Stanislaw Jarecki, Nitesh Saxena, and Naveen Nathan. 2014. Two-Factor Authentication Resilient to Server Compromise Using Mix-Bandwidth Devices.. In *Proceedings of the 21st Annual Network and Distributed System Security Symposium, NDSS*.
[34] Agent Smith. 2016. 1 Billion Accounts are leaked from yahoo's database. https://latesthackingnews.com/2016/12/15/1-billion-accounts-leaked-

yahoos-database. (2016).
[35] Dawn Xiaodong Song, David Wagner, and Xuqing Tian. 2001. Timing Analysis of Keystrokes and Timing Attacks on SSH.. In *Proceedings of the 10th Conference on USENIX Security Symposium*. USENIX Association.
[36] Routhu Srinivasa Rao and Alwyn R. Pais. 2017. Detecting Phishing Websites Using Automation of Human Behavior. In *Proceedings of the 3rd ACM Workshop on Cyber-Physical System Security*. ACM, 33–42.
[37] StatCounter. 2018. Mobile Vendor Market Share Worldwide. http://gs.statcounter.com/vendor-market-share/mobile/worldwide. (2018).
[38] Zhu Tong, Qiang Ma, Shanfeng Zhang, and Yunhao Liu. 2014. Context-free attacks using keyboard acoustic emanations. In *Proceedings of the 21st ACM Conference on Computer and Communications Security*. ACM, 453–464.
[39] W3school. 2018. jQuery keydown() Method. https://www.w3schools.com/jquery/event_keydown.asp. (2018).
[40] Wikipedia. 2018. Network Time Protocol. https://en.wikipedia.org/wiki/Network_Time_protocol. (2018).
[41] Qiang Yan, Jin Han, Yingjiu Li, and Robert H. Deng. 2012. On Limitations of Designing Leakage-Resilient Password Systems: Attacks, Principles and Usability. In *Proceedings of the 19th Annual Network and Distributed System Security Symposium, NDSS*. Internet Society.
[42] Kehuan Zhang and XiaoFeng Wang. 2009. Peeping Tom in the Neighborhood: Keystroke Eavesdropping on Multi-user Systems. In *Proceedings of the 18th Conference on USENIX Security Symposium*. USENIX Association, 17–32.
[43] Linghan Zhang, Sheng Tan, Jie Yang, and Yingying Chen. 2016. Voicelive: A phoneme localization based liveness detection for voice authentication on smartphones. In *Proceedings of the 23rd ACM Conference on Computer and Communications Security*. ACM, 1080–1091.
[44] Li Zhuang, Feng Zhou, and J. Doug Tygar. 2009. Keyboard acoustic emanations revisited. *ACM Transactions on Information and System Security* 13, 1 (2009), 3:1–3:26.

A QUANTITATIVE USABILITY ANALYSIS FRAMEWORK

A quantitative analysis framework can be used for evaluating the usability cost of authentication systems [41]. This framework decomposes the process of human-computer interaction into atomic cognitive operations in psychology. The quantitative analysis framework consists of two components, including cognitive workload and memory demand.

Cognitive Workload. Cognitive workload is measured by the total reaction time required by the involved cognitive operations. Parallel recognition is a major cognitive operations in human-computer interactions. It can be considered as a matching process of comparing presented items with those stored in memory. According to the short-term memory capacity theory [12], the maximum number of parallel recognition channels is limited to 4 for an average volunteer. The reaction time of recognizing x items displayed simultaneously can be estimated as $RT = 0.4077 \cdot \lceil x/4 \rceil$.

Memory Demand. For memory demand, the cost for recalling k items is $k/29.6\%$. For memorizing a verification code, we consider the x-digit code as $\lceil x/4 \rceil$ items since the short-term memory capacity is 4 for an average volunteer [12]. Therefore, the memory demand for recalling x-digit code is $\lceil x/4 \rceil /29.6\%$.

B PROTOTYPE IMPLEMENTATION

Web Server Settings. Authentication server is implemented using CherryPy web framework. SQLite database is used to store username and password information. For experimental evaluation, we store each keystroke timing sequence and the corresponding random code into a text document and store the corresponding audio data into a 16-bit byte array in the debug version for data collection. In the released version of Typing-Proof, no keystroke timing information or audio is stored on server side. HTTPS is supported for communications between browsers/login computers and server, and between server and registered phones.

Web Client Settings. All major browsers support our prototype without any browser code modifications or plug-ins. In our experiment, we test our prototype on Google Chrome (version 55.0.2883.87), Internet Explorer 11 (version 11.0.9600.18860) and Microsoft Edge (version 41.16299.15.0). The client website is written entirely in HTML and JavaScript. We use jQuery keydown() Method [39] to record the timestamp of each key press event. We use Ajax to send and retrieve data from the server to the client asynchronously.

Mobile Client Settings. We develop an Android application and test it on a Google Nexus 5x, a Google Nexus 6 (both running on Android version 6.0.1) and a Huawei P10 (running on Android 8.0.1) smartphones. We use the Google Firebase Cloud Messaging (FCM) service [18] to send indicators to Android devices.

Time Synchronization. Typing-Proof requires that registered phones and corresponding login computers are loosely synchronized. For this reason, login computers and registered phones run a simple time-synchronization protocol (Network Time Protocol [40]) with the server. In a server-client scenario (where the client can be either login computer or registered phone), the client initiates a time-request exchange with the server so that the client is able to calculate the link delay and its local offset, and adjust its local clock to match the clock at the server's computer. The protocol can synchronize all participating devices and mitigate the effects of variable network latency. According to our experimental results, the NTP protocol usually maintains clock difference within tens of milliseconds, which is good enough for Typing-Proof.

C SYSTEM USABILITY SCALE

We list the items of the System Usability Scale [11]. All items are answered with a 5-point Likert-scale from *Strongly Disagree* to *Strongly Agree*.

Q1 I would like to use this system frequently.
Q2 I found the system unnecessarily complex.
Q3 I thought the system was easy to use.
Q4 I would need the support of a technical person to be able to use this system.
Q5 I found the various functions in this system were well integrated.
Q6 I thought there was too much inconsistency in this system.
Q7 I would imagine that most people would learn to use this system very quickly.
Q8 I found the system very cumbersome to use.
Q9 I felt very confident using the system.
Q10 I needed to learn a lot of things before I could get going with this system.

D POST-TEST QUESTIONNAIRE

We list the items of the post-test questionnaire. All items are answered with a 5-point Likert-scale from *Strongly Disagree* to *Strongly Agree*.

Q1 I thought this system was quick. (2FA-quick)
Q2 If 2FA were mandatory, I would use this system to log in (2FA-mandatory).
Q3 If 2FA were optional, I would use this system to log in (2FA-optional).
Q4 I would feel comfortable using this system at home (Use @ home).

Table 3: Comparison of Typing-Proof against Sound-Proof [23] and SMS-based 2FA [19] using the framework of Bonneau et al. [10]. We use 'Y' to denote that the benefit is provided and 'S' to denote that the benefit is somewhat provided.

	Usability								Deployability						Security										
Scheme	Memorywise-Effortless	Scalable-for-Users	Nothing-to-Carry	Physically Effortless	Easy-to-Learn	Efficient-to-Use	Infrequent-Errors	Easy-Recovery-from-Loss	Accessible	Negligible-Cost-per-User	Server-Compatible	Browser-Compatible	Mature	Non-Proprietary	Resilient-to-Physical-Observation	Resilient-to-Targeted-Impersonation	Resilient-to-Throttled-Guessing	Resilient-to-Unthrottled-Guessing	Resilient-to-Internal-Observation	Resilient-to-Leaks-from-Other-Verifiers	Resilient-to-Phishing	Resilient-to-Theft	No-Trusted-Third-Party	Requiring-Explicit-Consent	Unlinkable
Typing-Proof		Y	S	Y	Y	Y	S	S	Y	Y		Y		Y	S	S	Y	Y		Y	Y	Y	Y	Y	
Sound-Proof		Y	S	Y	Y	Y	S	S	Y	Y		S		Y	S		S	Y		Y	Y	Y	Y	Y	
SMS-based 2FA			S		Y	S	S	S	S	S		Y	Y	Y	S	S	Y	Y		Y	Y	Y	Y	Y	

Q5 I would feel comfortable using this system at workplace (Use @ workplace).
Q6 I would feel comfortable using this system at café (Use @ café).
Q7 I would feel comfortable using this system at library (Use @ library).

E COMPARISON RESULTS

Table 3 shows the comparison results. In particular, the metrics of usability include the columns from "Memorywise-Effortless" to "Easy-Recovery-from-Loss". The metrics of deployability include the columns from "Accessible" to "Non-Proprietary". The metrics of security include the columns from "Resilient-to-Physical-Observation" to "Unlinkable".