TELKOMNIKA, Vol.17, No.2, April 2019, pp.693~702
ISSN: 1693-6930, accredited First Grade by Kemenristekdikti, Decree No: 21/E/KPT/2018
DOI: 10.12928/TELKOMNIKA.v17i2.9187

2FYSH: two-factor authentication you should have for password replacement

Sunderi Pranata, Hargyo Tri Nugroho*
Internet of Things Research Group, Department of Computer Engineering,
Universitas Multimedia Nusantara, Tangerang, Indonesia
*Corresponding author, e-mail: hargyo@umn.ac.id

Abstract

Password has been the most used authentication system these days. However, strong passwords are hard to remember and unique to every account. Unfortunately, even with the strongest passwords, password authentication system can still be breached by some kind of attacks. 2FYSH is two tokens-based authentication protocol designed to replace the password authentication entirely. The two tokens are a mobile phone and an NFC card. By utilizing mobile phones as one of the tokens, 2FYSH is offering third layer of security for users that lock their phone with some kind of security. 2FYSH is secure since it uses public and private key along with challenge-response protocol. 2FYSH protects the user from usual password attacks such as man-in-the-middle attack, phishing, eavesdropping, brute forcing, shoulder surfing, key logging, and verifier leaking. The secure design of 2FYSH has made 90% of the usability test participants to prefer 2FYSH for securing their sensitive information. This fact makes 2FYSH best applied to secure sensitive data needs such as bank accounts and corporate secrets.

Keywords*: NFC, password replacement, secure authentication, two-factor authentication, usable security*

Copyright © 2019 Universitas Ahmad Dahlan. All rights reserved.

1. Introduction

Password is a cheapest and the most general authentication method we use today. However, good passwords are hard to remember. A good password should be long enough; unpredictable; combined with uppercase, lowercase, symbols, and numbers. Based on password survey conducted by Hoonakker, et al [1] on 836 employees with the user composition of 3% novice users (new computer users), 69% average users (able to use word commonly used apps like word processors, spreadsheets, email, browsers), 22% advanced users (able to install software, configure settings in app), and 6% expert users (able to install OS, write programs and apps), the research founds that 94% of the respondents are not practicing the best practice of password usage [1]. In terms of security, password also performs badly. Beside passwords are easily guessable, passwords are prone to well-known attacks such as brute force, shoulder surfing, key logging, eavesdropping, phishing, and verifier leak. Some of the attacks can even beat the strongest passwords. For example, attackers can breach the database at the verifier and get the passwords. When the same passwords are used on multiple sites, attackers can have access to those sites.

All approaches for human authentication rely on at least one of the following: Something You Know (SYK), Something You Have (SYH), and Something You Are (SYA) [2]. These problems on passwords are actually quite solved by the Multi-Factor Authentication (MFA) concept [3-5]. MFA combines two or more authentication factors from the 3 authentication factor types. One of the most used forms of two-factor authentication (2FA) is password and one-time password (OTP). When using this 2FA combination, users are still required to type the main password, then users have to wait for SMS, or open an app, or click certain buttons on a dedicated authentication device (token) to get the OTP needed and have to remember them for short amount of time to actually enter it to the system. The process just takes too much time and is quite hard to learn. Besides the downside of the usability in OTP, OTP actually is still technically a password and can be phished [6-10].

Meanwhile, NFC cards are widely used in places like universities and offices for verifying attendance and building access. Also, the number of cashless transactions using NFC cards has increased exponentially over the years. Smartphone manufacturers also producing

Received March 5, 2018; Revised November 14, 2018; Accepted December 5, 2018

more smartphones with NFC chip in it. It is estimated that 64% of smartphones will be NFC enabled phones by 2018 [11]. Assessing the downsides in password and OTP systems, in this research we propose a new authentication protocol called 2FYSH. 2FYSH is a secure-token based authentication protocol using two SYH factors which are mobile phones and NFC cards. 2FYSH also offers the third factor to people who lock their phone with either SYK (pattern lock, PIN, password, etc.) or SYA (face recognition, and fingerprint) factor.

2. Related Works

One-time password (OTP) is a password that is valid only after one authentication [12]. OTP authentication was created mainly to secure the system from eavesdropping and replay attacks on usual password authentication method [13]. RSA SecurID [14] and Google Authenticator [15] are well-known examples of OTP implementation. Unfortunately, common OTP mechanisms such as [16, 17] have the possibility of vulnerability on the server side. When compromised, attackers might obtain the shared secrets and compute the OTP to authenticate. Compared to that, 2FYSH protocol stays safe even when the server is compromised and cannot be phished-thanks to 2FYSH's private and public key with challenge-response scheme. In terms of usability, 2FYSH acts like requiring users to bring both RSA SecurID and smartphone at the same time, but a smartphone and an NFC card is something that user relatively always have with them.

FIDO U2F (Universal Second Factor) has created the login process very easy for users. By only presenting the second factor in the form of a USB device and tapping it when the light blinks, users can authenticate themselves to the system. Since the USB device acts like a physical key, the implementation is usually combined with a PIN. FIDO U2F uses the same public and private with a challenge-response protocol [18, 19]. It is still relatively a burden for users to bring a USB device all the time with them, compared to 2FYSH that requires the user to bring an NFC card which usually is worn as a name tag or placed inside the wallet. The main advantages of 2FYSH are that 2FYSH doesn't require the user to buy any special device, compared to the FIDO U2F USB device which still costs some amount money (by November 2017).

Pico [20] is a password replacement authentication proposed by Stajano in 2011. While 2FYSH is using the same two hardware tokens concept to authenticate, 2FYSH and Pico act a little bit different. Pico can have a lot of picosiblings which is recommended to be wearable devices such as earrings, watch, and necklaces. To authenticate, Pico device will need to communicate actively by sending ping and pong with the picosiblings to unlock the secret key located at Pico device. Because of that, Pico supports continuous authentication which means when Pico separates from picosiblings, it will automatically log the user out from the system.

Because Pico is recommended to be a dedicated device with capabilities of camera and radio, it requires the user to bring/wear minimum one extra hardware token with them (proposed as wearable device). Since Pico and picosiblings need to communicate actively, it will need a lot amount of energy which results in the need of frequent charging on the devices (Pico & picosiblings). In contrast, 2FYSH only need a smartphone to recharge which should not really matter because it is considered normal to recharge a smartphone. 2FYSH also requires the user to bring much normal tokens (such as a card compared to techy earrings) for users than Pico. Moreover, to use Pico, it is definitely will cost more rather than 2FYSH since all of Pico system are some unusual extra dedicated tokens.

The implementation of 2FYSH is quite similar to Loxin [21] by using public and private key with the private key stored in the smartphone. The use of trusted third party is a minus score on the security aspect according to Bonneau's proposed UDS framework [22]. Loxin still uses a trusted third party in the system while 2FYSH doesn't require it. 2FYSH is safe because it does not use CA by design and also offers the second protection by the need of presenting an NFC card to authenticate.

3. Proposed Design

2FYSH is an abbreviation of "Two Factor (authentication) You Should Have" and read as "two fish". By the name, the protocol is offering two layers of authentication of SYH factor. Those two factors are a mobile phone and an NFC card. Simply put, to authenticate to the relying party, the user should be able to present a registered NFC card to the mobile phone.

The main strength of this authentication protocol is on the two factors of SYH. 2FYSH could also be three factors if the users lock their phone with some kind of security such as SYK factor (e.g. PIN, password, pattern lock) or SYA factor (e.g. fingerprint, face recognition) factor. The additional third factor is considered as not an additional cognitive burden to the user since it is frequently used every day.

2FYSH protocol uses **TLS-like public-private key scheme** combined with **challenge-response protocol** to provide secure authentication. The private key should be stored on the mobile phone which has a special key storage hardware called secure element. This way, the secrets will be kept secret not only by software but also hardware; thus, preventing any (unauthorized) access to the private key even the user has rooted the phone [23, 24]. The NFC card used to authenticate will also be secured by encrypting the salt (will be explained further in the next section) data with the public key. **Requirement:** To make sure that the NFC card's salt data could not be compromised, all communication has to be over a secure channel (eg. HTTPS, IPSec, etc). Otherwise, 2FYSH becomes one-factor authentication (smartphone only).

3.1. Architecture

In the 2FYSH protocol, there will be 4 main entities, as shown in Figure 1:

1. 2FYSH server: A back-end server for 2FYSH service. This back-end server stores the 2FYSH users' credentials, generate challenges, verify user authentication.
2. 2FYSH app: A 2FYSH authenticator application installed on the users' mobile phone. It can also be installed on computers but it will be less practical for the user since computers are not designed to be as mobile as mobile phones. This application stores all 2FYSH credentials of the user. 2FYSH app also reads the challenge generated by the 2FYSH server and sends the authentication response to the server for verifying purposes.
3. Client: An application which users use that provides the interface to sign up or sign in. This client can be in any form such as desktop/mobile browser, desktop/mobile app.
4. NFC card: A token which users need to present to the 2FYSH app in order to sign in.
   The NFC card will hold a salt data encrypted with the user credential's public key.

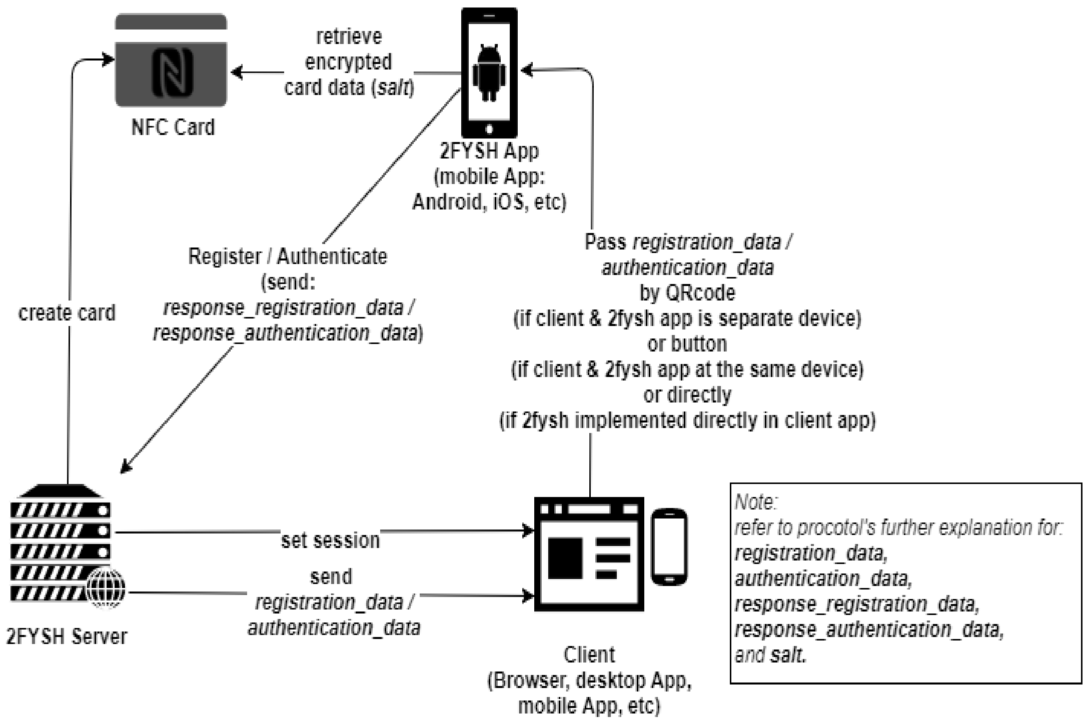

NFC Card
retrieve encrypted card data (*salt*)
2FYSH App (mobile App: Android, iOS, etc)
Register / Authenticate (send: *response_registration_data* / *response_authentication_data*)
Pass *registration_data* / *authentication_data* by QRcode (if client & 2fysh app is separate device) or button (if client & 2fysh app at the same device) or directly (if 2fysh implemented directly in client app)
create card
set session
send *registration_data* / *authentication_data*
2FYSH Server
Client (Browser, desktop App, mobile App, etc)
Note: refer to procotol's further explanation for: *registration_data*, *authentication_data*, *response_registration_data*, *response_authentication_data*, and *salt*.

Figure 1. 2FYSH protocol model

The data on the client could not be just transferred to the 2FYSH app since they may be on separate devices. The 2FYSH protocol also defines some special schemes for data passing between the client and the 2FYSH app.

1. 2D visual code: This scheme is used whenever client and 2FYSH app are separate devices. For example: the client is a desktop browser and the 2FYSH app is on the mobile phone (always).
2. Button (URI link): This scheme is used whenever client and 2FYSH app are on one same device. For example: the client is a mobile browser and on the same device with the 2FYSH app.
3. Directly from the app: This scheme is used whenever the client is the mobile app and 2FYSH app functionality is implemented directly in the app. With this, the data sent from the 2FYSH server can be directly processed in the app without the need of passing data to the 2FYSH app. For example: Facebook mobile app implements 2FYSH protocol directly to the app.

3.2. Registration

On the registration process, there are two types of data being sent. The first data being sent is called registration data, the data is being sent by 2FYSH server to the client. The second data is the response data from the client to the server and will be called registration response data.

3.2.1. 2FYSH Registration-Parameters

The registration data sent from server to client will be having 5 key parameters: action, username, app ID, challenge, and register portal. Parameter action is to identify “registration” or “authentication” action for the client. Since the user may have more than one credential for each app (App ID), parameter username is needed for identifying purposes. The challenge is a random 128, 192, or 256 bits (encoded to HEX), generated by the 2FYSH server with secure random byte generator. All the parameters will be processed by the 2FYSH app and are going to be sent to the register portal which acts as service.

The 2FYSH app will process the data needed and sends back registration response data to the server. The registration response data will be having 4 parameters: username, challenge, public key, and key handle. The 2FYSH app will process the registration data and create a key pair (public and private key). The private key will be stored on the phone’s secure element (key store) and the public key is sent through a secure channel to the 2FYSH server. Since the private key is stored on a secure element, it will need an alias to call the specific key for doing procedures. The alias is created as a random 32, 64, or 128 bits (encoded in HEX) and it is called as key handle. The challenge and key handle parameters should be sent in encrypted form using private key.

3.2.2. 2FYSH Registration-Stored Data

Aside from all the data being sent, there will also some data being created and stored through the registration process.

1. Counter: is an integer value with an initial value of 0; stored by both 2FYSH server and 2FYSH client. The counter will act as the last measure of the security to detect if somehow in the future, the system could be broken. Counter value will be incremented by the 2FYSH server and the 2FYSH app every successful authentication. Whenever the counter value of 2FYSH server is larger than the 2FYSH client, there will be a high chance that the 2FYSH app has been cloned.
2. Salt: is a random 32 or 64 bits (encoded to HEX). The value of salt will later be encrypted by the 2FYSH server using public key received on the registration process and the encrypted value will be written to an NFC card. The card will be issued to the user for authentication by the relying party.

3.2.3. 2FYSH Registration-Process

On the registration process, registration_data will be sent by 2FYSH server to the client. The client will then pass the data through the stated data passing schemes (look point 3.2.1). The 2FYSH app will receive the data and then create the key pair and the key handle to access it. The 2FYSH app will store the credential data needed for authentication and user interface (key handle, username, app id, counter=0). After all that, the app will send back registration_response_data to the server. The 2FYSH server will create salt for the corresponding credential and store the credential data to its database (username, public key, salt, key handle, counter=0). The salt value will be encrypted with public key, and the encrypted value will be written to an NFC card. Then, the relying party will issue the NFC card to the user.

3.3. Authentication

On the authentication process, there are two types of data being sent. The first data being sent is called authentication_data, the data is being sent by 2FYSH server to the client. The second data is the response data from the client to the server and will be called authentication_ response_data.

3.3.1. 2FYSH Authentication-Parameters

The authentication data sent from server to client will be having 6 key parameters: action, username, app ID, challenge, key handle, and authentication portal. The authentication response data sent from the client back to the server will be having these 3 key parameters: username, challenge+salt, and counter. Challenge+salt is concatenated bits of challenge and salt. The value of salt is retrieved by 2FYSH app through the NFC card. Since the NFC card only holds the public-key-encrypted value of salt, the app will decrypt it by using the credential’s private key (recognized by the key handle). The 2FYSH app stored counter value will also be sent to the server. The challenge+salt and counter data will be sent in encrypted form using private key.

3.3.2. 2FYSH Authentication–Process

On the authentication process, authentication_data will be sent by 2FYSH server to the client. The client will then pass the data through the stated data passing schemes (look point 3.2.1). The 2FYSH app will receive the data and will also wait for the NFC tap to the phone to get the encrypted value of salt. By referencing to the key handle, the 2FYSH app will do three things: retrieve the current value of counter from the database, decrypt the encrypted salt value using its corresponding private key to get the plain salt value, and encrypt the challenge+salt value and the counter. The 2FYSH server will decrypt the encrypted data and get the challenge+salt value. The authentication process is done on the 2FYSH server. The server will concatenate the challenge that had been sent with stored salt value and compare it to the challenge+salt value received; and if it matches, the authentication is successful; thus, the 2FYSH server could set a session/establish a connection to the client.

4. Implementation

We implemented the initial prototype of the protocol design with a web server as the 2FYSH server, desktop/mobile browser as the client, android application as the 2FYSH application, and NTAG215 NFC card.

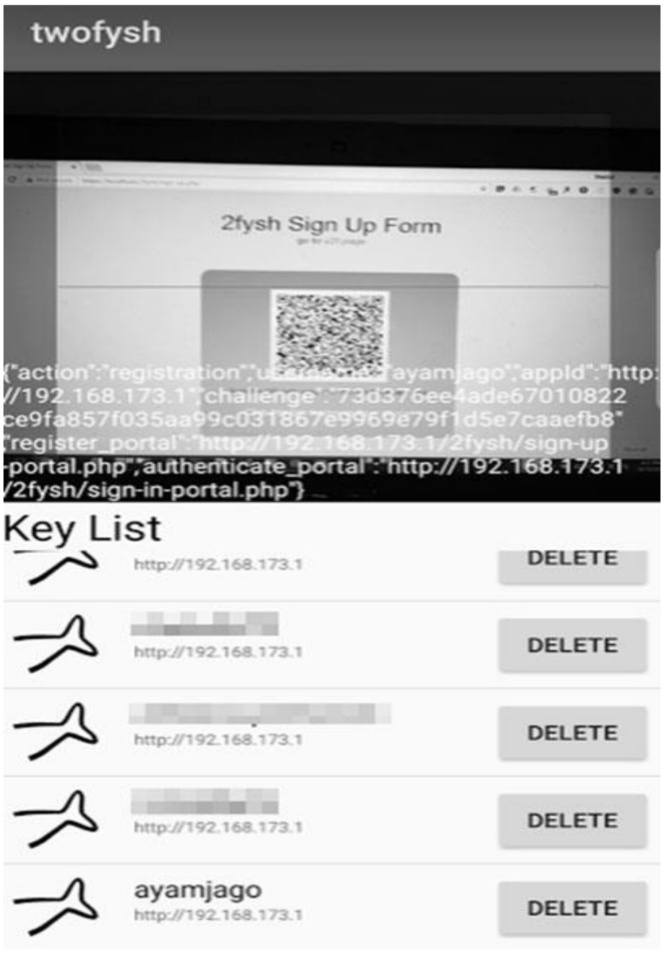
4.1. 2FYSH Server Implementation

2FYSH prototype on the server side is using simple web interface. First, the user will need to enter the username registered. After that, the browser screen will show a QR code and a URI link. The QR code is used when the browser is on the desktop platform. The 2FYSH app will have a viewfinder searching for QR code with the 2FYSH data format. The URI link is used when the browser is on the same mobile device with the 2FYSH app. The communication data used in this prototype implementation is JSON. The JSON data is embedded in the QR code and the URI link.

4.2. 2FYSH App Implementation

The implementation of the 2FYSH app is made as simple as possible. The main interface of the app constructed mainly by a camera view as shown in Figure 2. In order to do either registration or authentication process, no button is needed to be pressed. The user will just need to point the camera to the QR code shown from the browser and it will prompt for NFC card tap to the phone. Once the NFC card tap has been done, the browser will automatically load the requested page. If the browser is in the same mobile phone as the 2FYSH app, the user will just have to click the URI link and the 2FYSH app will open directly to the NFC card tap prompt page and the rest is the same.

The 2FYSH app design supports the interface of key list to the users. Key list is a list of credentials that had been registered to the 2FYSH app. With the key lists, users can look at the credentials they had registered and revoke them whenever they want. A single key is as shown in the red box in Figure 2, with a button DELETE on the right to delete the credential.



```
{"action":"registration","username":"ayamjago","appId":"http:
//192.168.173.1","challenge":"73d376ee4ade67010822
ce9fa857f035aa99c031867e9969e79f1d5e7caaefb8"
,"register_portal":"http://192.168.173.1/2fysh/sign-up
-portal.php","authenticate_portal":"http://192.168.173.1
/2fysh/sign-in-portal.php"}
```

Key List

http://192.168.173.1		DELETE
[redacted]	http://192.168.173.1	DELETE
[redacted]	http://192.168.173.1	DELETE
[redacted]	http://192.168.173.1	DELETE
ayamjago	http://192.168.173.1	DELETE

Figure 2. 2FYSH Mobile App User Interface

5. Analysis

5.1. Usability Test

Based on Jakob Nielsen's mathematical model on usability testing at 1993, only minimum 3 to 5 people are needed to address 85% of the usability problem [25]. The experiment on this paper was conducted to 2 groups of 5 people (total 10 people). The first group consists of average users (can operate computers, send email, browsing, work with word processors and spreadsheets), and the rest consists of expert users (can write a program, install OS). The chosen 10 people were told to try the 2FYSH authentication in desktop and mobile. For comparison, we also asked them to try the existing solution on authentication, FIDO U2F [18]. The test participants were given the following task: (1) Listen on how to do 2FYSH registration, (2) Do the 2FYSH registration, (3) Listen on how to do 2FYSH authentication, (4) Do the 2FYSH authentication (first), (5) Do the 2FYSH authentication (second), (6) Listen on how to do 2FYSH mobile authentication, (7) Do the 2FYSH mobile authentication, (8) Listen on how to do U2F registration, (9) Do the U2F registration, (10) Listen on how to do U2F authentication, (11) Do the U2F authentication (first), (12) Do the U2F authentication (second).

Each person completed the usability test by 20-30 minutes. Each task was noted on how much time spent to complete one task. The interviewer also observed the errors occurred during the completion of the task. Based on the usability experiment conducted, here are some findings: The average time spent to complete the authentication process decreases from the first to the second try. For average users, they only require 16.2 seconds for desktop authentication and 11.5 seconds for mobile authentication. Authentication with token is generally harder for average users than expert users. However, it in happens not only 2FYSH but also U2F. As users use the new authentication system, the error count decreases. This indicates that 2FYSH is relatively easy to be learnt. According to the participants, effort to authenticate with 2FYSH is at 3.1 from a 11-point Likert scale. Test participants feels that it is easy to learn using 2FYSH and u2f authentication.

Most of test participants prefer U2F when compared to 2FYSH, U2F, password. When the test participants were explained the trade-offs between each authentication system in terms of security, usability, and economy; 5 test participants choose 2FYSH, and the other 5 test participants choose password. When asked, the reasons of them not choosing U2F is because the U2F authenticator costs a lot.From the 5 test participants who chose 2FYSH, the reason they chose 2FYSH is because 2FYSH is safer and only 2 test participants state that 2FYSH is usable.To protect sensitive information, 9 out of 10 test participants prefer to use 2FYSH. There are some things that test participants do not like from 2FYSH such as problems concerning smartphones and cards. Test participants also think of it as hassles.

5.2. UDS Framework Analysis

Bonneau et al. [22] evaluated two decades worth of proposal on replacing text passwords for general-purpose user authentication on the web, resulting in a comprehensive comparative evaluation framework of web authentication schemes. The framework is also called Usability-Deployability – Security Framework (UDS Framework). In this paper, we will be using their framework to evaluate the security and deployability on our newly proposed authentication protocol. In the framework, there will be three possible answers to the benefits. “Yes” if the benefit is fully offered, “No” if the benefit is not offered at all, and “Quasi” if the system almost offers the benefit [22].

5.2.1. Security Analysis

S1 Resilient to Physical Observation [YES]: 2FYSH uses public and private key along with a challenge-response scheme to authenticate. Also, the private key is stored in the secure element (key store).

S2 Resilient to Targeted Impersonation [Quasi]: Even with the two tokens, 2FYSH almost offer this benefit. However, there is a scenario of social engineering attack where close acquaintances, friends, or even family could know the phone security and steal the phone along with the NFC card and authenticate in a relatively short time, for example: while having a toilet break. Only the average of 14.8 seconds is needed to do authentication based on our usability test.

S3 Resilient to Throttled Guessing [YES]: 2FYSH protocol uses public and private key along with a challenge-response scheme to authenticate. Beside the technical defense property of the protocol, throttled guessing cannot be done because of the requirement of two SYH factor.

S4 Resilient to Unthrottled Guessing [YES]: 2FYSH protocol uses public and private key along with a challenge-response scheme to authenticate. The attackers can deliberately send the guessed encrypted challenge along with the salt but will still likely to fail because there are a lot of combinations which can be made with asymmetric encryption.

S5 Resilient to Internal Observation [YES]: 2FYSH app stores its private key on the secure element (key store) which is safe even if the mobile phone is infected with malware.

S6 Resilient to Leaks from Other Verifiers [YES]: 2FYSH app creates a key pair for every registration process regardless the site and the username.

S7 Resilient to Phishing [YES]: 2FYSH uses public and private key along with a challenge-response scheme to authenticate and the only way to solve the challenge is to present the registered mobile phone.

S8 Resilient to Theft [Quasi]: 2FYSH offers this benefit only to the people who lock their phone with some kind of security such as PIN, pattern lock, password, face recognition, fingerprint, etc. Otherwise, 2FYSH will just act like traditional key scheme such as car keys, which if stolen, it will be vulnerable to attacks.
S9 No trusted Third Party [YES]: 2FYSH protocol needs 2FYSH server, 2FYSH app, client, and NFC card to authenticate. No third party required.
S10 Requiring Explicit Consent [YES]: 2FYSH authentication requires explicit effort to take pictures of QR code and NFC tap.

S11 Unlinkable [YES]: 2FYSH app creates a key pair for every registration process regardless the site and the username.

5.2.2. Deployability Analysis
D1 Accessible [NO]: 2FYSH needs coordinated movement on pressing the button, pointing the camera, and tapping the NFC card.
D2 Negligible Cost per User [Quasi]: The cost for the user to use 2FYSH authentication could be free/negligible since mostly the card is provided by the third party such as universities, companies, and banks. That is provided only if the users' phone has already equipped with NFC chip in it.
D3 Server Compatible [NO]: 2FYSH uses specialized scheme on authentication. To implement, a 2FYSH server needs to be coded to accept the special parameters as well as the database structures.
D4 Browser Compatible [YES]: 2FYSH protocol can be run on the standard browser which supports HTML 5 with Javascript enabled.
D5 Mature [NO]: 2FYSH protocol is newly developed, there has not been any implementation on a real web/app.
D6 Non-Proprietary [YES]: Free and an open protocol.

5.2.3 Usability Analysis
U1 Memory-wise Effortless [YES/Quasi]: Quasi memory-wise effortless if a form of SYK (e.g.: PIN, password) authentication is used on the phone (Type of everyday used of SYK counts as quasi). Totally memory-wise effortless if SYA (e.g.: finger print, face recognition) authentication is used on the phone.
U2 Scalable for Users [YES]: 2FYSH is designed to accommodate this feature. Increasing number of credentials require the exact same step to authenticate.
U3 Nothing to Carry [NO]: 2FYSH is token based, so this benefit is not offered.
U4 Physically Effortless [NO]: 2FYSH protocol requires user effort for phone unlocking, camera aiming, and card tapping
U5 Easy to Learn [YES]: Based on the experiment conducted on this paper, 2FYSH protocol quite easy to learn.
U6 Efficient to Use [YES]: Based on the experiment conducted on this paper, 2FYSH is efficient to use.
U7 Infrequent Errors [Quasi]: Based on the experiment conducted on this paper, 12.5% of users suffer from user technical errors (such as delay in QR scanning or NFC card tapping). The errors only effects in delay in the authentication process and do not imply failures of the authentication.
U8 Easy Recovery from Loss [NO]: The design of 2FYSH does not support to recover from loss easily. The loss of NFC card will need the third-party card issuer to reissue the card and the loss of registered phone will need the relying entity to revoke the old phone authority and register it to the new one.

5.3. Omitting the Use of Username
By completely omitting the use of username for the user, user will never need to type anything to register and authenticate anymore. The current proposed 2FYSH protocol does not specify how to achieve this yet but, by letting 2FYSH app reads the app ID and determines the corresponding credential, the concept can be achieved. So, in order to authenticate, the user will just need to take the phone, scan the 2D visual code, and tap the NFC card. However, note that the user may need to be prompted on which credential to be chosen when there are more

than one credentials in a single app ID on the 2FYSH app. Omitting the use of username will definitely strengthen the usability aspect of 2FYSH but weaken the security since username is also one form of secret.

6. Conclusion

According to our experiment, survey, and analysis, 2FYSH is secure yet usable. 2FYSH protects the user from usual password attacks such as man-in-the-middle attack, phishing, eavesdropping, brute forcing, shoulder surfing, key logging, and verifier leaking. 2FYSH is generally fast for authentication with a high success rate. For average users, they only require 16.2 seconds for desktop authentication and 11.5 seconds for mobile authentication. 2FYSH is easy to learn and prefered to protect sensitive information over U2F token and passwords. This fact makes 2FYSH best applied to secure sensitive data needs such as bank accounts and corporate secrets. However, there are some things that test participants do not like and possible for future works such as the need of using 2 tokens (smartphone and card) and the method to authenticate which still considered by some as a hassle.

References

[1] Hoonakker P, Bornoe N, Carayon P. *Password Authentication from a Human Factors Perspective: Results of a Survey among End-Users*. Proceedings of the Human Factors and Ergonomics Society Annual Meeting. 2009; 53: 459-463.
[2] National Institute of Standards and Technology. SP 800-63-2. Electronic Authentication Guideline. NIST; 2013.
[3] D Dasgupta, A Roy, A Nag. Multi-Factor Authentication. *Advances in User Authentication*. Cham: Springer International Publishing. 2017:185–233.
[4] D Dasgupta, A Roy, A Nag. Toward the design of adaptive selection strategies for multi-factor authentication. *Computers & Security*. 2016;63:85–116.
[5] Y Shah, V Choyi, L Subramanian. *Multi-factor Authentication as a Service*. 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, San Francisco, CA, USA. 2015:144–150.
[6] C Mulliner, R Borgaonkar, P Stewin, J P Seifert. SMS-Based One-Time Passwords: Attacks and Defense. *Detection of Intrusions and Malware, and Vulnerability Assessment*. Berlin, Heidelberg: Springer Berlin Heidelberg. 2013;7967:150–159.
[7] G Varshney, M Misra, P Atrey. Secure Authentication Scheme to Thwart RT MITM, CR MITM and Malicious Browser Extension Based Phishing Attacks. *Journal of Information Security and Applications*. 2018; 42:1-17.
[8] Jingai Xu, Jiang Qi, Xi Ye. *OTP Bidirectional Authentication Scheme Based on MAC Address*. 2nd IEEE International Conference on Computer and Communications (ICCC). 2016:1148–1152.
[9] A K Mohan, T Gireesh Kumar. Secure Seed-Based Sturdy OTP via Convenient Carry-on Device. *Artificial Intelligence and Evolutionary Algorithms in Engineering Systems*. New Delhi: Springer India, 2015; 324:447–455.
[10] *ReelPhish: A Real-Time Two-Factor Phishing Tool n.d FireEye Inc*. https://www.fireeye.com/blog/threat-research/2018/02/reelphish-real-time-two-factor-phishing-tool.html. (Accessed Feb 15, 2019).
[11] Urien P. *Cloud of Secure Elements: an Infrastructure for the Trust of Mobile NFC Services*. IEEE 10th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob). 2014:213-218.
[12] Barlian H P, Heru Nurwarsito, Wijaya Kurniawan. One-Time Password Implementation on Lego Mindstorms NXT. *TELKOMNIKA*. 2014; 12(3): 689-694.
[13] Haller Neil, Craig Metz, Phil Nesser, Mike Straw. *A One-Time Password System*. No. RFC 2289. 1998.
[14] *Two Factor Authentication | RSA SecurID Software Tokens*. n.d. RSA.Com. http://www.rsa.com/en-us/products/rsa-securid-suite/rsa-securid-access/securid-software-tokens. (Accessed July 31, 2018)
[15] *Google Authenticator - Apps on Google Play.* n.d. https://play.google.com/store/apps/details?id=com.google.android.apps.authenticator2&hl=en. (Accessed July 31, 2018)
[16] Rodrigues B, A Chaudhari, S More. *Two Factor Verification Using QR-Code: A Unique Authentication System for Android Smartphone Users*. 2nd International Conference on Contemporary Computing and Informatics (IC3I). .2016: 457–462.
[17] Aloul F, S Zahidi, W El-Hajj. *Two Factor Authentication Using Mobile Phones*. IEEE/ACS International Conference on Computer Systems and Applications. 2009: 641–644.

[18] Juan Lang, Alexei Czeskis, Dirk Balfanz, Marius Schilder, Sampath Srinivas. *Security Keys: Practical Cryptographic Second Factors for the Modern Web*. International Conference on Financial Cryptography and Data Security. Berlin, Heidelberg. 2016: 422-440.

[19] S Pranata, H T Nugroho, H Yamaki. Analisis dan Implementasi Protokol Otentikasi FIDO U2F. *Jurnal Ultima Computing*. 2017; 9(1): 30-35.

[20] Stajano, Frank. *Pico: No More Passwords!.* International Workshop on Security Protocols. 2011. Berlin, Heidelberg. 2011: 49-81.

[21] Zhu B, Fan X, Gong G. *Loxin: a Solution to Password-less Universal Login*. IEEE Computer Communications Workshops. 2014: 488–93.

[22] Bonneau, J cormac Herley, Paul C van Oorschot, Frank Stajano. *The Quest to Replace Passwords: a Framework for Comparative Evaluation of Web Authentication Schemes*. IEEE Symposium on Security and Privacy. Oakland. 2012:553-567.

[23] *Android Keystore System* n.d. https://developer.android.com/training/articles/keystore.html (accessed September 10, 2017).

[24] S S Kumbhar, Y Lee, J Yang. *Hybrid Encryption for Securing SharedPreferences of Android Applications*. 1st International Conference on Data Intelligence and Security (ICDIS). South Padre Island, TX. 2018: 246–249.

[25] Knapp J, Zeratsky J, Kowitz B. Sprint: How to Solve Big Problems and Test New Ideas in Just Five Days. New York: Simon and Schuster. 2016.