WifiOTP: Pervasive Two-Factor Authentication Using Wi-Fi SSID Broadcasts

Huseynov, Emin; Seigneur, Jean-Marc

WIFIOTP: PERVASIVE TWO-FACTOR AUTHENTICATION USING WI-FI SSID BROADCASTS

Emin Huseynov, Jean-Marc Seigneur

University of Geneva

ABSTRACT

Two-factor authentication can significantly reduce risks of compromised accounts by protecting from weak passwords, online identity theft and other online fraud. This paper presents a new easy solution to implement two-factor authentication without affecting user experience by introducing minimum user interaction based on standard Wi-Fi. It has been validated with different software and hardware implementations in a real life environment to show it can easily be deployed in many cases.

Keywords— user-friendly security, multi-factor authentication

1. INTRODUCTION

Traditional two-factor authentication solutions use standalone hardware or software tokens (often isolated from the primary system, e.g. a mobile application running on a smartphone) that generate one-time passwords (OTP) for the second step of the login process [1]. Users need to transfer these OTPs to the primary system to complete the process. In most of the cases (with a few exceptions described in Section 2), users need to type the OTP manually. This introduces a certain level of inconvenience that leads to negative user experience and ultimately user resistance.

WifiOTP is a concept of simplifying user interaction with systems requiring two-factor authentication by eliminating the need of typing OTPs manually; instead, a special device (WifiOTP Token) will generate and broadcast OTP as a part of wireless network service set identifiers (SSID). This SSID will contain a system ID (a prefix to distinguish between other SSIDs) and an OTP part encrypted with a symmetric algorithm.

In this paper we present the concept on WifiOTP. In Section 2, we discuss related work. Section 3 presents the model of our solution and Section 4 shows how we have validated it with different software and hardware implementations to show it can easily be deployed in many cases. We conclude in Section 5.

2. RELATED WORK

We have reviewed a number of research and commercial products in the same or similar area. We believe the examples below have limited success in reaching the balance of strong security and minimal user interaction.

2.1. Google Authenticator

As our proof-of-concept will be a “drop-in” replacement of popular strong security systems, we review one of the most commonly used two-factor authentication systems, Google Authenticator, a mobile application concept used to secure services provided by Google, as well as many other services [2]. The application generates one-time passwords (OTP) by calculating a hash based on a secret shared key (known to mobile application and the authentication server) and the current timestamp with a 30 seconds modulo as defined in RFC 6238, Time-Based One-Time Password Algorithm (TOTP) [14].

As per described procedures, the authentication process assumes that user would manually type in the OTPs generated by mobile application. Obviously, this process is not very user friendly, and we will attempt to minimize user interaction by keeping the same security level.

2.2. Zero Interaction Authentication

Zero Interaction Authentication (ZIA) is one example of an effort for making security easy to use. A few academic papers include concepts of using existing wireless or wired network components for ZIA. However, the proposed systems are based on actually connecting to common wireless [3] or wired [4] networks which makes it impossible to use in systems that do not allow multiple wireless connections and presents additional risks of network based attacks. These papers also use constant characteristics of network components, such as BSSID of a WLAN network or a MAC address of network routers, which could make the systems vulnerable to replay attacks.

2.3. Context-aware application and Wi-Fi proximity

A system based on WIFI SSID is proposed by Namiot [5], [6] where SSID is used as a context-aware application concept. This paper describes using the information exchanged between access points and client devices to determine proximity data and using this proximity data to send promotional information, similar to Apple's iBeacon technology but based on Wi-Fi rather than BLE (Bluetooth Low Energy). This paper uses similar concept of utilizing Wi-Fi SSID to relay information, but does not provide any security analysis as the system is not intended to be used as a security mechanism.

2.4. One-touch financial transaction authentication

SSID broadcasts based systems are researched by authors [7], where they propose to channel the authentication data via SSID, however their approach assumes two-way communication between the client and the SSID Access point. This approach is logical if the purpose is to be used in a typical online banking system where the user interacts with the second factor authentication system. In this approach, the client device sends a packet and receives a response from the system broadcasting SSID in both directions. The limitation of this method is that the client device will need to emit SSID broadcasts and not only scan for SSIDs; this method is a significant obstacle for systems using WLAN as their primary connection. This would not be required for TOTP based systems as only one-way data flow is needed for such systems.

2.5. Amigo: proximity-based authentication of mobile devices

Amigo [8] is another example of proximity authentication based on Wi-Fi proximity, which utilizes promiscuous mode for 802.11 frame packet scanning. This system requires at least three trusted devices in the close proximity (few meters), which can be considered a major drawback compared to WifiOTP which requires only one device providing Wi-Fi coverage with minimal signal strength (up to 100 meters).

2.6. BLE based OTP tokens

Another system with closer implementation is proposed [9], where TOTP broadcasts are emitted by a BLE based token beacon device. This concept may be further developed using Eddystone, an open beacon format recently announced by Google [10]. The drawbacks of these systems are: BLE is only supported on limited types of devices and also Bluetooth is usually not activated on devices as users often perceive it as a battery hog.

2.7. Authy Bluetooth

Authy Bluetooth is a TOTP based implementation of two factor authentication very similar to WifiOTP concept. As it is based on BLE protocol, the use of Authy Bluetooth [11] is limited to situations were both a client access system (e.g. a laptop) and the system running the token (e.g. a mobile phone with the Authy app) support the BLE protocol. For this reason, Authy Bluetooth is only supported on recent Mac devices as clients, iPhone 4s and above, plus Android devices running version 4.4.4 and above with BLE support as mobile device.

In addition, the current implementation can hardly be considered as a system with "minimal user interaction" as users need to: launch Authy application, select an account and after the current OTP is copied to clipboard, users are advised to paste it to the form requiring the OTP.

2.8. Yubico hardware tokens

Yubico offers a number of products appearing to be the achieving the real minimum of user's interaction required to submit second factor [12]. Yubico's Nano and Neo hardware tokens are designed to send generated OTPs via NFC or USB (emulating keyboard input). These devices are indeed making two-factor authentication much easier for end users, however there are still some disadvantages. The USB based token is impossible to use in the majority of mobile devices without additional equipment, and the activity range of NFC token is limited to 15 centimeters [13]. Furthermore, the number of accounts each token can use is limited to maximum 2 keys per device (up to 2 keys per device), whereas WifiOTP as a concept poses no such restriction.

3. OUR WIFIOTP SOLUTION

3.1. System model

The client part will be an application running on user's device that queries the WLAN network adapter for the list of currently available SSIDs and finds the one with required prefix (System ID) then decrypts the OTP part using shared key and sends as text to the relevant input fields either automatically or when requested by the user (e.g. by pressing a button or a keyboard shortcut). As a result, the final authentication credentials will contain three authentication components: the username (entered by user), password (entered by user) and OTP (automatically read from SSIDs and decrypted instantly upon activation).

The solution consists of two main components:

- WifiOTP client: a connector application or a service/daemon running on the client device that scans the broadcasted SSIDs periodically or when initiated by users

- WifiOTP token: a Wi-Fi access point running that broadcasts a periodically changing SSID that contains the encrypted one-time password together with other data (e.g. system ID etc.)

It is important to mention that WifiOTP concept does not require the clients to be connected to the detected WifiOTP network; in fact, it would be rather inconvenient as the SSID broadcasted by WifiOTP token changes periodically as mentioned above. The clients can be connected to any other network, wired or wireless, or via cellular data connections. Therefore the type of the encryption used for WifiOTP wireless network does not matter, in our implementation we created secured WLAN with randomly changing pre-shared key value.

The system's principle is illustrated in Figure 1. The device used as an access point is using a locally stored secret hash to generate OTP values and broadcasts an SSID that includes the system identifier and the current OTP. OTP values change periodically (every 30 seconds as per TOTP specification [14]), therefore the broadcasted SSID equally changes.

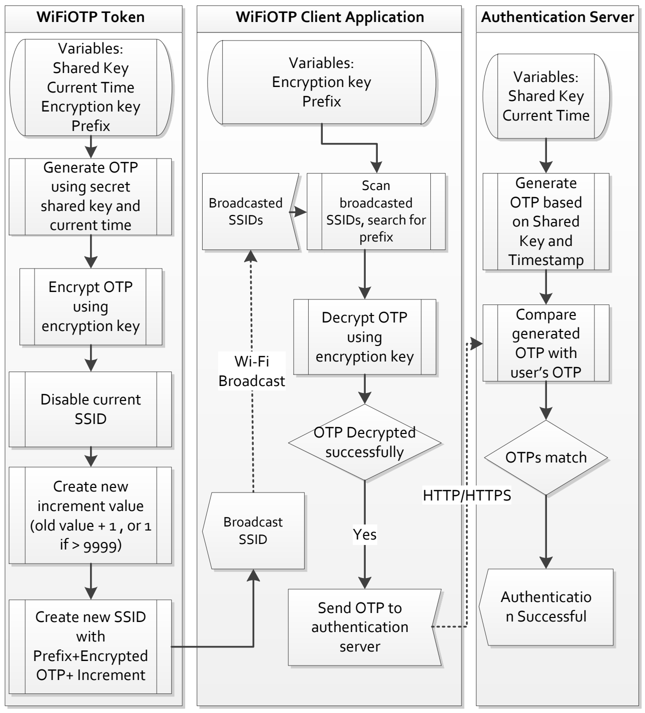

WiFiOTP Token
WiFiOTP Client Application
Authentication Server
Variables: Shared Key Current Time Encryption key Prefix
Generate OTP using secret shared key and current time
Encrypt OTP using encryption key
Disable current SSID
Create new increment value (old value + 1, or 1 if > 9999)
Create new SSID with Prefix+Encrypted OTP+ Increment
Variables: Encryption key Prefix
Broadcasted SSIDs
Scan broadcasted SSIDs, search for prefix
Decrypt OTP using encryption key
Wi-Fi Broadcast
OTP Decrypted successfully
Broadcast SSID
Yes
Send OTP to authentication server
HTTP/HTTPS
Variables: Shared Key Current Time
Generate OTP based on Shared Key and Timestamp
Compare generated OTP with user's OTP
OTPs match
Authentication Successful

Figure 1. WifiOTP Logical diagram

The format of SSID broadcasted by WifiOTP token is shown on Figure 2.

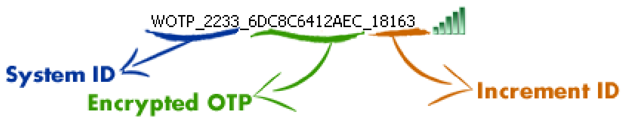


Figure 2. Format of SSIDs broadcasted by WifiOTP tokens

The connector application on the client device scans the broadcasted SSIDs periodically searching for SSIDs starting with predefined prefix (in the example above, "WOTP__") then parses the SSID name to extract the system ID and encrypted OTP. The system ID is used to distinguish between multiple WifiOTP accounts running in parallel on the same token device. The last portion of SSID, Increment ID, is required to overcome the SSID name "caching" on the client systems. The value of Increment ID will increment on every OTP change and the SSID with larger increment value will be used as the current OTP (if the increment reaches 99999, the SSID with increment equal to 1 will be considered the most current one). The client application may use a predefined API to pass the authentication data to the validating server, or just pass the parsed data to another application (i.e. a web browser) using keyboard shortcut or other methods. This data flow is illustrated in Figure3.

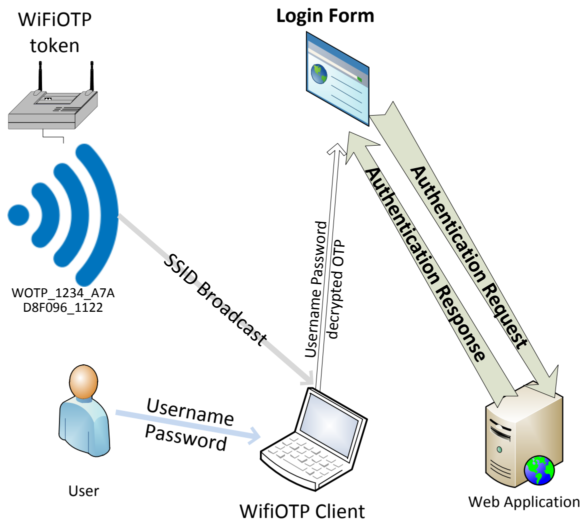


Figure 3. WifiOTP Data flow model

In this example, the current OTP (decrypted by the client application) is passed to the server together with the first authentication factor (username and password). At the final step, all submitted data is verified on the server: username and password checked for validity, and submitted OTP checked with the OTP generated on the server using the same secret hash.

3.2. One-Time password generation and encryption

As described above, we will be using TOTP as the standard for generating OTP. In principle, TOTP is a version of HOTP where current time is used as a part of secret key [14]. The value of OTP is calculated using function:

$$TOTP\ (K, T) = Truncate\ (Hash\ (K, T))$$

where:
T – The current timestamp's increment value,
K – The shared secret key (stored on the authentication server and the WifiOTP device),
Hash – a hash function (HMAC-SHA-256, HMAC-SHA-512 or other HMAC-based functions),
Truncate- a function to select a certain portion of the generated hash to be used as the OTP.
With WifiOTP, OTP is encrypted with a symmetric encryption algorithm in order to avoid transmitting current OTPs in plaintext. Due to the limitations of SSID name length (maximum 64 characters), the algorithms that can be

used for this step are also limited. The value transmitted using WifiOTP is calculated using the following function:

WifiOTPServer (K, T, E) = CiphEncr (TOTP (K, T), E)

where:
CiphEncr – a symmetric encryption function (e.g. RC4),
E - a key for encrypting the OTP (known to WifiOTP device and the client application).
The client application will scan the broadcasted networks and select one SSID matching the defined conditions (e.g. having a specific prefix and the highest increment number). Then, the OTP to be transferred to the authentication server will be calculated using the function:

WifiOTPClient (COTP,E) = CiphDecr (COTP,E)

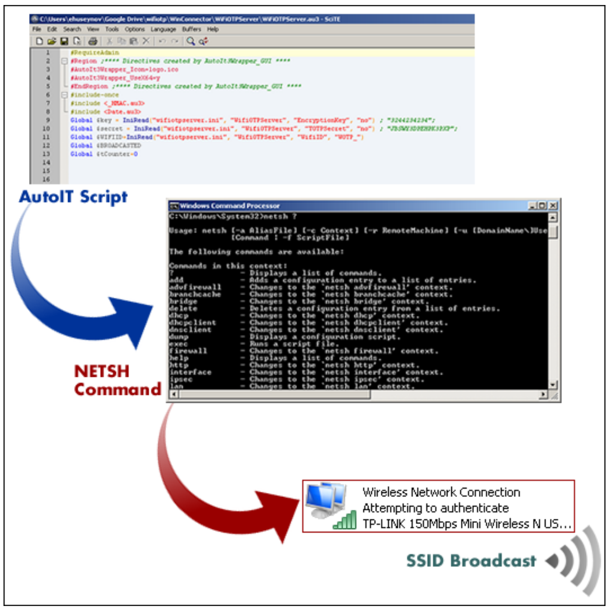
where:
COTP- ciphered value of OTP broadcasted as a part of SSID,
CiphDecr - a decryption function utilizing the same key as in WifiOTPServer function.

4. VALIDATION OF OUR WIFIOTP SOLUTION

In order to validate WifiOTP in practice we have created the following system prototypes as a proof of concept:
- WifiOTP Token: a service running on a Windows 7 computer with a wireless network card,
- WifiOTP Client:
    a) an application on Windows 7
    b) an Android application
    c) an Android custom keyboard

4.1. WifiOTP token

WifiOTP Token is the central component of the system. It periodically generates the one-time passwords based on stored secret hash key and current time and broadcasts it as a part of a wireless network name (SSID) in encrypted format.
As building or configuring a standalone WifiOTP Token device might be rather complex, in order to ease the validation a Windows application has been created to be used as a WifiOTP Token. Windows application is based on creating computer-to-computer (ad hoc) wireless network using "***netsh hostednetwork***" command. An application has been created using Autoit [15] that generates and encrypts one-time passwords and passes SSID name as an argument to netsh command. This application can run on any computer equipped with a WLAN network card and a recent Windows operating system (tested on Windows XP, Windows 7, Windows 8.x and Windows 10 Preview). The parameters, such as SSID prefix, secret shared key and encryption key are stored in an ini file.


C:\Users\ehuseynov\Google Drive\wifiotp\WinConnector\WifiOTPServer\WifiOTPServer.au3 - SciTE

```
#RequireAdmin
#Region ;**** Directives created by AutoIt3Wrapper_GUI ****
#AutoIt3Wrapper_Icon=logo.ico
#AutoIt3Wrapper_UseX64=y
#EndRegion ;**** Directives created by AutoIt3Wrapper_GUI ****
#include-once
#include <_HMAC.au3>
#include <Date.au3>
Global $key = IniRead("wifiotpserver.ini", "WifiOTPServer", "EncryptionKey", "no") ; "3244234234";
Global $secret = IniRead("wifiotpserver.ini", "WifiOTPServer", "TOTPSecret", "no") ; "JBSWY3DPEHPK3PXP";
Global $WIFIID=IniRead("wifiotpserver.ini", "WifiOTPServer", "WifiID", "WOTP_")
Global $BROADCASTED
Global $tCounter=0
```

Windows Command Processor

```
C:\Windows\System32>netsh ?

Usage: netsh [-a AliasFile] [-c Context] [-r RemoteMachine] [-u [DomainName\]Use
             [Command | -f ScriptFile]

The following commands are available:

Commands in this context:
?              - Displays a list of commands.
add            - Adds a configuration entry to a list of entries.
advfirewall    - Changes to the `netsh advfirewall' context.
branchcache    - Changes to the `netsh branchcache' context.
bridge         - Changes to the `netsh bridge' context.
delete         - Deletes a configuration entry from a list of entries.
dhcp           - Changes to the `netsh dhcp' context.
dhcpclient     - Changes to the `netsh dhcpclient' context.
dnsclient      - Changes to the `netsh dnsclient' context.
dump           - Displays a configuration script.
exec           - Runs a script file.
firewall       - Changes to the `netsh firewall' context.
help           - Displays a list of commands.
http           - Changes to the `netsh http' context.
interface      - Changes to the `netsh interface' context.
ipsec          - Changes to the `netsh ipsec' context.
lan            - Changes to the `netsh lan' context.
```

Wireless Network Connection
Attempting to authenticate
TP-LINK 150Mbps Mini Wireless N US...

Figure 4. WifiOTP Token application for Windows

4.2. WifiOTP client applications

WifiOTP Client searches for the SSID with encrypted one-time passwords broadcasted by WifiOTP Tokens. For this proof of concept, we created a Windows client application and an Android app. The user interaction model of each application is explained within the use cases section of this paper.

4.2.1. Windows application

Standard Windows netsh command is capable of scanning the SSIDs broadcasted ("***netsh wlan show networks***") [16]. The parsed SSID data needs to be decrypted and sent to input field requesting OTP, which is then submitted the validation server together with other data. A simple system daemon has been developed using Autoit [15] monitors a specific keyboard shortcut (e.g. Ctrl+Alt+X) to send currently broadcasted OTP to active text input. Optionally, it can send return character together with OTP to minimize user's interaction: e.g. when user is prompted to enter OTP in a web application, pressing Ctrl+Alt+X will insert the required OTP and submit the current form immediately. A screenshot of a running WifiOTP Client application is shown on Figure 5. The window below shows the current OTP for demonstration purposes only; the client application should run silently in the background with only an icon displaying its activity in the tray area.

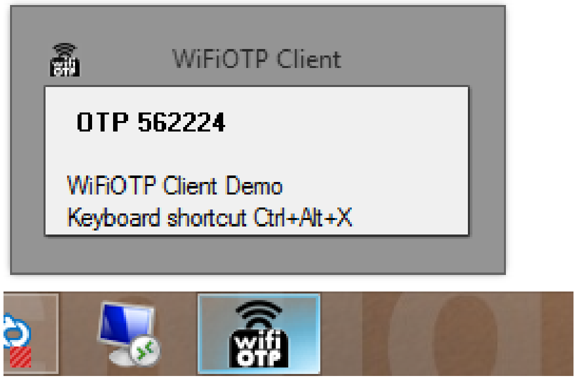


Figure 5. WifiOTP Client for Windows

4.2.2. *Android mobile application*

We decided to prototype an Android application for WifiOTP Client using PhoneGap [17] platform. A PhoneGap plugin scanning Wi-Fi networks currently in range was created for this prototype application. The broadcasted OTP is fetched by the application using the same methods as with the Windows application. The unencrypted OTP can subsequently be copied to the clipboard allowing pasting of the current OTP to a relevant field in any application (e.g. a web browser).

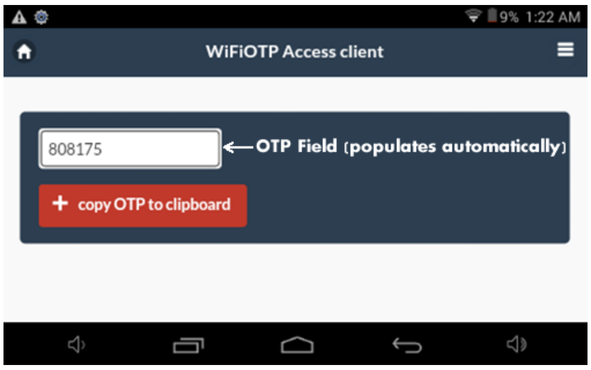


Figure 6. WifiOTP Android client. Clipboard mode

The application can also act as a web browser, and in this mode, the field requesting the OTP, is populated automatically.

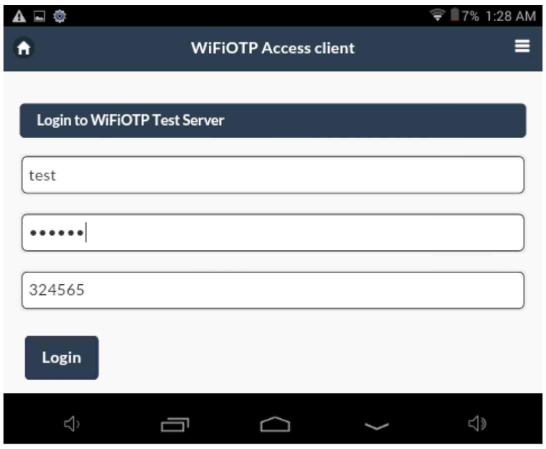


Figure 7. WifiOTP Android client. Web browser mode

4.2.3. *Android custom keyboard*

Using a separate mobile application introduces a number of restrictions, the main one being the inability to use WifiOTP with any standard application, such as web browser. To resolve this, we need a resource containing WifiOTP Client code, which would be available to any application throughout the system. Android allows developers to create custom keyboards and run any type of code associated with its keys, including scanning for available Wi-Fi networks [18].
We have created a custom keyboard, based on a sample provided within the Android developer guide [19]. The keyboard consists of two keys: the first will execute Wi-Fi scanning, parse and decrypt OTPs broadcasted by WifiOTP Token and insert the OTP to current input field, the second will delete contents of current input field. User interaction required for our initial implementation of WifiOTP Android custom keyboard is shown in Figure 8. As can be seen from the image, the user interaction for entering second factor to authenticate can be reduced to two actions: selecting WifiOTP keyboard and hitting “Insert OTP” key. This process can be simplified further to reduce the number of actions to one: this will require the keyboard to automatically send an OTP upon activation.

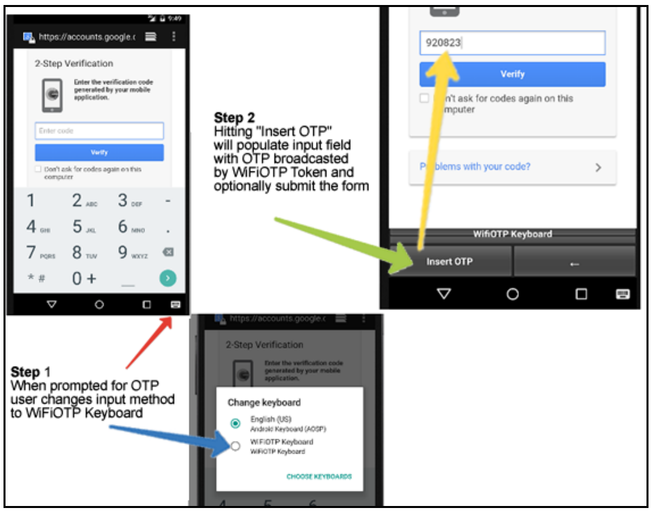

https://accounts.google.c
2-Step Verification
Enter the verification code generated by your mobile application.
Enter code
Verify
Don't ask for codes again on this computer
Step 2
Hitting "Insert OTP" will populate input field with OTP broadcasted by WiFiOTP Token and optionally submit the form
920823
Verify
n't ask for codes again on this mputer
P blems with your code?
WifiOTP Keyboard
Insert OTP
Step 1
When prompted for OTP user changes input method to WiFiOTP Keyboard
Change keyboard
English (US)
Android Keyboard (AOSP)
WiFiOTP Keyboard
WifiOTP Keyboard
CHOOSE KEYBOARDS

Figure 8. WifiOTP Android custom keyboard

Having an additional keyboard only for authentication purposes may introduce a certain level of inconvenience for users, especially for users frequently using more than one keyboard layout. To overcome this, a custom keyboard containing standard language layout and one additional key to insert OTP, can be created. With this keyboard set as default, user interaction to enter the OTP in the relevant field is reduced to pressing a key when prompted. See the example below (Figure 10) of such a keyboard based on English (US).

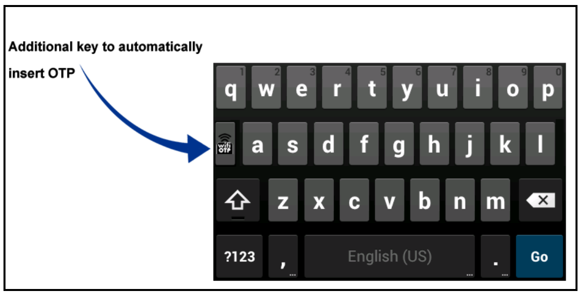


Figure 9. Custom WifiOTP keyboard based on English (US) layout

4.3. Use cases

In this section, we present two use cases to illustrate the usage of the WifiOTP in real-life scenarios. Both cases will consider logon to a web application with two-factor authentication enabled account. As a part of use case review, we will compare user experience with a classic two-factor logon process that has the following steps (assuming correct credentials are supplied):
1) User navigates to a login page
2) User enters first factor credentials (username and password)
3) User submits the logon form, either by clicking on a button or hitting Enter key on the keyboard
4) On the next window, the system asks for the second factor (one-time password), where the user manually enters the digits shown on the device or mobile application (OTP)
5) Login process completes
The flowchart of the classic process looks like shown on Figure 10.

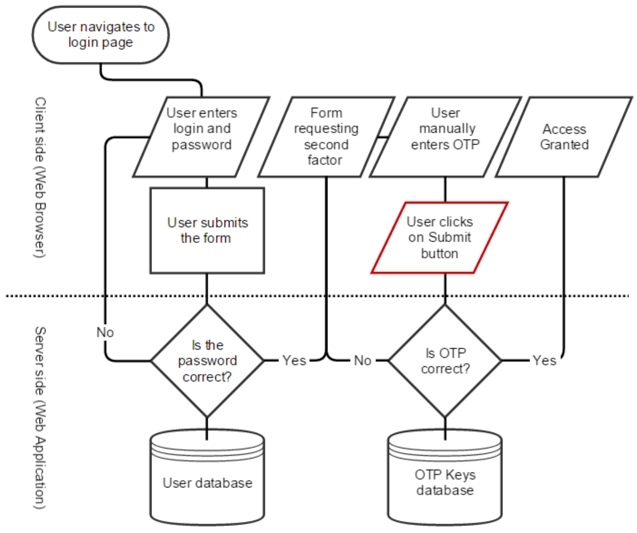

User navigates to login page
Client side (Web Browser)
User enters login and password
Form requesting second factor
User manually enters OTP
Access Granted
User submits the form
User clicks on Submit button
Server side (Web Application)
No
Is the password correct?
Yes
No
Is OTP correct?
Yes
User database
OTP Keys database

Figure 10. Classic two-factor authentication flowchart

4.3.1. Use case 1: minimal user interaction

Using WifiOTP Client for Windows is an example of this use case. Assuming a WifiOTP Token is correctly configured and active, the procedure of logging in to a standard system with two-factor authentication will consist of the following user interaction stages:
1) User navigates to login page
2) User enters first factor credentials (username and password)
3) On the next window, when the system asks for the second factor (one-time password), user presses Ctrl+Alt+X combination on the keyboard
4) Login process completes
As can be seen from the procedure, also illustrated on Figure 11, the second factor is entered automatically, with the only difference of using a specific keyboard shortcut (Ctrl+Alt+X) instead of hitting enter or clicking on Submit button.

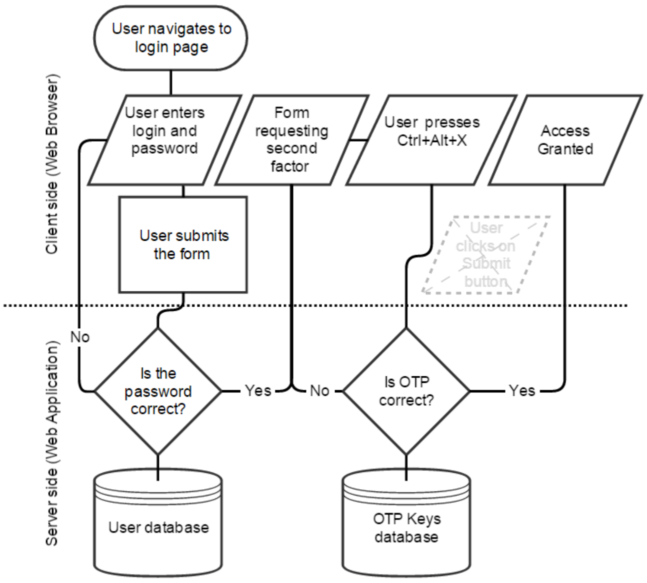

User navigates to login page
Client side (Web Browser)
User enters login and password
Form requesting second factor
User presses Ctrl+Alt+X
Access Granted
User submits the form
User clicks on Submit button
Server side (Web Application)
No
Is the password correct?
Yes
No
Is OTP correct?
Yes
User database
OTP Keys database

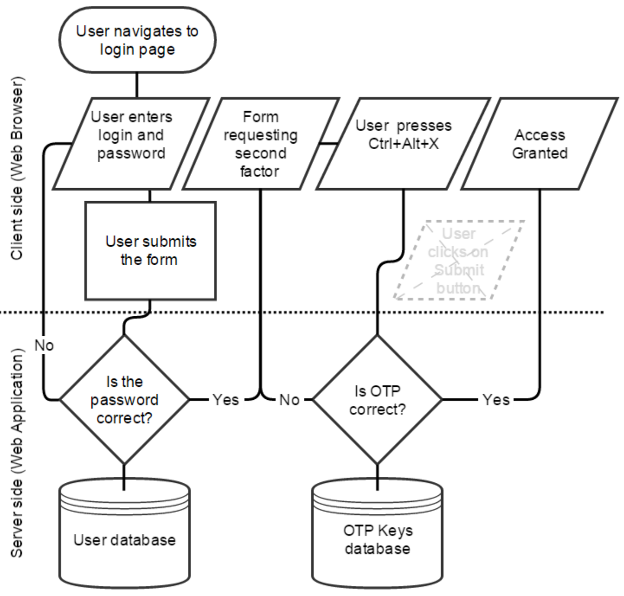
Figure 11. Two-factor authentication flowchart with WifiOTP Windows client

This use case requires no modification on the server side, thus can be used on existing systems with two-factor authentication implemented in one (where both factors are requested in the same time, e.g. on the same login form) or two steps using any standard software. This use case was successfully tested on a number of public services (including Gmail and Dropbox) using a standard web browser, as well as special applications (e.g. Google Drive). This use case is also valid for Android custom keyboard.

4.3.2. Use case 2: zero user interaction

To demonstrate this use case, we have developed an Android WifiOTP Client application that will allow zero user interaction for providing second factor during the authentication process. Procedure for this use case is as follows:
1) User launches the mobile application
2) User enters first factor credentials (username and password)
3) On the next window, when the system asks for the second factor (one-time password), the mobile application

automatically populates the relevant field with OTP and submits the form without any user interaction
4) Login process completes
The flowchart shown on Figure 12, illustrates that there are no additional actions required from the users to securely authenticate, which makes the user experience similar to one-factor authentication systems.



Client side (Web Browser)
Server side (Web Application)
User navigates to login page
User enters login and password
Form requesting second factor
User presses Ctrl+Alt+X
Access Granted
User submits the form
User clicks on Submit button
No
Is the password correct?
Yes
No
Is OTP correct?
Yes
User database
OTP Keys database

Figure 12. Zero user interaction two-factor authentication using Android mobile application

4.3.3. Summary of reviewed use cases

Both use cases demonstrated that using WifiOTP simplifies users' interaction compared to classic two-factor authentication systems. Although case 1 still requires additional user action, it has its advantages, as it can be used with existing client side applications without any modification of authentication systems. With Use case 2, user interaction is reduced to zero, but the method can only be used via a special mobile application. A detailed summary table of use cases is provided in Table 1 below.

Table 1. Use case comparison

Comparison aspect	**Classic two-factor authentication**	**Use Case 1**	**Use Case 2**
User interaction	User should manually type OTPs generated by token	Minimal (keyboard shortcut)	Zero
Software requirements	No additional application on client system.	Can be used with any application on Windows operating system	Access to systems can only be done via WifiOTP Android application
Server side web application	Any	Any	Any
WifiOTP Token	Any	Any	Any
Hardware requirements	Network access equipment (wireless, wired or cellular)	Wireless Network Card	Wireless Module

Additionally, we would like to clarify that the platforms chosen for both use cases are only examples chosen for our validation and not based on any technical restriction: i.e. Use Case 2 can be easily implemented as a Windows application and vice versa. Due to the fact that the interaction flows are similar for Windows Client and Android custom keyboard based solution, we have not reviewed it as a separate use case.
Both use cases demonstrated no interference or other negative effect to any of existing wireless or wired networks: we successfully tested the functionality on systems connected over different networks (such as Wi-Fi, wired network and cellular data network on mobile devices) as per current and proposed connectivity standards [20] [21] [22].

4.3.4. Security analysis and discussion

Generic security analysis of proposed two-factor mechanism is already done by many authors, and we will refer to existing work [23] as full analysis would be out of scope of this paper.
An additional security risk is introduced by transmitting OTPs via SSID name broadcast, which is publicly readable. This risk is still minimal even if OTPs are transmitted in plain text and equal to a situation when attackers gain access to the OTP device (e.g. a hardware token). However, even with this minimal risk, we attempted to eliminate it by introducing symmetric encryption of broadcasted OTPs using RC4 encryption algorithm. RC4 is rather weak compared to other modern cryptographic methods [24], however the limitation of the SSID length [22] does not allow many options to choose from.
Applications for other platforms (such as MacOSX, Linux and Windows Phone) could be also created. Only iOS devices can't have WifiOTP as long as API methods for scanning Wi-Fi networks are not publicly available. Future versions of iOS may allow it though.

5. CONCLUSION

User authentication is a balance of security and user experience. This paper presents the possibility of creating a simple and low-cost two-factor authentication system that simplifies user's interaction compared to existing solutions by minimizing or completely eliminating the actions required to add the second factor for authentication.
The solution proposed also presents a possibility of introducing an additional authentication factor – the physical location of the user, which we will investigate in future work.

REFERENCES

[1] F. Aloul, S. Zahidi and W. El-Hajj, "Two factor authentication using mobile phones," IEEE/ACS International Conference on Computer Systems and Applications, pp. 641 - 644, 2009.

[2] Google Inc., "Open source version of Google Authenticator," 15 June 2015. [Online]. Available: https://github.com/google/google-authenticator/. [Accessed 20 July 2015].

[3] T. Christophersen, Zero Interaction Multi-factor Authentication, Kongens Lyngby: Technical University of Denmark, 2010.

[4] M. Corner and B. Noble, "Zero-Interaction Authentication," in SIGMobile, Ann Arbor, MI, 2002.

[5] D. Namiot, Network Proximity on Practice: Context-aware Applications and Wi-Fi Proximity, Moscow: International Journal of Open Information Technologies, 2013.

[6] D. Namiot, "Wi-Fi Proximity as a Service," SMART 2012: The First International Conference on Smart Systems, Devices and Technologies, pp. 62-68, 2012.

[7] D. V. Bailey, J. G. Brainard, S. Rohde and C. Paar, One-touch Financial Transaction Authentication, Bochum: SECRYPT, 2009.

[8] A. Varshavsky, A. Scannell, A. LaMarca and E. de Lara, "Amigo: Proximity-Based Authentication of Mobile Devices," UbiComp 2007: Ubiquitous Computing, pp. 253-270, 2007.

[9] R. van Rijswijk-Deij, Simple Location-Based One-time Passwords, Utrecht: Technical Paper, 2010.

[10] Google Inc., "Eddystone™, the open beacon format from Google," 15 July 2015. [Online]. Available: https://developers.google.com/beacons/. [Accessed 20 July 2015].

[11] Authy, "Authy | The Future," 22 04 2015. [Online]. Available: https://www.authy.com/thefuture#bluetooth.

[12] Yubico Inc., "YUBIKEY STANDARD & NANO," 2015. [Online]. Available: https://www.yubico.com/products/yubikey-hardware/yubikey-2/.

[13] R. Want, "Near Field Communication," IEEE Pervasive Computing vol.10, no. 3, pp. 4-7, 2011.

[14] D. M'Raihi, S. Machani, M. Pei and J. Rydell, "TOTP: Time-Based One-Time Password Algorithm," May 2011. [Online]. Available: http://www.rfc-editor.org/info/rfc6238.

[15] Autoit Consulting, "AutoIt : Overview," [Online]. Available: https://www.autoitscript.com/site/autoit/. [Accessed 15 05 2015].

[16] Microsoft, "Netsh Command Reference," 2 07 2002. [Online]. Available: https://technet.microsoft.com/en-us/library/cc754516%28v=ws.10%29.aspx. [Accessed 27 05 2015].

[17] Y. Patel and R. Ghatol, Beginning PhoneGap: Mobile Web Framework for JavaScript and HTML5, New York: Apress, 2012.

[18] Google Inc., "Android Developer Guides: Creating an Input Method," 26 03 2015. [Online]. Available: http://developer.android.com/guide/topics/text/creating-input-method.html. [Accessed 28 05 2015].

[19] T. G. Takaoka and Google Inc., "Android samples: SoftKeyboard," 15 10 2014. [Online]. Available: https://android.googlesource.com/platform/development/+/master/samples/SoftKeyboard/. [Accessed 28 05 2015].

[20] IEEE, 802.1X-2004 - IEEE Standard for Local and Metropolitan Area Networks, IEEE, 2004.

[21] R. Valmikam, "EAP Attributes for Wi-Fi - EPC Integration," 5 01 2015. [Online]. Available: https://tools.ietf.org/html/draft-ietf-netext-Wifi-epc-eap-attributes-16. [Accessed 26 05 2015].

[22] IEEE, 802.11 standard for LAN/MAN, 2012.

[23] A. Dimitrienko, C. Liebschen and C. Rossow, "Security Analysis of Mobile Two-Factor Authentication Schemes," Intel Security Journal, vol. 18, no. 4, pp. 138-161, 2014.

[24] S. Fluhrer, M. Itsik and S. Adi, Weaknesses in the key scheduling algorithm of RC4, Berlin: Springer , 2001.

[25] D. Tavares, M. Lima, R. Aroca, G. Caurin, A. C. de Oliveira Jr, T. Santos Filho, S. Bachega, M. Bachega and S. da Silva, "Access Point Reconfiguration Using OpenWrt," in Proceedings of The 2014 World Congress in Computer Science, Computer Engineering, Las Vegas, 2014.

[26] R. Swan, "SIMPLE OATH TOTP RFC 6238 IN PHP," 09 05 2013. [Online]. Available: http://www.opendoorinternet.co.uk/news/2013/05/09/simple-totp-rfc-6238-in-php. [Accessed 15 05 2015].

[27] OpenWRT, "Nexx WT3020," [Online]. Available: http://wiki.openwrt.org/toh/nexx/wt3020. [Accessed 15 05 2015].

[28] D. Howett, "iPhone Development Wiki: MobileWifi.framework," 12 07 2014. [Online]. Available: http://www.iphonedevwiki.net/index.php/MobileWifi.framework. [Accessed 28 05 2015].