Hardware TOTP tokens with time synchronization

Huseynov, Emin; Seigneur, Jean-Marc

How to cite

HUSEYNOV, Emin, SEIGNEUR, Jean-Marc. Hardware TOTP tokens with time synchronization. In: 2019 IEEE 13th International Conference on Application of Information and Communication Technologies (AICT). Baku, Azerbaijan. [s.l.] : [s.n.], 2019. doi: 10.1109/AICT47866.2019.8981762

This publication URL: https://archive-ouverte.unige.ch/unige:125190

Publication DOI: 10.1109/AICT47866.2019.8981762

Hardware TOTP tokens with time synchronization

Emin Huseynov
Faculté des Sciences de la Société
University of Geneva
Geneva, Switzerland
emin@huseynov.com

Jean-Marc Seigneur
CUI, ISS & Medi@LAB, Faculté des Sciences de la Société
University of Geneva
Geneva, Switzerland
seigneuj@gmail.com

***Abstract*—** Although many users nowadays prefer to use mobile applications for TOTP, there are still use cases for separate/isolated standalone hardware TOTP devices. For various technical restrictions, such as the accuracy limits of oscillators used in real-time clock chips, classic TOTP hardware tokens have a natural tendency to introduce time-drift after some period. This paper describes an innovative product allowing to overcome this disadvantage by introducing a new category of programmable TOTP hardware tokens allowing adjusting the system hardware clock.

***Index Terms*—** digital identification, identity management, authentication, authorization, privacy, context-based authentication, strong security, one-time password, TOTP, time drift.

I. Introduction

As per the RFC, the TOTP-based authentication mechanisms [1] expect hardware tokens to have real-time clocking capacity by implanting an oscillator in the device. A token’s clock drift requires to be considered and adjusted respectively by the server. The protocol's recommendation also suggests the server to implement “look-ahead” and “look-behind” steps for resynchronization when an adequate amount of clock skews have occurred on the hardware token.

While this is not as significant when mobile authenticator apps are used for the second factor, the system clock devices used in TOTP hardware tokens have a general bent to begin to time-drift after a certain amount of time. This a recognized weakness of most oscillator-based real-time clock (RTC) chips, also mentioned in the RFC 6238#6 [1], and consequently, the authentication systems must be able to automatically adjust the potential time-drift with TOTP tokens to minimize any possible influence on the users' authentication process, which may eventually lead to access being denied for this reason. While the majority of the systems with two-factor authentication can follow this RFC recommendation, there are still multiple implementations where this aspect is ignored.

In this paper, we review the potential issues that can be caused by the time drift effect of hardware tokens (section II). Later, in section IV, we propose a solution to compensate the time drift and in section V we analyze the security risks of implementing the time sync as well as the mitigation methods. Section VI reviews the implementation of a hardware token with time sync feature. We conclude the paper in Section VII.

II. Problem statement

The issue of time-drift is not as important when the second factor is based on mobile applications running on smartphones, the time synchronization takes place via the Internet or cellular networks [2]. This is probably the reason why there are several popular services that ignore the TOTP RFC recommendation of automatically adjusting the time drift.

However, there are still use cases for preferring standalone hardware tokens over mobile app-based authenticators. These use cases include, but not limited to:

- Users not possessing a corporate smartphone, or not willing to use their personal devices for authentication
- Policies or conditions are not allowing to use smartphones on-premises (i.e. for military bases, intelligence service offices or other locations with higher security standards on carried electronic devices)
- Enterprises considering applications running on smartphones insecure due to the risks of viruses, device compromisation or similar.

With hardware tokens, the potential side-effects of time drifts and its significance are different. The common time drift for modern TOTP hardware tokens statistically averages to 2 minutes per year or even more. [3]

After a couple of years, numerous tokens will introduce a time drift falling outside of the global synchronization window. While the accuracy of RTC chips has not changed a lot for the last 20 years, the battery capacity of modern tokens is now higher, making the tokens good enough to work for over 5 years, which means that after its 4th year, even though the battery is still good to allow using the token for another year, the token cannot be re-enrolled to a different authentication system as its time drift would be almost 10 minutes and very few systems would support such big disparity. This allows saying that RFC 6238 №6 is an absolute requirement for authentication mechanisms if they want to leverage hardware tokens as one of their second-factor contexts. Unfortunately, many systems are ignoring this recommendation, for example, Cisco Duo [6]:

> *“TOTP token drift and resynchronization are not supported. As a result, imported TOTP tokens may not work for authentication with Duo Security, or may fail to work for authentication after a variable period of time.”*

But even in the case, the time drift adjustment is correctly handled, a token that is not utilized very frequently is anticipated to drift even further ahead of the synchronization range that an authentication system may tolerate. Besides, businesses are hesitant to retain a big stock of physical tokens: a token that is not used at all will have its battery nearly at its full capacity, but the inaccuracy of the RTC chip will not permit even enrolling it, which makes before-mentioned expenses to be entirely wasted.

While TOTP as an authentication mechanism is secure enough and incredibly easy to realize, the problem outlined above is regarded as one of the main limitations when it comes to using hardware tokens.

III. Review of existing solutions

An academic review, as well as market research, were conducted at the time of writing and no commercial solutions nor concepts addressing this issue were identified. A preliminary version of this paper was published in 2018 by the authors to serve as a proof of novelty [7].

Programmable hardware tokens exist for a few years now and the main concept is based on the possibility of transferring a shared secret (seed) via contactless protocols such as NFC [4] but without the possibility of adjusting the system clock of the devices.

Examples of such programmable hardware tokens are Protectimus Slim Mini [8] and HyperSecu Edge.

The same concept is widely used in a few devices in many industries [5].

IV. Proposed solution

To address the issue above we are proposing a solution that would allow syncing the clock of hardware tokens using a special API via NFC protocol. The proposed device will contain a processing chip with hardware clock control feature which will accept a timestamp value transferred as a part of SDK API argument. The seed will be stored to a write-only accessible memory segment to ensure the security of the shared secret: the API will only allow writing the seed and will never allow reading the seed.

A schematic illustration of the device is shown in Figure 1.

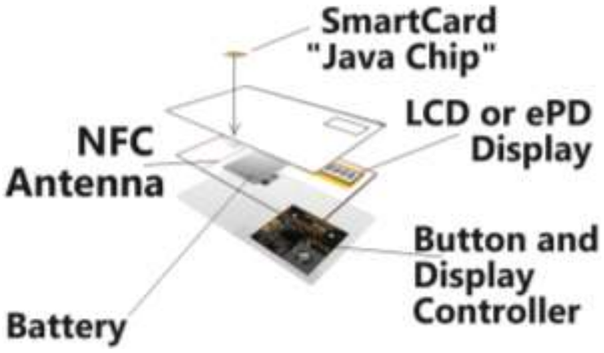


Figure 1. Programmable hardware token

The procedure of transmitting the current timestamp as well as the seed to be used for generating one-time passwords using TOTP algorithm will be done via NFC using standard NFC transmitter modules, such as the NFC chip in modern Android phones or laptops running under Windows. A wide range of ready libraries exist under different platforms (for example for Android [10]) allowing to rapidly develop prototype apps. As described later in Section V, an Android application and a Windows application is to be developed. The application will send and receive specially formed commands over NFC to perform operations defined in the API.

V. Security analysis

This section analyzes the security aspects of the proposed solution in the context of a classic two-factor authentication, where the first factor is static (i.e. username and password) and the second factor is dynamically changing – where the changing counter is time-based, namely, with Time-based One-Time passwords (TOTP) algorithm.

Adjusting the time on a hardware token is not as simplistic as setting a wristwatch: there is a possible security danger if it is only the system time of the RTC chip that is being modified. The replay attack can be described this way: assume a user currently under a targeted attack initiative and the intruder can physically reach the hardware token, yet for a short time. If only the RTC modification is allowed, the attackers can set the clock to a future time and record a significant amount of OTP codes. This can be considered as a partial compromisation of the second factor and is almost equal to attackers getting access to token's seed. With this list of OTP codes obtained, the attackers only require the first factor, e.g. the password, which is much more straightforward to obtain (i.e. via a phishing attack or a keylogger installed) and subsequently, when the time of validity of the OTP codes arrives, all factors become compromised. To solve the TOTP code replay attack described above, we propose the time sync procedure to be combined with resetting the seed of the token. So, the system time of an RTC chip will automatically erase the stored seed and therefore, eliminate the code replay attack vector.

The fact that the seed can only be set and never read from our programmable tokens will make sure the seed is only accessible by the authentication server. Therefore, unauthorized access to the time adjustment of the hardware tokens will not result in the replay attack. Contrary to this, if the time setting is set by a legitimate user (i.e. the administrator), the seed set together with the correct time value will also be set at the authentication server, or vice-versa, a new seed will be requested to be generated by the authentication server to be written to the token together with time synchronization.

Meantime, we would still like to mention that the chance of before-mentioned attack type is insignificant and can be accomplished only if all of the following circumstances are satisfied:

- The username and the password of the user being attacked have already been compromised
- Attackers can physically access the hardware device serving as the second authentication factor
- They can connect to the device via NFC for a significant amount of time. This time is required to adjust the RTC clock, set it to a future time and generate a list of OTP codes.

Those requirements are almost impossible to be satisfied and can be matched to a condition where a hardware token is stolen.

VI. Implementation

Hardware TOTP tokens with time sync have been manufactured based on the algorithms and specifications provided by the authors of this paper by a number of factories. The final product[1] is shown in Figure 2

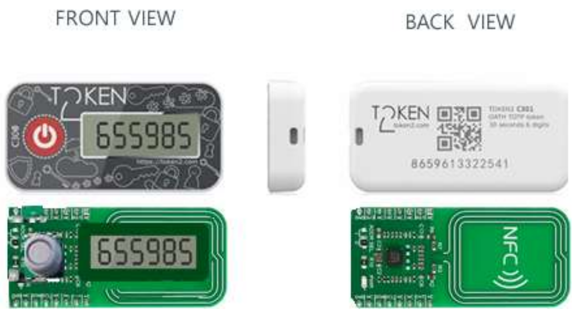


Figure 2 . C301 TOTP hardware token

The time synchronization feature is implemented via a special API which allows communicating with the device via NFC protocol. Schematic visualization of the process is shown in Figure 3.

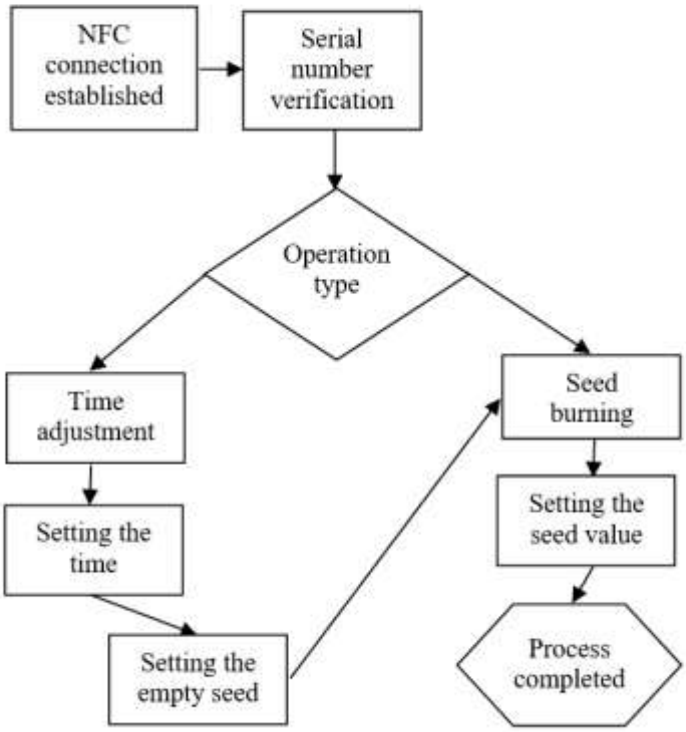


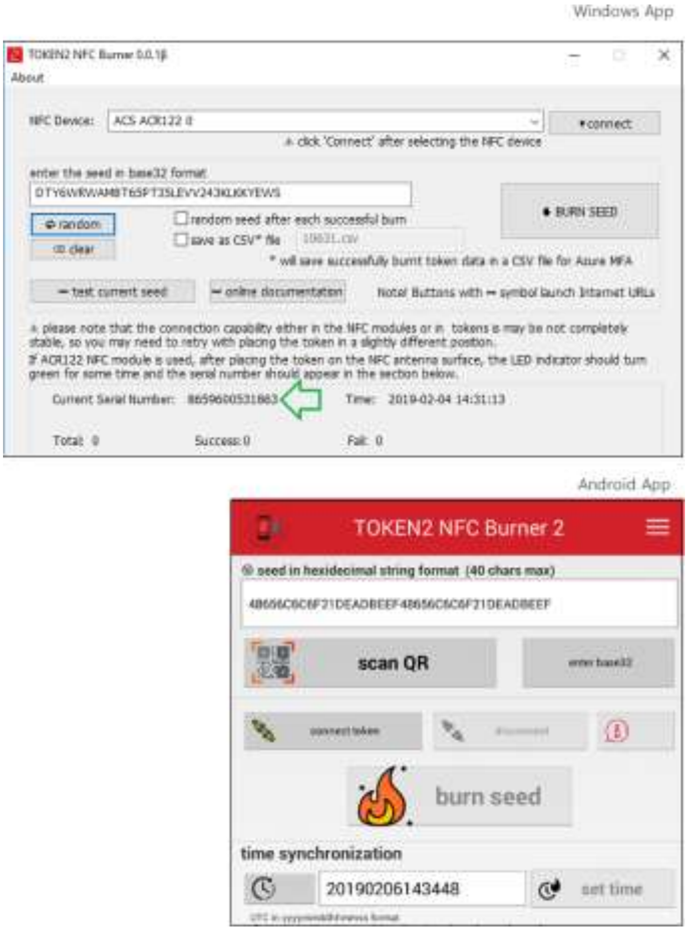
Figure 3. API Flowchart visualization

As described in V, API will automatically launch an empty seed burning process if the time of the device is set to be modified in order to avoid any potential risk of a replay attack.

The API implemented will consist of the following functions:

ReadSnAndTime char* pszR ea derName, char* pSn, int* pSnLen, char* pTime, int* pTimeLen)
Function : gets the serial number and current time from the token;
SetTime (char* pszReaderName, char* custom key , char* pszTime)
Function : sets the time of the token; pszTime [in]: time to set to the token, the format should be “yyyy/mm/dd/hh/mm/” – in UTC Timezone
BurnSeed (char* pszR ea derName, char* custom key , hexSeed int nSeedLen)
Function: burns a seed to the token; hexSeed [in]: the seed value, it should be hexadecimal format; nSeedLen[in]: length of hexSeed, it should be less than 32 bytes.

A Windows and Android applications to serve as a graphical user interface to the above-mentioned API have also been developed.



NFC Device: ACS ACR122 0

click 'Connect' after selecting the NFC device

enter the seed in base32 format: DTY6WRWAMBT65PT35LEVV243KLKKYEWS

□ random seed after each successful burn

□ save as CSV* file 10631.csv

* will save successfully burnt token data in a CSV file for Azure MFA

Note! Buttons with ➡ symbol launch Internet URLs

please note that the connection capability either in the NFC modules or in tokens is may be not completely stable, so you may need to retry with placing the token in a slightly different position.

If ACR122 NFC module is used, after placing the token on the NFC antenna surface, the LED indicator should turn green for some time and the serial number should appear in the section below.

Current Serial Number: 8659600531863 Time: 2019-02-04 14:31:13

Total: 0 Success: 0 Fail: 0

seed in hexidecimal string format (40 chars max)

48656C6C6F21DEADBEEF48656C6C6F21DEADBEEF

time synchronization

20190206143448

UTC in yyyymmddhhmmss format

Figure 3. NFC Burner applications for Windows and Android

[1] As the product is already at its commercialization phase the exact chip models and implementation methods are not disclosed as a part of this paper.

The main use case of the programmable hardware tokens is serving as a drop-in replacement of mobile authenticator apps. Therefore, the Android app has an additional feature for scanning the QR code shown for creating TOTP profile with such mobile apps (for example Google Authenticator). The QR code will then be parsed to extract the seed to be used for burning into the hardware token.

VII. Conclusion and future work

In this paper, we have analyzed the problem of a time drift occurred in modern TOTP hardware tokens and reviewed the solution and its real-life implementation. A security analysis of the potential replay attack utilizing the time sync was carried out and the commercial implementation in two different methods were studied as follows.

a) ***unrestricted time sync***
where time can be set keeping the current seed; and

b) ***restricted time sync***
where setting the time will automatically clear the seed for security purposes to avoid the risk of a replay attack.

This product can be used as drop-in replacement of mobile authenticator apps in cases where only one TOTP profile is needed; for most of the users, there are usually multiple TOTP profiles needed (i.e. one per service or authentication server etc.) . This means that multiple hardware tokens would have to be enrolled and used by users, which is quite inconvinienct. A device capable to hold multiple TOTP profiles/seeds would be a solution to this. Such a device, multi-profile TOTP hardware token, is currently being researched and will soon be presented as a separate academic work.

VIII. References

[1] J. Rydell, P. Mingliang and M. Salah, *TOTP: Time-Based One-Time Password Algorithm*, IETF, 2011.

[2] B. Sterzbach, "GPS-based clock synchronization in a mobile, distributed real-time system," *Real-Time Systems* , vol. 1, no. 12, pp. 63-75, 1997.

[3] C. A. Latha and H. L. Shashidhara, "Clock synchronization in distributed systems," in *5th International Conference on Industrial and Information Systems*, 2010.

[4] TOKEN2, "Programmable hardware tokens," 2018. [Online]. Available: https://www.token2.com/shop/category/programmable-tokens. [Accessed 15 05 2019].

[5] M. Bhattacharyya, W. Gruenwald and B. Dusch, "A RFID/NFC based Programmable SOC for biomedical applications," in *2014 International SoC Design Conference (ISOCC)*, IEEE, 2014.

[6] Duo, "Does Duo support HOTP or TOTP tokens?," Duo, 01 01 2018. [Online]. Available: https://help.duo.com/s/article/2112?language=en_US. [Accessed 15 05 2019].

[7] TOKEN2, "Time drift: a major downside of TOTP hardware tokens" Medium, 07 12 2018. [Online]. Available: https://medium.com/@token2/time-drift-a-major-downside-of-totp-hardware-tokens-c164c2ec9252 [Accessed 15 07 2019].

[8] Protectimus.com. Protectimus Slim NFC - Programmable Hardware OTP Token. ,2019. [Online] Available at: https://www.protectimus.com/slim-mini/ [Accessed 24 Jul. 2019].

[9] "Edge", Hypersecu.com, 2019. [Online]. Available: https://hypersecu.com/edge. [Accessed: 24- Jul- 2019].

[10] Coskun, V., Ok, K., & Ozdenizci, B. (2013). Professional NFC application development for android. John Wiley & Sons.