Multi-Factor Authentication for Smart-Lock Devices

Aditya Sirish Arunkumar Yelgundhalli
asy278

Wei Wang
ww1306

Xidan Xu
xx717

Shahran Islam
ssi256

Vineet Somanchi
vhs238

***Abstract*—Smart-Locks are designed to replace the traditional locks on your doors. They typically offer several smart features such as current status, a log of all entries and exits, and the ability to remotely provide access to third-parties. However, several commercial products have been found to be quite unsecure and are often vulnerable to relatively straightforward attacks. While these vulnerabilities must be patched and defended against separately, we propose a solution that implements a Multi-Factor Authentication that utilizes the Time-Based One-Time Password algorithm to authenticate every unlock attempt of the smart-lock.**

Keywords—multi-factor authentication, smart-lock, bluetooth, two-factor authentication, totp, time-based one-time password, one-time password

I. Introduction

Multi-Factor Authentication (MFA) is the concept of having multiple layers of authentication so that even if one layer is compromised, the other layers are able to fend off attackers[1]. The first layer is more straightforward, typically a username and password combination, something that an authorized user knows. The second layer requires authorized users to have some additional information that can confirm they are indeed who they are claiming to be. This information is typically linked to a specific session or instance of authorization, and it must compulsorily be obtained from a source that's not linked to the source of the first layer.

Multi-Factor Authentication is a useful option to check the authorization a person has to unlock a smart-lock. The most ubiquitous form of MFA today is a one-time password, where a random code is generated by the service and sent to the user via some previously established trusted means of contact. MFA, however, can also incorporate other authentication mechanisms, especially with biometric scanners becoming more prevalent.

Several third-party applications that offer multi-factor authentication for services have risen to prominence. Some of the bigger ones are Google Authenticator, LastPass Authenticator, Duo, and Authy. Typically, these applications implement a Time-Based One Time Password generation algorithm, where they continuously generate a one-time password based on an algorithm that they've previously agreed to with the service. Every time the user requires a one-time password, the app simply provides the latest one.

While several smart-lock manufacturers include multi-factor authentication in their locks, they typically incorporate it for users to login to the app via their credentials. There is no multi-factor authentication involved in the actual unlocking of the smart-lock.

II. Background

A single authentication technique is no longer secure enough for scenarios that must be kept secure, simply because of the number of moving parts involved in any moderately complex product. Any straightforward authentication can be easily bypassed with either social engineering or due to user error where they re-use passwords across multiple different websites and services – if any of them are attacked, the same credentials could then pass muster on your service.

Therefore, for any system that requires a high degree of security, we must implement multi-factor authentication. This ensures that if an attacker manages to get access to one authentication technique (maybe via leaked credentials from some other service), they're still locked out of the protected system that uses multiple sources of credentials or authentication keys.

Thus, it is vital that the multiple factors of authentication are truly from different sources. If they're from the same source, a successful attack on just that one source is sufficient to breach the system.

Multi-Factor Authentication in smart-locks must be implemented in a cost effective manner. That automatically limits options by taking away more complicated and expensive methods such as advanced biometric scanners. A more straightforward fingerprint scanner, which is cheaper but also less accurate, might also be susceptible to certain attacks[2]. Thus, it makes more sense to stick to a traditional implementation - the one-time password. To implement a one-time password-based system, we have to consider several factors:

A. Source of the one-time password

B. Transmission of the one-time password

C. Algorithm used

D. Redundancy and backup options

A. Source of the one-time password

Primarily, we must decide if the one time password is to originate from a centralized server or from the smart-lock controller itself. The centralized server may be operated by the

manufacturer of the lock or some other trusted middleman. However, this has certain implications on availability. What happens if the server is overloaded or unavailable for some reason? With increased risks of packet sniffing, packet forgery, and DNS spoofing, centralized server usage poses immediate threats to all users, whether it be leaving people stranded outside their houses or comprising access to their locks[3]. If the company operating the server goes out of business, every single smart-lock device that requires a one-time password to be generated by the server is essentially rendered a brick.

On the other hand, making every smart-lock controller capable of generating and dispatching codes on demand can be computationally expensive. What's required is an algorithm that is capable of quickly generating and dispatching the one-time password using the limited hardware capabilities of the smart-lock while balancing the cost of transmission and the vulnerabilities associated with it.

B. Transmission of the one-time password

The various options for securely transmitting the one time password are:

i. Short Message Service
ii. Phone Call
iii. Email
iv. Trusted Devices

i. Short Message Service

While SMS was widely used to deliver one time passwords in the early days of two-factor authentication, it has since dropped in popularity simply because it's not a very secure mode of communication. There are numerous examples of potential threats. SS7 probes can be implemented to capture data for legal intercepts especially on untrusted networks like the internet. Additionally, easy kits online enable users to tamper and intercept outbound data on any individual's phone. Finally, SMS data resides at SMS gateways and staging posts putting users at threat of a man-in-the-middle attack where large amounts of bulk data can be compromised[4]. Overall, SMS has proven to be unsecure.

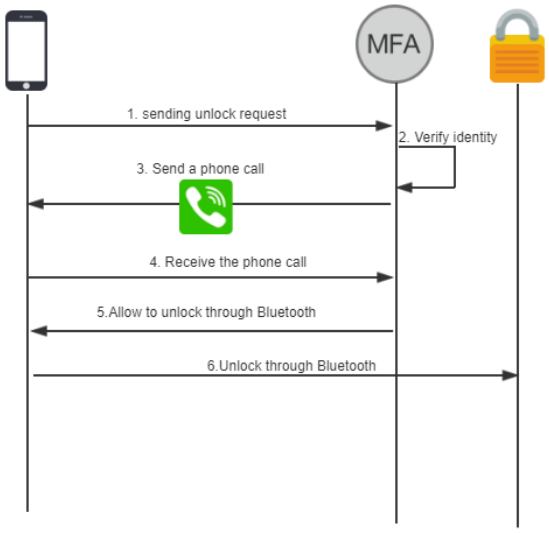

Fig. 1. Multi-Factor Authentication using a phone call to a trusted number

ii. Phone Call

A phone call can be used to reach an authorized number. A one-time password can be verbally conveyed to the user or the simple act of receiving the phone call and selecting an option can be confirmation that an authorized person is attempting to unlock the smart-lock. In some cases, as seen in Fig. 1, the act of responding to a phone call is considered sufficient authorization. However, phone calls are typically susceptible to the same vulnerabilities that SMS is.

iii. Email

Email is used especially by certain financial institutions to send codes, especially for situations where a customer is abroad and SMS is not a viable option. But email may also be a poor solution from a security point of view as we're reliant on the email provider to maintain security. It's also possible, and in many cases probable, that the user's password isn't secure either.

iii. Trusted Devices

Push notifications can be sent to trusted devices as seen in Fig. 2. These devices must be set up before hand – each lock must have a limited number of devices that can receive these notifications for confirmation from the user.

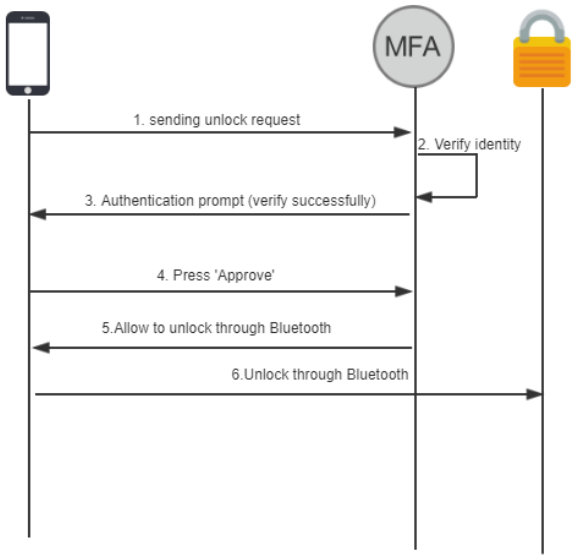

Fig. 2. Multi-Factor Authentication using a push notification

However, while a lost trusted device can still potentially leave an attacker locked out (assuming the phone is reasonably well protected), it can also leave an authorized user stranded. So the next step is to develop a solution where there is a failsafe backup for authorized users to unlock the smart-lock. For any

solution with a one-time password, we must include a way to enter the code into the lock. The lock must therefore include a keypad in some form.

We must also separate the receiving app on the device from the smart-lock's main app. If the main user credentials for the smart-lock app are compromised, the attacker must not be able to receive the one time password. So, either the smart-lock manufacturer must include a separate app to generate one time passwords, or the user must have the option of using any third-party app capable of scanning the algorithm from the lock and generating one-time passwords. Several such apps already exist as detailed above.

C. *Algorithm used*

There are multiple options available. We must broadly decide between the following two options:

i. Time Synchronized one-time passwords

ii. Mathematical / Hash one-time passwords

i. Time Synchronized one-time passwords

Algorithms such as Time-Based One-Time Password (TOTP)[5] use accurate clocks to generate a one-time password based on the current time. They are valid for a short duration of time, usually about 30 seconds. The clocks on the unlocking device must be closely synchronized with the smart-lock itself, so that the code can be matched. Since a one-time password is always available in this situation, these locks aren't dependent on the transmission. The user always has access to a valid one time password on a trusted device. Using the TOTP algorithm also removes the requirement of transmission of the one-time password to the trusted device, as generation happens on both ends independently, as seen in Fig. 3.

ii. Mathematical / Hash one-time passwords

These algorithms[6] rely on previous OTPs to generate the next OTP. They usually involve one-way hash functions, typically cryptographic hash functions where calculating the inverse is computationally. Some of the passwords generated are valid until they're actually used. This could be a weakness, however, and they should have limited validity. They are usually generated and dispatched on demand[7] - when a user is actually attempting to unlock the smart-lock.

D. *Redundancy and backup options*

Many problems can arise by relying on the correct transmission of the one-time password. Most transmission options can be hijacked, allowing attackers to gain access to the one-time password. Alternatively, transmission failures can cause availability issues, where an authorized user is left stranded and unable to unlock the device simply because they are unable to receive the code.

If we're transmitting the code to the users, we have to consider situations where users are not connected to the Internet, or when the centralized server goes down. The manufacturer's server is usually a prime target and the manufacturer may also

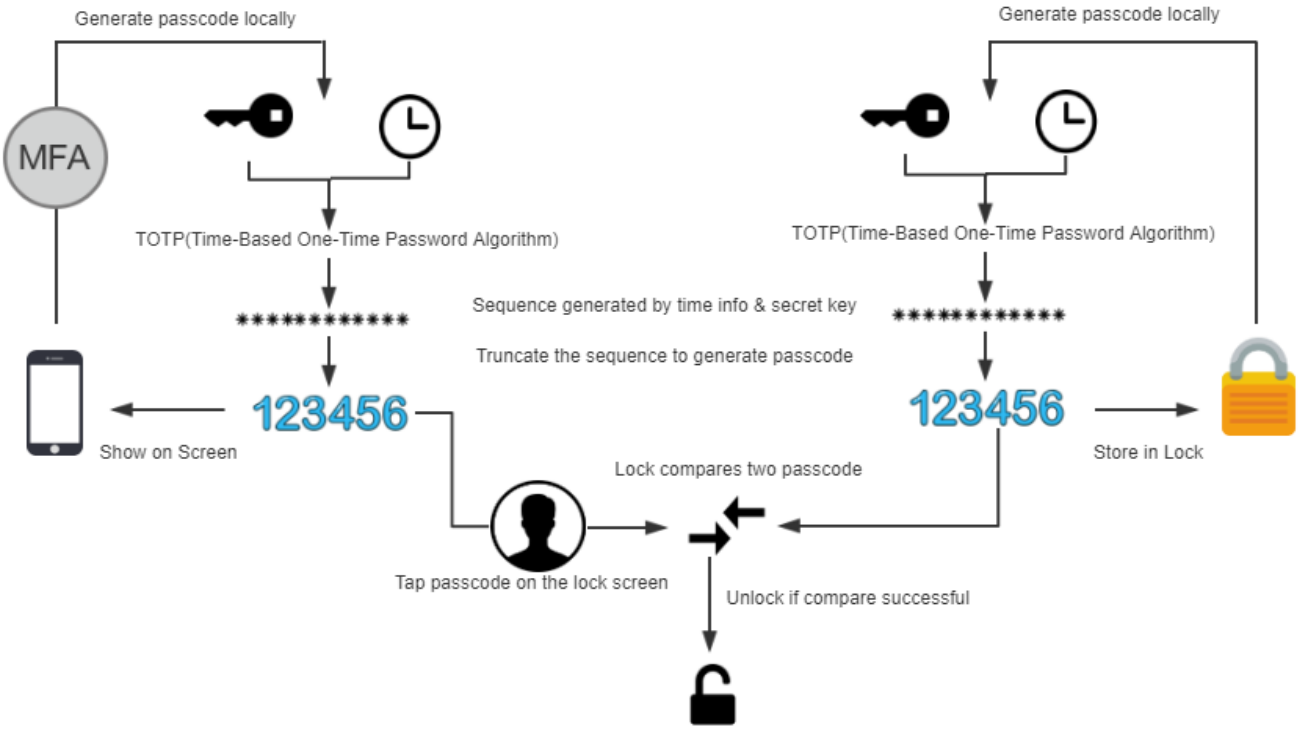

Fig. 3. Using the TOTP algorithm to generate the one-time password independently at the lock and on the user's device

have lesser resources. Thus, it may be better to outsource this by using services offered by the likes of Microsoft, Google. and Amazon.

There must be multiple options for transmitting the code to the user that offer varying degrees of security, but still ensure that an authorized user is not incorrectly considered an attacker.

To summarize, any usable solution must:

- Provide appropriate security via a reasonably random one-time password of some minimum length
- Use a way of generating and transmitting (if applicable) codes that can't be hijacked
- Have fallback options for situations without network connectivity or in case of power failures

III. Threat Model

We previously discussed possible attacks for a smart-lock that doesn't necessarily include multi-factor authentication at the unlocking event itself. The attack tree for such a system is given in Fig. 4.

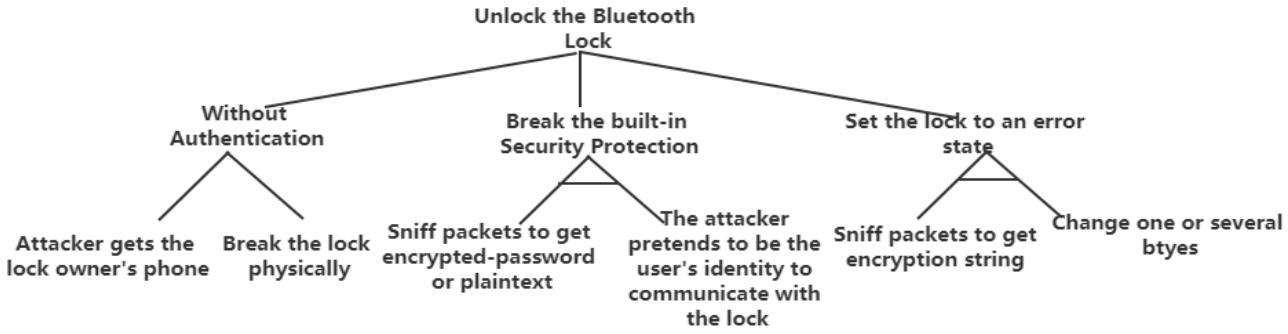

Fig. 4. Attack trees for smart-lock systems without multi-factor authentication

IV. Design and Implementation

Now that we have some understanding about what a good solution incorporates, we can discuss our primary solution. We propose that smart-locks use Time-based One-Time Password algorithm to generate codes based on the current time and some shared secret. There are several standalone applications that support generating one-time passwords by scanning a code for the specifics of the algorithm. The scanning process must also include identification details for the smartphone or app instance, to allow for later revoking – this identification also includes a secret known only to the lock and the device, and this secret is used to actually generate the one-time password each time. Revoking would lead to the smart-lock not generating a one-time password that matches the revoked device or app instance.

At the first setup of the smart-lock, one smartphone is setup as an authorized device. All future new devices must require authentication from an existing authorized device in the form of a valid one-time password.

It should also be possible for an authorized user to add a new remote device or user as long as they also have physical access to the smart-lock controller in the unlocked state. This will allow for users to give temporary access to guests or other short-term visitors.

For the unlock process itself, the TOTP compatible app on an authorized device generates a code using the algorithm. If this code matches the one generated by the controller, the user is granted access.

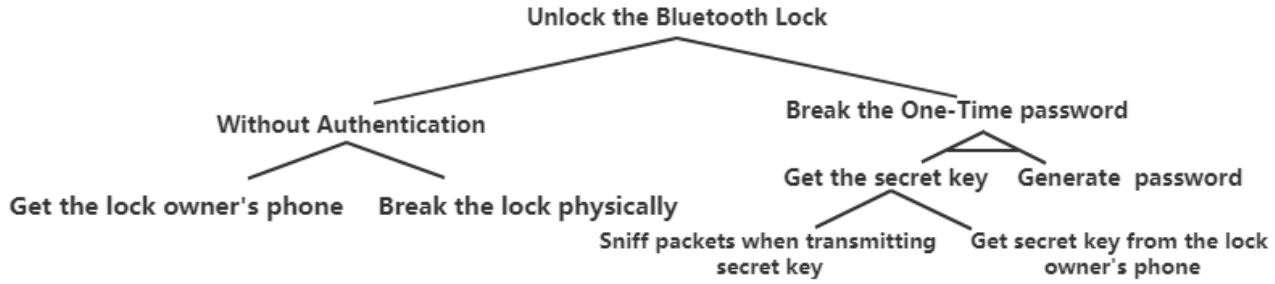

Fig. 5. Attack trees for smart-lock systems with multi-factor authentication

A vast number of threats can technically continue to exist but still not work successfully due to the introduction of multi-factor authentication. Fig. 5 depicts an attack tree for a system with multi-factor authentication for the unlocking event. Clearly, the breadth of the attacks is far more limited. A major liability is the possible transmission of a one-time password – however, any system that bypasses this is automatically secure on that front.

There can, however, be situations where the users are unable to authorize because of something as simple as a dead battery all the way up to theft or loss of the device. It's vital for the system to provide several single-use codes that can be used to authenticate in place of the one-time password. It's up to the user to protect these codes – typically, it's sufficient to memorize two or three of these codes or store them in a safe location outside the domain protected by the smart-lock.

As the algorithm is dependent on the current time, it's vital to regularly synchronize the time of the smart-lock and the smartphone used. It's also important to allow for user delays in entering the one-time password – for example, they may accidentally enter a password just as it becomes invalid. The TOTP algorithm provides for such scenarios by accepting older passwords for a time frame just beyond the validity of each password.

V. Evaluation

Smart-locks are unfortunately not as secure as they should be considering what they're actually protecting. As they're dependent on the merging of various different technologies, vulnerabilities can be present at each of those, sometimes beyond the control of the manufacturer. By implementing Multi-Factor Authentication, the smart-lock manufacturer is able to introduce a layer of security that is completely under their control. As long as manufacturer implement the algorithm in a secure manner and with a sufficient degree of randomness, the solution should help mask any other vulnerabilities beyond their control.

It's important to emphasize that the proposed solution protects the smart-lock itself using multi-factor authentication. Several existing smart-lock implementations incorporate multi-factor authentication but only in protecting the smart-lock account – the user can't login to the app of the smart-lock without using multi-factor authentication. This should still be present too, but its implementation will obviously differ from the one detailed here.

A. A typical unlocking scenario

In a typical scenario, users are required to use their smartphone to indicate they wish to unlock the smart-lock. This can be streamlined to allow the smart-lock to automatically detect the presence of an authorized device. The smart-lock then prompts them for their unique one-time password, which they then enter using a keypad at the door. Several optimizations can be introduced on the mobile device to streamline the process of generating the code and making it available to the user. They are, however, operating system specific.

Alternatively, the one-time password must be entered into the smart-lock's main application. Since the process of logging into the app doesn't automatically give users access to the one-time passwords, this technique is still secure. This removes the necessity of a keypad at the lock.

B. Allowing access to a third party

An authorized user can give remote access to third-parties, perhaps with an expiry or other rules of access such as a specific time interval of entry. The TOTP algorithm accepts an additional secret which is unique to that instance – this secret and the time stamp are used to generate one-time passwords via a cryptographic algorithm. This secret must be shared with the user's TOTP compatible app. Typically, this can be scanned directly using a QR code, but it can also be sent remotely. However, the generation of this new instance requires the lock to be in an unlocked state, to ensure only an authorized user can add the third party.

The lock generates a one-time password for each of the instances, and thus, revocation of one of these instances is as simple as not generating the corresponding password at the lock.

C. Weaknesses

As with most solutions, our proposed solution relies on the degree of care taken by the user. Backup single-use one-time passwords must be stored securely – failure to do so could allow an attacker to bypass the multi-factor authentication. Ideally, they must be stored in a place accessible only to authorized users and protected by some other means of security.

If a user or home is specifically targeted, an attacker can gain access to a trusted device which allows them to impersonate an authorized user. Just like the traditional lock which requires users to protect their keys, it is the users' responsibility to protect their smartphones.

VI. Related Solutions

A. On-demand one-time password generation and dispatch

In this technique, the one-time password is generated on-demand. This one-time password is then securely transmitted to a trusted device. Users then utilize this code to unlock the smart-lock.

The primary issue with this technique is where the code is generated. It's unwise to rely on external servers of say, the manufacturer, as any attack on this single server can affect all users of the lock. On the other hand, if we move code generation to the lock controller itself (as seen in the proposed solution), we must then allow for secure transmission of this code. This technique also has greater probability of failure as both the smartphone and the controller must always have a network connection. In the proposed solution, a network connection is absolutely necessary only for resynchronization of the clocks on both devices.

B. Biometric alternatives to one-time passwords

The solution could also instead use a biometric alternative[8] to the one-time password. Authorized users must setup their fingerprints or any other predetermined biometric information and use this for secondary authentication. However, several smart features of the lock may be lost, such as giving access to a third party even when they're not physically present at the location of the lock.

Biometric scanners tend to also be more cost prohibitive, especially those with a high degree of accuracy. On the other hand, fingerprint scanners that scan more cursorily can also be vulnerable to attacks similar to DeepMasterPrints[2].

VII. Conclusion

Certain trade-offs have been made in the interest of security – authorized users cannot authorize anyone else unless they have physical access to the smart-lock in its unlocked state. On the other hand, the proposed solution offers fallback options to adequately handle situations where users are separated from their authorized devices or are just not in a position to use them. The solution also allows for sharing access with some third party – a major positive factor of a smart-lock – in a secure manner, and also for later revocation of access. We can therefore conclude that implementing secure multi-factor authentication for the actual event of unlocking a smart-lock can be a major strength for security.

References

[1] Jim Reno. “Multifactor Authentication: Its Time Has Come.” Technology Innovation Management Review, Iss August 2013: Cybersecurity, Pp 51-58 (2013), no. August 2013: Cybersecurity, 2013, p. 51.

[2] Philip Bontrager, Aditi Roy, Julian Togelius, Nasir Memon, Arun Ross. “DeepMasterPrints: Generating MasterPrints for Dictionary Attacks via Latent Variable Evolution”. arxiv 1705.07386.

[3] Chien, E. Centralized Server Threats. Stanford University, 2 Feb. 2016.

[4] Scotland, N. (2016). Information Risk Assessment: Use of Short Messaging Service (SMS) in NHSScotland[Scholarly project]. Retrieved December 5, 2018.

[5] D. M’Raihi, S. Machani, M. Pei, J. Rydell. (May 2011). "TOTP: Time-Based One-Time Password Algorithm". IETF. RFC 6238.

[6] M. H. Eldefrawy, K. Alghathbar and M. K. Khan, "OTP-Based Two-Factor Authentication Using Mobile Phones," *2011 Eighth International Conference on Information Technology: New Generations*, Las Vegas, NV, 2011, pp. 327-331.

[7] D. M’Raihi, J. Rydell, S. Bajaj, S. Machani, D. Naccache. (May 2011). "OCRA: OATH Challenge-Response Algorithm". IETF. RFC 6287.

[8] Scott Mahnken, Today's authentication options: the need for adaptive multifactor authentication, Biometric Technology Today, Volume 2014, Issue 7, 2014, Pages 8-10, ISSN 0969-4765, https://doi.org/10.1016/S0969-4765(14)70126-2.