STRONG AUTHENTICATION BASED ON MOBILE APPLICATION

Helsingin yliopisto
Faculty of Science
Department of Computer Science

Master's thesis
Spring 2020
Harri Salminen
Supervisor: Valtteri Niemi

HELSINGIN YLIOPISTO

Tiedekunta - Fakultet - Faculty	Osasto - Avdelning - Department
Faculty of Science	The Department of Computer Science

Tekijä - Författare - Author
Harri Salminen

Työn nimi - Arbetets titel

Title
STRONG AUTHENTICATION BASED ON MOBILE APPLICATION

Oppiaine - Läroämne – Subject
Computer Science

Työn laji/ Ohjaaja - Arbetets art/Handledare - Level/Instructor	Aika - Datum - Month and year	Sivumäärä - Sidoantal - Number of pages
Master's thesis / Valtteri Niemi	June, 2020	58

Tiivistelmä - Referat - Abstract

The user authentication in online services has evolved over time from the old username and password-based approaches to current strong authentication methodologies. Especially, the smartphone app has become one of the most important forms to perform the authentication. This thesis describes various authentication methods used previously and discusses about possible factors that generated the demand for the current strong authentication approach.

We present the concepts and architectures of mobile application based authentication systems. Furthermore, we take closer look into the security of the mobile application based authentication approach. Mobile apps have various attack vectors that need to be taken under consideration when designing an authentication system. Fortunately, various generic software protection mechanisms have been developed during the last decades. We discuss how these mechanisms can be utilized in mobile app environment and in the authentication context.

The main idea of this thesis is to gather relevant information about the authentication history and to be able to build a view of strong authentication evolution. This history and the aspects of the evolution are used to state hypothesis about the future research and development. We predict that the authentication systems in the future may be based on a holistic view of the behavioral patterns and physical properties of the user. Machine learning may be used in the future to implement an autonomous authentication concept that enables users to be authenticated with minimal physical or cognitive effort.

Avainsanat – Nyckelord

Keywords
Authentication, Mobile computing, Software and application security

Säilytyspaikka - Förvaringsställe - Where deposited

Muita tietoja - Övriga uppgifter - Additional information

1 Introduction

Today millions of customers of online services are using smartphones. It is very common that the primary accessing channel for the service is either purpose built mobile application, or a web browser interface optimized for the mobile phone use. Also, the online services commonly use the smartphone of the end-user as an asset that identifies the individual user. This thesis studies the factors that helped the smartphone to become such an asset. The thesis further studies where the technical development is heading next and presents important topics that may affect the future.

1.1 The past

During many decades, the most used access control mechanism on digital services has been username and password. The users are identified (authenticated) by the username and password pair that they present during entering the service. This has been relatively simple and low-cost solution for the designers and the owners of the services.

On the other hand, the passwords caused many concerns for the users and the service providers. Passwords may be forgotten or stolen. They are also sometimes easily guessable. Often users re-use passwords between different online services and so data breach on one service usually generates a real threat to other services. Despite these problems, more secure and more advanced user authentication mechanisms (which existed) were not widely taken into use outside military, banking or corporate domain until the last decade.

As mentioned, more secure authentication mechanisms have existed many decades. For example, in the corporate use the end-users have been given small electronic devices, so called hardware tokens, that they could use when accessing the critical systems. At banking sector, some institutions have traditionally used transaction authentication numbers in the form of a list of number sequences printed on card board. These numbers are used by customers during entering the bank online service.

It is an interesting question why the more secure authentication mechanisms have not achieved more popularity compared to the password. Lot of research has been done about the usability of the authentication methods. Generally speaking, it has been discovered that the secure authentication methods might have been too expensive for consumers, clumsy to use, or not intuitive.

1.2 The present

The total count of mobile subscribers worldwide has been estimated to be around 5 billion [1]. Over half of them have smartphones [2], so majority of world’s

population are potential users for online services that use the smartphone as the access channel. Many services use the smartphone as an identification asset for these users. However, lack of generic and widely adopted standards has slowed down the evolution of the smartphone based user authentication.

Security of the mobile apps and the mobile platforms has been a concern [3]. There are many attack scenarios threating the authentication of the users in smartphone. The user authentication performed by a smartphone application is usually based on secret keys that are stored inside the persistent memory of the mobile device. These secret keys can be stolen by malicious software under certain circumstances. Also, the data that the mobile applications transfer over the network may be monitored and altered by unauthorized parties unless there are mechanisms preventing it.

Despite the security related challenges of the mobile apps, they have gained lot of popularity. Today, all financial mobile apps in Europe are required to fulfill regulatory specifications that tie the user identity to the smartphone [4]. Also, there exist some generic mobile apps (like national electronic id app) that can be used like an electronic passport.

1.3 The future

One future scenario about the digital services is that they could be supplied to the consumers autonomously (but of course by the consent of the user). Advanced development of the computer vision may help to identify the persons “on-the-fly” without any explicit maneuvers done by the individuals. The rich data that can be collected continuously from the users by various sensors of the mobile devices are input source for machine learning algorithms. These algorithms can support user authentication and various other useful purposes.

2 Basic concepts and definitions

Many of the concepts and terms are not generally expected to be known by the reader of this thesis. In particular, the chapters that contain elements of the cryptography may be difficult without background knowledge. The following chapters try to list and clarify the most important concepts and terms that are discussed in deeper level later.

2.1 Authentication

In computing, the term *authentication* is used for a process of verifying the identity of the user (person). Also, the authentication can mean verifying the identity of a device. Furthermore, the use cases of *transaction verification* have similarities to the verifying the identity of the user. In transaction verification, the user is requested to assert if the presented details about the transaction are correct and if the user accepts them (for example payment in a web shop).

2.2 Single sign-on

In the single sign-on (SSO) model [5] [6] [7] the end-users are using a single digital identity when they are logging into many different software systems. In some early implementations of the single sign-on the users had one common username/password pair, that they could use when they logged into the related systems. Single sign-on has traditionally been popular in corporate use.

2.3 Federated login

The federated login (FID) is related to the single sign-on [8]. In FID, the end-user may also login to various software systems using single digital identity. Federated login means that the software system assigns the task of authentication to an external authentication system. In the federated login concept, the assignor software system is called typically as *relying party* and the assignee, the external authentication system, as *identity provider*. The assignor software system can also be called as *service provider* in a certain context where the trust dependency between the parties is not specially under interest.

2.4 Mobile app authenticator

In this thesis, the term *mobile app authenticator* covers all mobile applications for smartphones that execute strong authentication during their usage. Mobile app authenticator may be purpose built for authentication use only. Also, the generic

mobile apps, whose main purpose is other than just authentication (for example financial management apps) are classified as mobile app authenticators in this thesis when they include a strong authentication component. Mobile applications that do not use strong authentication are not covered here.

2.5 Strong authentication

Today the term strong authentication [9] can be seen as variation of the term multi-factor authentication. The challenges of the terminology are explained later in chapter 3.1. The logical basis of the strong authentication is to have at least two of the three authentication elements in the authentication process of the end-user. These elements are *knowledge*, *possession* and *inherence*.

Knowledge can be for example a password that only the user knows. It is important regarding this factor that the information is not shared with anyone else than the user or stored anywhere in an unprotected format.

The *possession* element is typically met when the user has a dedicated device (for example, mobile phone) which is used to perform the authentication. It is questionable if shared devices can fulfill the requirements of possession.

Inherence refers to use-cases where individual properties of a human are used to prove the identity. These individual properties may be, for example, fingerprint or even keyboard typing patterns.

2.6 Personal identification number

The personal identification number (PIN) [10] has been common knowledge based authentication method for decades. PIN is a sequence of numbers that user has selected to identify himself/herself. Randomness of the PIN is crucial for the security of the authentication scheme.

PIN has been used, for example, with automated teller machines of banks, accessing Wi-Fi networks and answering machines supplied by teleoperators.

2.7 Blockchain

Blockchain [11] [12] is based on a linked list of records (blocks). Each block in the chain is tied cryptographically to the previous block. This creates a recursive cryptographic proof of the data validity in the chain. No block can be altered without breaking the cryptographic proof of the chain.

2.8 White box cryptography

Normally the cryptographic keys used by computer programs are observable by such foreign parties who can access the program runtime, with the help of specific tools. This possibility reduces the security of the programs and make them vulnerable to attacks.

White box cryptography is a methodology that reduces the observability down to a point where the access to keys may require so much effort that it exceeds the possible gain [13] To achieve this low level of observability the keys and the used cryptographic algorithms are altered in multiple ways. These alterations are explained later in this thesis.

2.9 Public key cryptography

Public key cryptography [14] is cryptographic concept that is based on a mathematically tied key pair. One of the keys in the pair is public and can be shared freely. Another key (private key) is confidential and should always be kept secret.

Public key cryptography offers two main functions: authentication and encrypting. For example, if sender of a message wants to ensure the confidentiality of the message content, he/she can encrypt the message payload with the public key of the receiver. Only the holder (owner) of the private key can successfully decrypt the message and so only the receiver is able to access the payload.

Authentication in the public key cryptography is based on digital signatures. The digital signature is generated by a signature algorithm that uses the private key and piece of data as input. Anyone with access to the public key can verify that the signature was created using the private key. The verification is performed by signature validation algorithm.

2.10 Public key infrastructure

Public key cryptography, which was briefly explained in the previous chapter, can be applied to real-life use-cases several ways. One common way is to use public key infrastructure (PKI) [15].

PKI consists typically roles, policies, technical standards and a central certificate authority (CA) to manage the digital certificates. The digital certificates are issued to the entities owning private keys. Typically, the CA signs the certificate that contains an identity, a corresponding public key and some other information such as the expiration time of the certificate.

2.11 Identity and access management

As the user authentication and user role management have been very important features in the enterprise software systems from the very beginning, a specific industry has risen around the problem area.

Identity and access management (IAM) is the industry which supplies software, standards and solutions around the authentication, access management and user role management. The global IAM business market size was estimated to be 9,53 billion USD in year 2018 [16].

3 Statement of the problem

We have a premise that the mobile app authenticators became one of the most successful strong authentication solution during the last decade. The premise is based on various observations. For example, there has appeared number of new companies on the current identity and access management (IAM) markets supplying mobile application as their main offering. Additionally, the surrounding legal frameworks in IAM and payments area (in Europe specially) has been set in a way that the mobile authentication use-cases have been clearly identified and supported.

Generally speaking, high proportion of companies are doing business successfully with a strategy that the mobile applications are their main user interaction channel. If they need payments and they do business in Europe, they likely must use strong authentication by the regulation.

In this thesis, we try to find out the factors that helped the mobile app authenticators to gain popularity. We try to gather information about how the mobile app authenticators are typically constructed and how they work. With this information, we can reflect the current situation to the future and try to state potential scenarios about the development. We set the question: what could be the next generation of strong authentication solutions and how they are built?

Also, we explore the potential threats against mobile app authenticators. These threats may be factors that affect to the popularity of the current strong authentication solutions. They may assist the rise of the next generation solutions that perhaps address the challenges seen with the current ones.

3.1 Challenges and limitations

One major challenge is the terminology of the strong authentication. There are multiple overlapping terms used for “strong authentication”. Some information sources and publications handle this through a term *two-factor authentication* (2FA). Two-factor authentication means an authentication process that has two phases. At the first phase, the user typically uses the username/password pair. The second phase is often performed by using additional device like hardware token or mobile phone.

After common stabilized usage of the term 2FA, appeared the second European payment services directive (PSD2) and the regulatory technical standard (RTS) about strong customer authentication and secure communication [4]. These were published by the European Union and the European Banking Authority (EBA).

PSD2 and the related RTS defined the term *Strong Customer Authentication* (SCA). SCA, by the definition in the regulation, requires that the system identifies the person with at least two independent factors. On the other hand, a term multi-factor authentication is sometimes used.

Yet another confusing term exist in some commercial surveys regarding usability of the authentication. In some publications, the term *push* may cover there all the mobile authentication apps that utilize push notifications to notify end-user. However, the push notification is not a mandatory element for a mobile app based authentication scheme. So, the classification of the mobile app authenticators around the word *push* is questionable.

4 Common authentication solutions

Various weak and strong authentication (or 2-factor authentication) methods have been available for commercial use for decades. As stated before, the usage of the strong authentication methods has been common first in other than consumer targeted services: military, corporate, government services etc. The adoption of the strong authentication has later been spread to the services for consumers due the legislation and the rising common knowledge about the security risks, for example, in banking.

Strong authentication can be built on multiple authentication methods that have varying security levels. We also cover some old weaker legacy authentication methods here because they can (depending on the requirements) be used as individual authentication factors forming together the strong authentication.

4.1 Hardware tokens

Hardware tokens are peripheral devices that are used to allow access to restricted resources. They can be connected or disconnected.

Typically, the tokens are used in combination with a computer to perform authentication to a restricted resource like bank web site or corporate system. *Disconnected token* does not have a physical or logical connection to the computer. Hardware code generator is common example of this type of a token.

Connected tokens do interact with the computer during the authentication event. They may be connected to the computer through physical communication interface like USB-port or through wireless connection (Bluetooth for example).

The WebAuthn [17] and previous U2F/UAF standards of the FIDO consortium have generated a common authentication framework for using hardware tokens for authentication in mobile and browser environments. This has helped connected tokens to gain popularity over the traditional non-connected tokens.

4.1.1 Tokens with input mechanism

It is possible that hardware has an input mechanism for a PIN (personal identification number) or biometric input mechanism like fingerprint scanner [18]. This is remarkable enhancement if we think the strong authentication factors and the strength of the authentication performed with the token.

Hardware token that has no input mechanism supplies only the possession factor of the strong authentication. The user can prove during the authentication process that he/she has access to a specific pre-registered hardware token. But it is not proved that the user is the person who the token is registered to.

Token with PIN input capability allows the access to the authentication functionalities only for a person who knows the PIN. Of course, the PIN can theoretically be guessed by very small likelihood or user can share it to another user, (which typically violates the terms of the service). However, the PIN brings the additional knowledge factor to the authentication. So, the authentication event performed with PIN equipped hardware token supplies two factors of the strong authentication.

Hardware token having a biometric input mechanism has the inherence factor of the strong authentication in addition to the possession. Some practical examples of this type of hardware token are Feitian BioPass-series that have fingerprint reader [19] and Yubikey Bio tokens by Yubico [20]. During the initialization of this type of token the fingerprint of the user is scanned and stored securely into the token. After this, the authentication functionalities of the token cannot be accessed without presenting the registered fingerprint.

4.1.2 Public key cryptography with hardware tokens

Public key cryptography (PKC) is common useful technology with hardware tokens. The public key concept of the PKC can be treated as the identity of the end-user. And the cryptographic signatures calculated on the private key can be

seen as an analogy to the consent of the user or a permission granted by the person.

However, the authentication using hardware token and public key cryptography requires typically multi-directional communication between the parties involved into the process. The authentication system must have communication channels and protocols supporting this.

The communication requirements of PKC with hardware tokens

We can set following questions, when examining the communication requirements of an authentication with PKC and hardware token (specially in an online service context).

The authentication using PKC is based on digital signatures. The capability to build digital signatures requires access to the private key that is protected by the PIN. *How the end-user is able to input the PIN code to the token*?

The digital signing algorithms require a source message as one input parameter. *How to transmit the source message for the digital signing process inside the token*?

It is possibly required, that the end-user must see the authentication request details during the authentication [4]. *How these details are shown to the end-user*?

The digital signing algorithms produce signatures which must be verified by the party which controls the access to the protected resource. In the context of the online services this means that the backend server or separate authentication server of the service must verify the signature. *How the signature is transferred to the verification process*?

Standardized communication channels and protocols

Standardized communication channels and protocols bring answers to the questions and challenges mentioned in the previous chapter. Standard API and interconnection protocols generate major benefits for the application developers specially if the system is designed to allow usage of tokens from many vendors.

PKCS#11 [21] is one notable standard for hardware tokens. It defines the standard programming API (application programming interface) for the vendor specific software libraries. The authentication system components running in a client computer (practically a PC) can connect to the hardware token and utilize the PKC services through this API.

The following is an example of an authentication sequence through the PKCS#11 API from the point of view of an application developer.

1. polling the presence of the physical token
2. generating a session to the token (PIN is required as input to open the session and access to the memory)
3. enumerating the key identifiers and certificates from the token in order to check if the token is initialized into the actual use
4. initialization of a signing event (setting the signed content into the shared memory and selecting the signature scheme)
5. committing the signing event
6. finishing the signing event
7. closing the session to the token

Hardware tokens in the web context

One example of the involved parties and required messaging of an authentication using PKC and hardware tokens in web context is presented in Figure 1 and Figure 2. However, it is notable to mention as a disclaimer for these figures that the web browser manufacturers have set major limitations during the last decade regarding the connectivity possibilities to native libraries.

Formerly the browsers supported widely a common API named NPAPI, that defined various programming interfaces for accessing native services outside the browser sandbox [22]. This enabled also accessing the functionalities of the connected hardware tokens. However, the NPAPI support has been deprecated from majority of the browsers [23] [24] [25]. It is possible that the NPAPI brought too much security risks compared to the benefits and so the browser manufacturers considered it better to discontinue the support.

On the other hand, during the fadeout of the NPAPI support there has risen a new series of standards based on the same problem area. FIDO consortium has published UAF, U2F, WebAuthn and CTAP standards to support strong authentication in web and mobile environments [17]. FIDO has addressed the challenge of the communication between the browser environment and the authenticator by defining set of APIs and communication protocols. In order to succeed, the FIDO standards need support from the all major browser manufacturers (Google, Microsoft, Apple, Mozilla). The required adoption seems not being achieved yet as the support from Apple is missing [26].

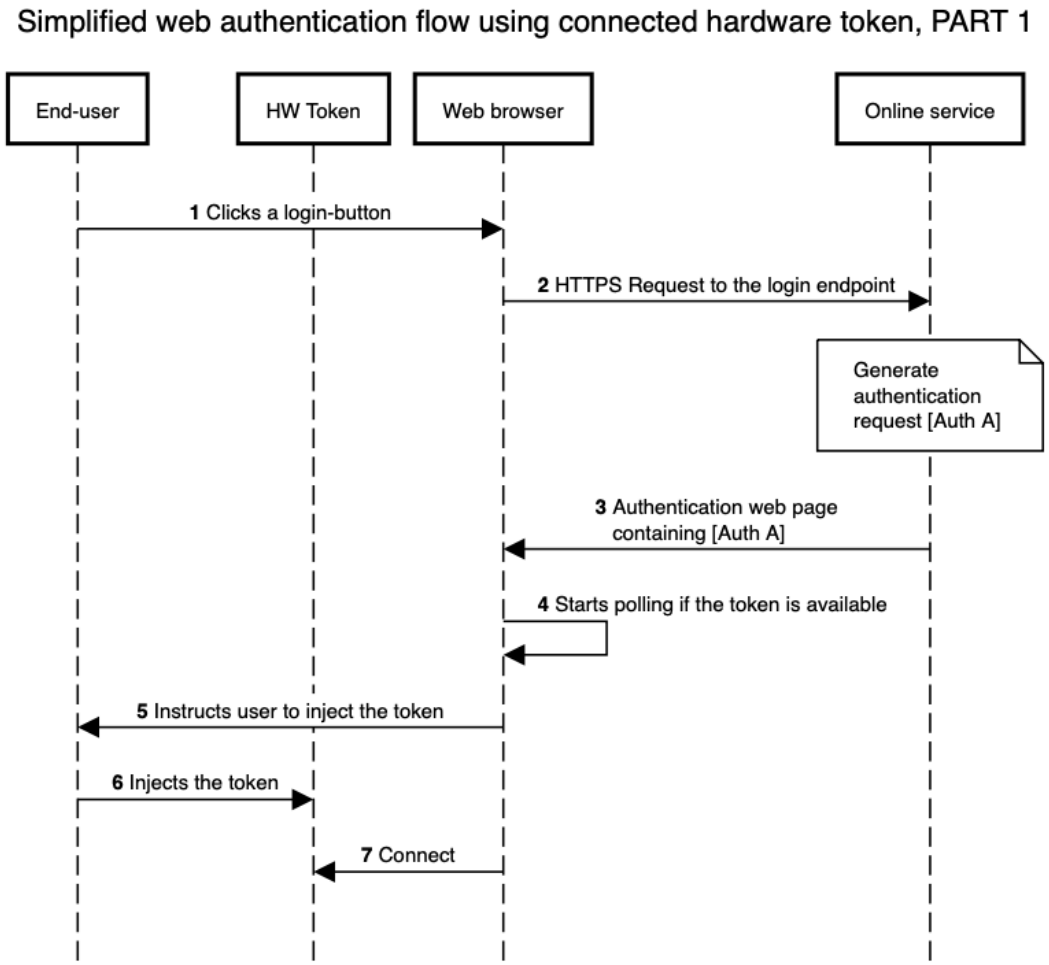


Figure 1 Simplified authentication flow using connected hardware token in a web environment. PART 1

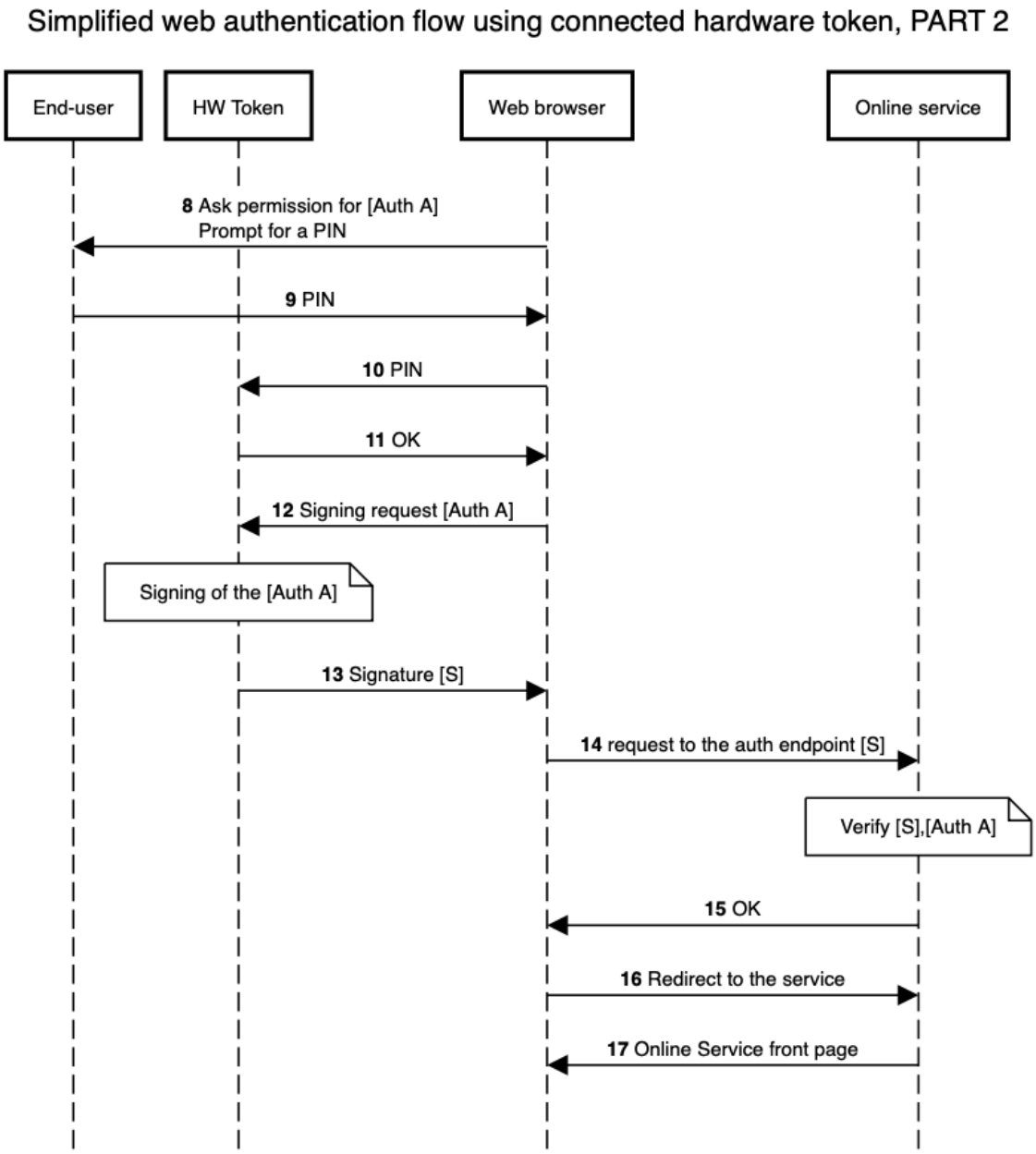


Figure 2 Simplified authentication flow using connected hardware token in a web environment. PART 2

4.2 Hardware code generators

Hardware code generator is typically plastic, battery powered digital device with LCD display for showing code that changes based on time [27]. The code is generated by a purpose-built digital circuit board that has an internal clock, persistent memory for a seed value and processing unit for the code generation algorithm. The *seed* is a value that is essential for the randomness of the code generation algorithm (and so essential for the security of the device). Typically, the seed is initialized into the token during the manufacturing process.

One popular standard for the timely basis changing codes is TOTP (Time-based One-Time Password algorithm). TOTP is defined by the RFC 6238 memo of the IETF (Internet Engineering Task Force) [28]. TOTP code generation algorithm is based on common shared secret between the server and client devices and the shared opinion about global time. The “time” means a certain physical time window, usually 30 seconds when the code stays the same.

Usually there may be some deviations in the time state between client and server. This is more likely to happen with offline devices that rely only on the internal clock of the circuit board and cannot reach any reference time from the network. These time deviations caused by clock drift may generate problems when the server checks the code calculated by the client. Common solution to compensate these deviations is that the server generates reference codes from the nearest time windows and compares the code also against them.

Hardware code generators can be classified to belong to the category of hardware tokens that is a broader class of various authentication devices. One of the most common hardware code generator in the commercial use has been RSA SecurId [29] that has been popular, for example, in corporate use.

4.3 Smart cards

Someone may set a question if a smart card is a hardware token or vice versa. The question relates to the fact that the cryptographic hardware modules are manufactured in various form factors. There are, for example, hardware security modules (HSM), that typically come as standalone box-shaped devices to be installed into a server rack. Then there are hardware tokens that are hand held gadget-like items. And after all there are the smart cards that have cryptographic capabilities as well.

We take the freedom to describe smart cards separately from the hardware tokens here. The typical form factor of the common smart card like SIM card of a mobile phone or EMV credit card with embedded chip is different than, for example, typical security token with a USB connector.

Smart cards are used widely in the payment industry as specific purpose-built solution for accepting payment transactions (EMV cards). Also, in the mobile industry the SIM card is the backbone of the identity of the mobile subscriber and the core services. SIM card can even support the subscriber identity services to custom applications through the SIM Toolkit functions that are standardized by European Telecommunications Standards Institute (ETSI) [30].

SIM Toolkit services is good basis for authentication solutions in mobile phone environments. The SIM card is designed to function as PKI backed cryptographic root of trust that is the hardest part to implement for an authentication system as we see later in this study.

Despite the excellent cryptographic foundation and enrollment of the root of trust the SIM Toolkit solutions have not turned out as widely adopted commercial success. One possible reason for this is the fact that standard SIM Toolkit functions have had limitations regarding the graphic user experience (in comparison to the current user experience standards in the smartphones). Also, the SIM Toolkit functionalities are deployed into the SIM by the mobile operator, so the commercial solutions require co-operation with the mobile operator that can be bureaucratic and slow. This might have reduced the popularity of the SIM Toolkit authentication. Same time the freedom of the iOS and Android app development brings great opportunities for even small companies to publish authentication solutions on the markets.

4.4 Chip authentication program

The Chip Authentication Program (CAP) is a technical specification developed by MasterCard. Basic idea behind it is to allow using banking smartcards for authenticating users online. Visa also adopted this technology under the name Dynamic Passcode Authentication (DPA). Details of the protocol are not public, but
researchers in Computer Laboratory of University of Cambridge reverse engineered the system and published a paper describing the protocol functional principles [31]. They also found out some weaknesses in the protocol.

When utilizing the CAP in authentication to online services, the end-user has the smartcard, a physical smartcard reader with number pad and display and a PC. PIN is used by the user to get access to the card after insertion of the card into the reader. Authentication is implemented by dedicated cryptographic challenge-response protocol between the online service (via PC) and the EMV card. The sequence is described in Figure 3.

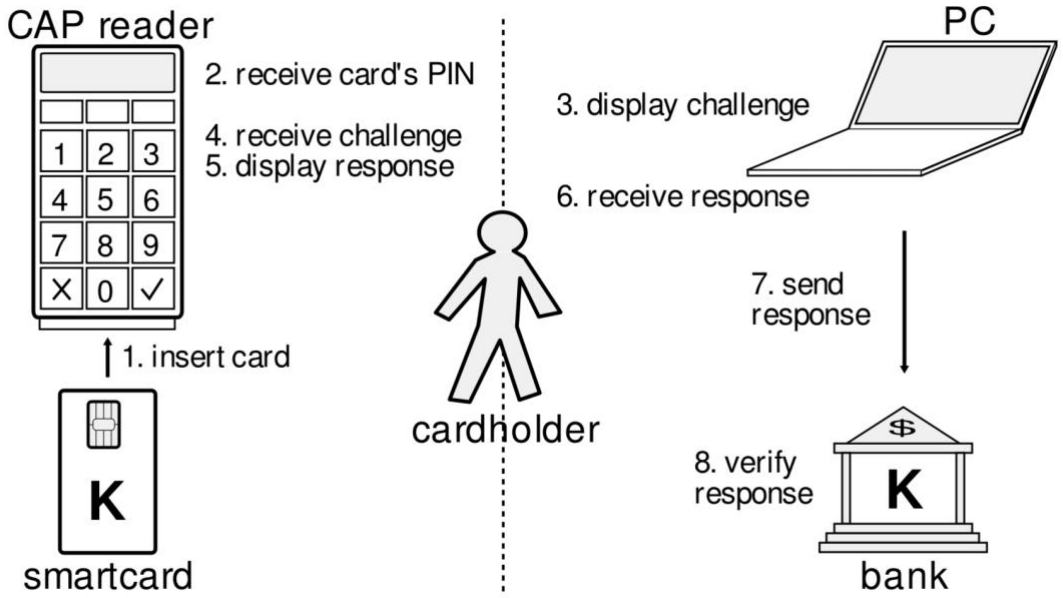


Figure 3 The CAP challenge-response protocol as described in the paper Optimised to Fail: Card Readers for Online Banking, Saar Drimer, Steven J. Murdoch, and Ross Anderson, 2009

4.5 Transaction authentication numbers

One 2-factor authentication method for online services has been the transaction authentication numbers (TAN). They are one-time passwords generated and printed to paper by the issuer. Transaction authentication numbers has been simple low-cost solution for banks [32].

TAN has usually an index and the matching number for the index. During authentication or transaction verification the online service presents the index number and the end-user must type a TAN matching the index.

Transaction authentication numbers can be used sequentially or by random selection of the index. When the index transmission is completed on an auxiliary side-channel like SMS or phone call, the authentication is seen to fulfill the requirements for a PSD2 SCA by the European Banking Authority [9] .

One drawback of the transaction authentication numbers is that they have quite limited amount of combinations due the amount of numbers used. So, the transaction authentication numbers are prone to brute force attack unless there are additional countermeasures deployed. In brute force attack the attacker may, for example, harness automated robots to try the authentication with random numbers. Also, the numbers can be copied and shared among users, which is a risk.

Typically, the input forms of authentication are throttled to mitigate the risk of brute force attacks. Throttling means that the system limits the amount of (authentication) requests allowed to enter into processing.

4.6 Short message service

The short message services (SMS) [33] of the mobile networks have lost popularity in common use as the 3G and later networks have allowed the usage of the internet protocols in mobile phones. Shortly, the internet displaced the short messages by bringing the rich multi-media content easily available for the end-users. However, the possibility to send short messages still exist in the smartphones.

As the mobile networks and SIM-units in the mobile phones are backed by the PKI system, the SMS has been seen as a reliable channel for sending one-time codes or challenge codes for transaction authentication numbers.

However, several successful attacks on the SS7 network as tracking of subscribers, eavesdropping and SMS redirecting was performed during year 2014 [34]. SMS brings vulnerabilities also in the mobile device. The SMS content can be read by various malware attacks inside the device [35].

In addition to the aforementioned pure technical vulnerabilities the SMS can be attacked with SIM-swapping that is human operated attack. In SIM-swapping attack the attacker gathers information about the victim and uses this information to impersonate him/herself as the victim when contacting the mobile operator. If attacker succeeds to convince the mobile operator personnel, the telephone number of the victim is associated to the SIM held by the attacker. After this the attacker receives the short messages that are intended for the victim.

National Institute of Standards and Technology (NIST) in U.S has addressed this potential attack scenario in their Digital Identity Guidelines 800-63B. They classify this attack under social engineering threats. NIST describes this attack shortly by following description. "*An out of band secret sent via SMS is received by an attacker who has convinced the mobile operator to redirect the victim's mobile phone to the attacker."* [36]

As conclusion, the SMS should be seen only as a weakly protected side channel for multi-factor authentication, not as a sole strong authentication solution.

4.7 Legacy and backup methods: security questions & email

Online users forget often their credentials to the websites. Traditionally this problem has been solved by backing up the account by weak authentication methods like private question/answer-pairs that are called "security questions".

When user has answered correctly to the questions he/she is permitted to enter the service and reset the lost password.

Security questions are nowadays even less secure than they were in the days when they were invented. As the personal information has become more public by various data leaks and by the possibilities of social engineering, the answers to security questions are too easily available. For example, Ariel Rabkin from University of California estimated in his survey about security questions among banks in U.S, that this method may become *dangerously insecure*. [37]

Email has also traditionally been used for the backup authentication method for online services. It has been used mostly for gaining access to the account when the credentials have been lost. The example sequence of the resetting logic is presented in Figure 4.

Email has major weaknesses regarding authentication. The access to the email account is typically device independent and protected only by a username and password. Passwords of email accounts also leak easily from misconfigured clients. NIST Special Publication 800-63 gives strong guidance that email should not be used for single factor or multi-factor authentication. [36]

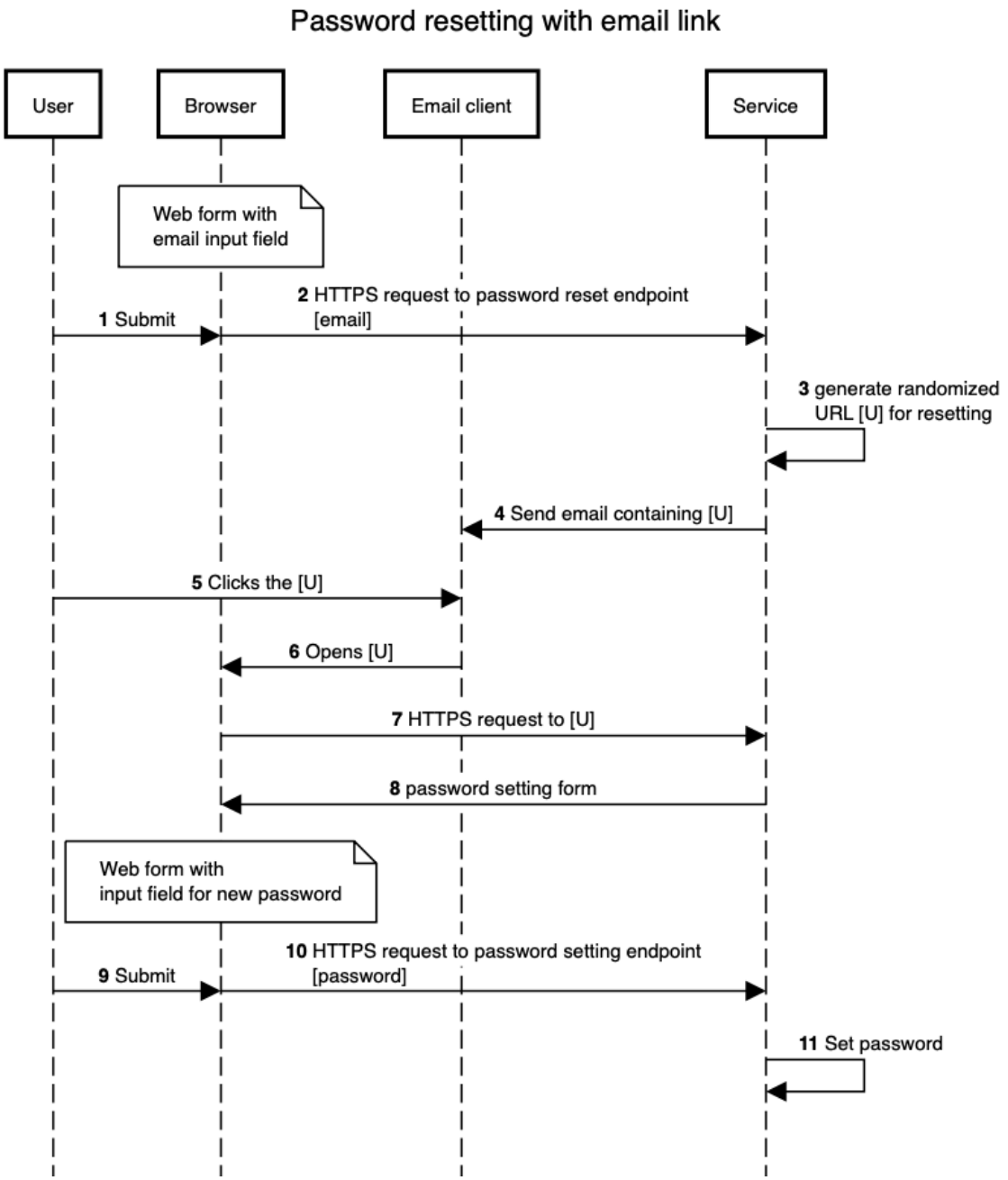


Figure 4 Password reset with email

4.8 Biometric authentication

Traditionally the authentication has relied on the secret information or on the possession of a device like hardware token or a smart card. Specially the need to carry an external authentication device has possibly been one motivation of inventing authentication schemes based on inherence. Obvious question is: Why to carry a device if *you* are the “device” that can be used to prove your identity.

4.8.1 Fingerprint reading

A long before the digital age, the fingerprint has been found as unique property of a person. Traces of fingerprint have been traditionally used for evidence collection in crime investigations. As the technology became accurate enough to scan the fingerprint reliably and the algorithms became quick and accurate enough to verify the human identity, the fingerprint became the dominant biometric authentication method in digital environments. [38]

Fingerprint reading has been combined in various technical form factors. One possible way to implement it is to add fingerprint reading pad into a hardware token. For example, Yubico has utilized this approach in their popular Yubikey series tokens. [39]

Some trials have also been made to combine the fingerprint reading to EMV card. Major credit card issuers like Visa and Mastercard have advertised about their development programs regarding this. [40] [41] This is natural development as the EMV cards have many similarities with hardware tokens regarding the basic functionalities, usage and enrollment.

However, the environment where the major success has happened regarding the adoption of this authentication technology is the smartphone. Today major part of the smartphones on the markets do have the fingerprint scanner. The scanning is used for many different use-cases starting already from the access of the phone. Also, the scanning is available for the mobile app makers by the APIs of the mobile operating system.

4.8.2 Face recognition

The history of the face recognition dates back to 1960’s when Woodrow Wilson Bledsoe developed a manual system that could classify photos of faces. [42] Wilson’s method utilized a special RAND tablet that was manually operated to locate certain facial properties from the pictures. The data of the processed image and associated identity were stored in a database. Later, the system could use a method based on distances between facial features to search the best matching picture from the database (compared to the provided picture).

Lots of development has happened since the first semi-automated face recognition attempts. Today the evolution of the computer vision technology and artificial intelligence has enabled deployment of the face recognition to various uses. Most common implementations are access control systems, for example, in airports. However, the most interesting use-cases for this thesis are the examples where the face recognition is used for web authentication or access to other digital services. Perhaps the most notable success in this category has been the Apple Face ID: the user authentication mechanism for certain high-end Apple smartphones. [43]

One challenge for the face recognition based authentication has traditionally been the liveness detection: the detection systems have been vulnerable to

attacks where the static 2D-image of the identity owner has been presented to the recognition mechanism and the authentication passed successfully. More advanced 3D face recognition methods have been developed to enhance the security. [44] Also the various sensors available today in smartphone environments have been explored as a solution to rise the security. [45]

5 Requirements for a successful authentication scheme

The replacement alternatives for the password authentication scheme have been explored tens of years. Researchers have found various schemes that are more secure than passwords. This is not surprising as it is widely known that the password authentication scheme has major problems regarding security. There are various on-line or off-line attack vectors against passwords. On the other hand, there are also user originated problem categories (password sharing etc.).

5.1 The four-factor model

Research made by Joseph Bonneau et. al. [46] [47] bring interesting views to the discussion about what are the factors that make the authentication scheme successful. One view is that research has possibly been focused too much on security. Schemes more secure than passwords have existed tens of years, but yet none of them is as widespread as passwords. The presented conclusion is that, to achieve worldwide adoption, the authentication scheme must be *secure*, highly *usable* and incur *low costs*. It is not enough to excel only in one or two of these categories. All three are necessary.

The tradeoff between security, cost and usability is mentioned also in the book LTE Security [48]. One main idea presented there is that the cost of implementing the security mechanism must be reasonable when compared to the value of the protected resource. Also, we must be aware that high amount of security mechanisms applied to a system may typically reduce the usability. It must be considered what is the threshold level of the worsening usability that should not be exceeded. How much can the end-user accept the inconvenience brought by the security?

In addition to this academic research, there is additional view brought by commercial strong authentication solution vendor Nok Nok Labs inc. They have pointed that the *privacy* is the fourth major factor that must be taken into account when designing an authentication scheme. [49] The criticality of privacy has risen after the General Data Protection Regulation (GDPR) of European Union [50] has been set into force. Especially the biometric data that is used to authenticate user must be protected adequately against theft.

As conclusion, the properties of an authentication scheme may be estimated or measured in four scales. The overall goodness of the scheme can be visualized by the Figure 5.

In the visualization, all the four factors have own scale running from center to the corners. Corners of the dashed rectangle indicate the value for each factor in their own individual scale. In the sample figure, the *usability factor* is relatively low and the *low-cost factor* is high (meaning that the cost is low). Total area inside the dashed lines expresses the score: the greater the area, the better. However, it is notable that the four scales in this model cannot be directly compared and cannot be used for accurate measurements. The model tries only to visualize the paradigm.

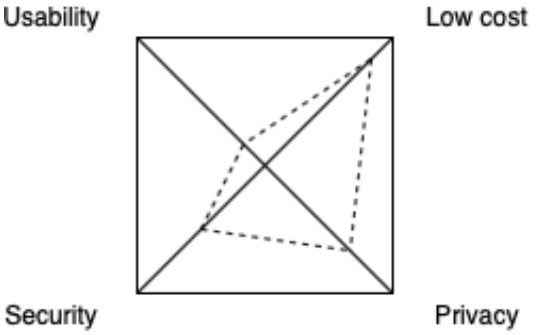


Figure 5: The 4 factors affecting to the popularity of an authentication scheme

5.2 Usability in the strong authentication

The usability of 2-factor authentication methods has been explored in various studies during last decade. [51] [52] [53] [54] [55]. Comparisons between different studies is difficult because of various reasons. The results can be seen contradictory and complex. Sometimes the research has been targeted to only single authentication method. On the other hand, the contexts are different among studies. Research has been conducted in various environments like banking, university et cetera.

Some observations have been found out in all sources: The 2-factor authentication seems to feel annoying for the end-users irrespective of the method.

Common recommendation for the 2-factor authentication implementations is that the number of the steps should be kept minimal. [54]

There was some evidence supporting the hypothesis that the solutions based on the push notifications were most popular among the users. For example, Jessica Colnago et al from Carnegie Mellon University and University of California found out in their study [54] that the three most popular methods used were push notification (91%), app-generated passcode (21%), and hard token (4%). Push notification was here used to invoke an authenticator app, Duo Mobile. Also study

from Ken Reese et al from Brigham Young University found out in their usability study of different 2-factor methods that the push notification approach (authenticator app) and U2F security keys helped the end-user to reduce login time [51]. Also, the push notification approach took the second least set-up time. Paper based transaction authentication numbers were faster to set-up in the study.

In addition to these results presented by academic groups there exist some commercial survey material about the usability in the authentication. One interesting survey is made by Duo Security inc. [56] They found out some contradictory results between the usability of push notification-based approaches and security keys. On the other hand, the security keys enjoyed greatest number in the question “I enjoyed using it” but also greatest number for the question “using it was stressful” or “using it was frustrating”. However, the push-based approach was successful in many questions indicating good usability.

Based on these surveys we set a question: is the push notification a critical factor in success of mobile app authenticators? On the other point of view, the push notification can be seen also as a feature that may reduce the security. End-users may accidentally accept transaction by habit if a hacker is able to initiate the transaction by guessing the username, phone number or other weak id of the end-user. Identity phishing is a rising concern for commercial services.

5.3 Cost, privacy and security

Security is very fundamental factor for an authentication method. It is therefore discussed further in dedicated chapters later in this thesis. Also, the value of privacy has also risen lately. In particular, the European general data protection regulation (GDPR) has set new standards for the privacy in the online services [50]. Due the limited scope of this thesis we must leave the privacy out of further examination. The same scope limitation applies to the cost factor. Nevertheless, the cost is important when we look for wide adoption and possible commercial success of an authentication solution.

6 Architecture of the mobile app authenticator

As stated before, most of the people in the western world are smartphone users [1] [2] today. Mobile apps are becoming the common way to solve everyday tasks like making daily purchases, managing financial affairs or maintaining social relationships. Many of the daily tasks require authentication. Many tasks require using strong authentication for the security reasons. Result of this long-term development has been that the mobile app makers have started to include strong authentication features to their systems or even developed separate purpose-built authenticator apps.

Also, companies doing business in the security and IAM sector have notified their opportunities here. Implementing the strong authentication properly requires special skills that regular mobile app makers do not necessary have. The following chapters illustrate the common concepts and building blocks of the mobile app authenticators and try to address the challenges that the developers may meet.

Mobile app authenticators can be built on many different architectural models: basic client-server model, edge computing model and peer-to-peer overlay networks, for example.

Computing power of the mobile devices gives much freedom for the mobile app authenticator designers regarding the architecture. Also, the wide scale of the services available by the mobile app platform may steer the designer to implement many computational tasks in the mobile device instead of the server. When we compare this, for example, to mainframe computing systems that were popular in industrial and financial institution use during many decades we see the change of the paradigm. On the other hand, battery life is one of the major concerns in mobile devices and this may slow down the development of pure peer-to-peer authentication solutions. [57]

6.1 Peer-to-peer architecture

One of the interesting questions in the peer-to-peer of architecture is the source of the trust. If the designer of an authentication system see that the mobile app authenticator should have no central root of trust and the authenticators should be equal peers in the network, the architecture can be based on distributed ledger, as in blockchain architectures.

One special requirement for a peer-to-peer architecture for mobile app authenticators is that there must exist special proxy nodes in the architecture in addition to the peers (mobile apps). Reason for this requirement is that the mobile phone networks do not support interconnectivity between the peers (by other than voice calls and SMS-messaging). A mobile phone and the apps running in it cannot act as a server because the inbound messaging on suitable protocols may be very limited by the network.

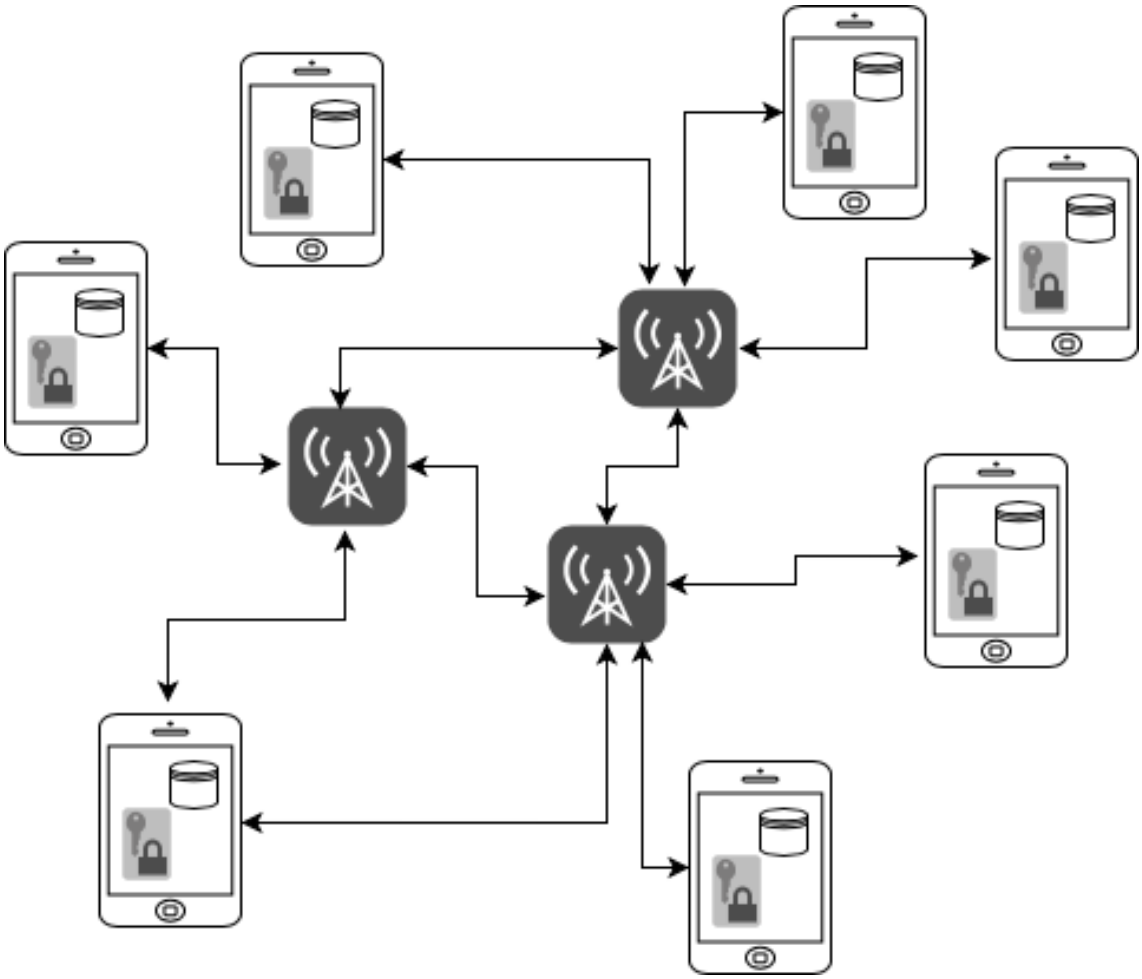


Figure 6 Generic sample architecture – peer-to-peer overlay network of mobile authenticator apps and supporting proxy nodes

In peer-to-peer architecture it is also notable that the system data may be distributed. In distributed model the data is not stored in a central data source where all the peers should connect in order to make queries or updates. Instead, the peers (all or special nodes in the overlay network) hold the data by themselves. How data is distributed, affects to the data redundancy, query latency, and data availability and other major architectural properties of the system, like scalability.

Hany F. Atlam et.al stated that the “large scale” is a challenge for the IoT (Internet of Things) networks in cloud computing [58]. The interconnecting activities among the devices (peers) require carefully designed algorithms and data distribution design to perform in the large scale.

Benefits of a peer-to-peer architecture are, for example:

- No central authority is required
- Control of the personal data may stay in the peers
- System may support peer anonymity

- Data may be distributed among the peers, that may allow faster response times for data queries.

- Computational efforts may be distributed and scheduled evenly across the peers

Disadvantages of a peer-to-peer architecture are, for example:

- Intercommunication between the peers may require noticeable amount of network traffic

- Data redundancy may be high unless the distribution algorithm is carefully designed

- System scalability is bad due the abovementioned reasons

- Anonymity may support criminal activities

Arguments presented above may however be questioned also. For example, the lack of a central authority can be seen as advantage or disadvantage depending on the point of view.

6.1.1 Distributed ledger and the blockchain

Blockchain [12] [59] has been explored intensively for the IoT (Internet of Things) use during the last decade [60]. Reasons leading to the rising interest of blockchain solutions have likely been the concern about common security in internet and the dominance of the major companies doing the business there (like Apple, Google, Microsoft, Alibaba and Tencent).

Basic idea of the blockchain is that the data is stored into nodes of a linked list. The nodes of the list are incrementally and cryptographically signed. No node can be altered in the chain without violating the cryptographic proof of the chain authenticity.

One well-articulated and simplified description of the distributed ledger concept is written into the developer documents of IBM. They define it as *"type of database that is shared, replicated, and synchronized among the members of a decentralized network."* [11] Distributed ledger and the blockchain can be used together to create a distributed authentication system. Many times, especially in the commercial publications, the concept of distributed ledger is left out and only the popularized term blockchain is used to describe a system that is based on both distributed ledger and blockchain.

One of the main benefits in the combined distributed ledger and blockchain concept is that the trust of the system and the peers is distributed; no central

authority is required to ensure the authenticity of the peers or the stored data [60] [61]. The trust is thus de-centralized.

In popularized publications related to “blockchain” the data is generally mentioned to be *distributed* into the peers. Distribution is generally speaking beneficial for networked systems, for example, in order to increase computing parallelism and system performance. However, the concept of *distributed data* may be implemented in various ways. In some cases, the term *distributed* may refer to a model where the peers do not hold same data ledger, but a small portion of it instead. Sharded Merkle tree [62] would be an example of this. On the other hand, the term *distributed* may refer to a concept where the system data is replicated to identical copies held by all the nodes.

Scaling is the major challenge for the peer-to-peer architectures. Blockchain and distributed ledger technology is not an automatic solution for the scaling problem. For example, the most known real life blockchain implementation, virtual currency Bitcoin [63], has performed poorly from the point of view of scalability [64].

One of the biggest problems of the scalability of Bitcoin is the ledger distribution. In Bitcoin ecosystem, the ledger is copied to all mining nodes in the overlay network. The concept of mining refers to certain computational work that is operated by the mining nodes to prove the existence and authenticity of transactions in each block of the blockchain. Mining will not be described here further, but it generates the trust in the Bitcoin ecosystem.

If all the peers must have the same redundant data about the all transactions made in the ecosystem, the system scales up badly. Anamica Chauhan et. al explored [65] the Bitcoin scalability and compared it to enhanced scaling concepts of the Ethereum. The results indicate that the ledger sharding mechanism enhances scaling of the blockchain system.

6.1.2 Crowd-trust

Another architecture candidate concept for a peer-to-peer authentication system could be a crowd-trust model. The concept *crowd-trust* means here a digital reputation value of the system members (peers), that is earned by manual reviews by the peers or by some automatic algorithmic approach of the system analyzing the transactions made among the peers.

In this architecture, the peers would not be equal in terms of their reputation. The peers would gain reputation by their transactions and the feedback received either from the other peers (“crowd-trust”) or the system itself. The trustworthiness of their claims about transactions would be a probability value calculated by an algorithm. One of this kind of trust management schemes is presented by a study by Jaydip Sen [66].

6.2 Client-server architecture

Usually the authentication platform that utilizes the mobile app authenticators is based on a model in which the authenticator apps are most of the time in contact with some central authority (client–server model).

The typical architecture variants of the mobile apps are native client and web view based architectures (and any hybrids between these). [67]

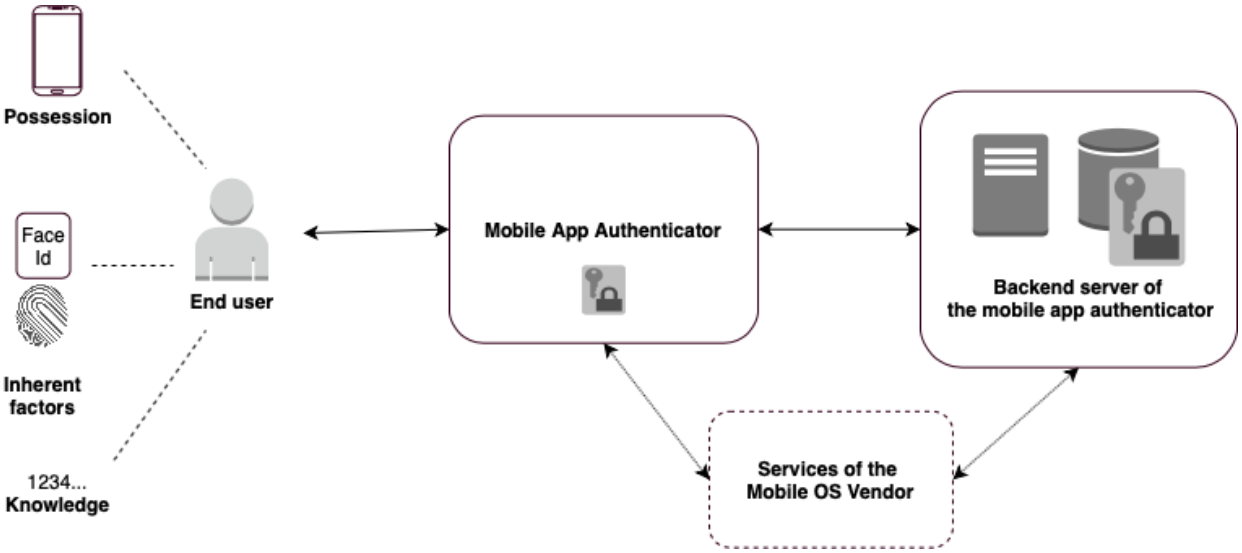


Figure 7 Simplified mobile app authenticator architecture based on cloud server backend and rich client.

6.2.1 Web view based mobile apps

Some could say that the easiest way to implement a mobile app today is based on web views. In this model, the app utilizes a standard web browser component supplied by the platform or common software development kits (SDK). The browser component loads content (for example HTML5) from a pre-defined URI and renders it to the screen. Major parts of the app logic are usually all coded into the HTML-content and runnable scripts (JavaScript) included.

Web views are easy to implement for multiple mobile platforms simultaneously, which is a benefit. Also updates to the app may sometimes be handled by updating the HTML-content, so control is at the server side. Updating the app at device is not necessarily needed. [67]

However, web views have some limitations that makes them less suitable choice as basis for an authenticator app. Typically, an authenticator app may need data or inputs from the sensors of the mobile device. The app may need access to native authentication services supplied by the platform (like fingerprint reading or facial recognition). Some years ago, when web views came available there was a great challenge to reach these functions from the web view. W3C has later published some libraries and common sensors API for this. [68] Despite the

development of these functionalities in the mobile platforms, the complexity of reaching the native functionalities from webview seems to have stayed.

Another challenge for a web view is that the authenticator app may need a secure persistent storage. For a mobile app authenticator, the word secure requires using encryption over the data staying in the storage. Access and management of cryptographic keys and using the cryptographic algorithms securely may be a challenge for a web view.

6.2.2 Native mobile apps

Native mobile apps are apps that are written directly for the target platform [69]. They are developed on programming languages like C and Java. Native apps render their user interface views without specific structural instructions like HTML. Native apps benefit from the features published by the operating system. Mobile platforms like Android and iOS supply today a great number of APIs and libraries for the developers of native apps.

Downside of the native apps is that the app must typically be programmed independently to all different mobile platforms. Even there may be some common components, major part of the code cannot be shared due the fundamental differences of the platforms. Also, whenever there is a need for a functional change for the app, the app must be developed, tested and published through the development and publishing platform of the platform vendor. Publishing and acceptance may take days by the platform. Slowness is not optimal, for example, for critical security updates.

Native app implementation strategy has major benefits in the authenticator app case. The tooling for making the apps, namely the libraries and APIs, are comprehensive. As mentioned in previous chapters, the access to sensor data and authentication functions like fingerprint reading, input of the access code, or facial recognition is important.

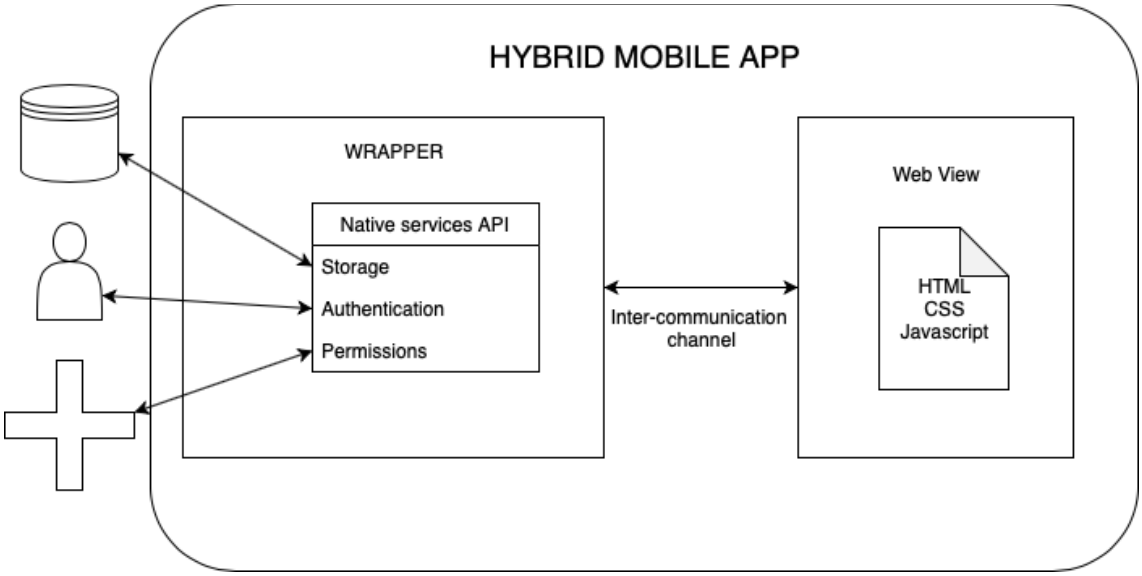
Native low-level programming enables using security features like white box cryptography [13] that gives protection against many typical activities of the hackers. These activities include static reverse engineering of the app binaries to source code (that enables access to logic of the app), reverse engineering of the memory space of the app during runtime, and various other attack vectors.

6.2.3 Hybrid mobile apps

In the hybrid model, the app contains a wrapper and a web view. It is questionable if there is any distinction between web view based app and hybrid mobile app. Classification is not clear. One possible distinctive feature could be the amount of the logic located in the web view. If there exist major logical parts in the both areas (native and dynamic web content) in the app, it could be claimed a hybrid app.

The wrapper

The wrapper is a native part of the app and acts as a core for the web view. Wrapper may supply an API containing the access to the native platform features. [67]



Pros and cons of the web view

Both app categories have advantages in certain areas. However, in the authenticator app use the web view should be used only for functionalities that are not security critical. These are, for example, showing the terms of service.

Table 1 Differences of the app categories. Source: *Shruthi Sasidaran, Survey on Native and Hybrid Mobile Application Development Tools 2017*

	Native	**Hybrid**
Development language	Native languages: Java for Android Swift for iOS	Native and web / web only
Device specific features	High	Moderate
Code portability	None	High
UI/UX	High	Moderate
Advanced graphics	High	Moderate
Application store	Available	Available
Development cost	Expensive	Reasonable
Device access	Complete	Complete
Speed	Fast	Medium
Access to native APIs	High	Moderate

6.3 Secure communication channel

If we think the very basic requirements for the communication between the parties in the authentication system architecture, they could be following terms: mutual trust and confidentiality.

There must be mutual trust between server and clients: mobile authenticator app must be sure that it is connected to a valid server and the server must be sure that the connected client is not a rogue one. If the architecture does not have a central authority, the same principle is still valid: the valid peers must have trust mechanisms to protect them from rogue peers.

The system must preserve the data confidentiality i.e. it must be protected against eavesdropping. Typically, the authentication system handles personal data (even an authentication system based on pseudonymous identities if fully valid concept as we know from virtual currencies, for example).

Naturally the communication channel and the protocol must suit the messaging design of the system. Some may prefer continuous stream of small messages sent on lower layers of the OSI model while other may find the classic sparsely used request/response pairs over HTTP adequate.

6.3.1 Networking protocol

Authenticator app designers have some alternatives for the networking solution between the app and the server. While the TCP could be good choice for a continuous type messaging between the app and the server (like is desired in gaming) the typical networking activity for an authenticator app might be just a certain sequence of request/response –pairs in a limited time scale. HTTP (over TLS) is good match for this.

TLS

Using transport layer security (TLS) [70] has become a de-facto security standard in various uses of HTTP. TLS utilizes asymmetric cryptography to protect the messaging confidentiality and can also optionally support mutual authentication between the messaging parties. Usually only the server is authenticated in web browsing usage, but in case of mobile app authenticator there must be a cryptographic mechanism to prove the identity of the client too. TLS client certificates is good candidate for this use.

The threat of a man-in-the-middle attack

While the TLS supplies principally very good security for the messaging between the mobile apps and their corresponding server back-ends, the proxy setups with TLS termination may dilute the situation.

A proxy server, as used in application layer communication in mobile networks, is a gateway that intercepts all traffic between the mobile device and public internet [71]. Proxy can also exist inside the mobile device as a software component that intercepts the traffic between apps and the network.

In both cases the operating system and the networking components of the client can be configured (with the acceptance given by the end-user) so that the encryption of the TLS is decrypted in the proxy. Originally these proxy mechanisms have been built for good intentions like content filtering, bandwidth usage limitations and privacy but it contains also the risk for misuse in form of a man-in-the-middle attack in certain conditions. [71]

One possible mitigation method against the man-in-the-middle threat is *certificate pinning* combined with an *additional authentication and encryption layer* implemented over the TLS networking. Also, this method should be built using white box cryptography at the client side to additionally slow down the efforts of the possible attackers.

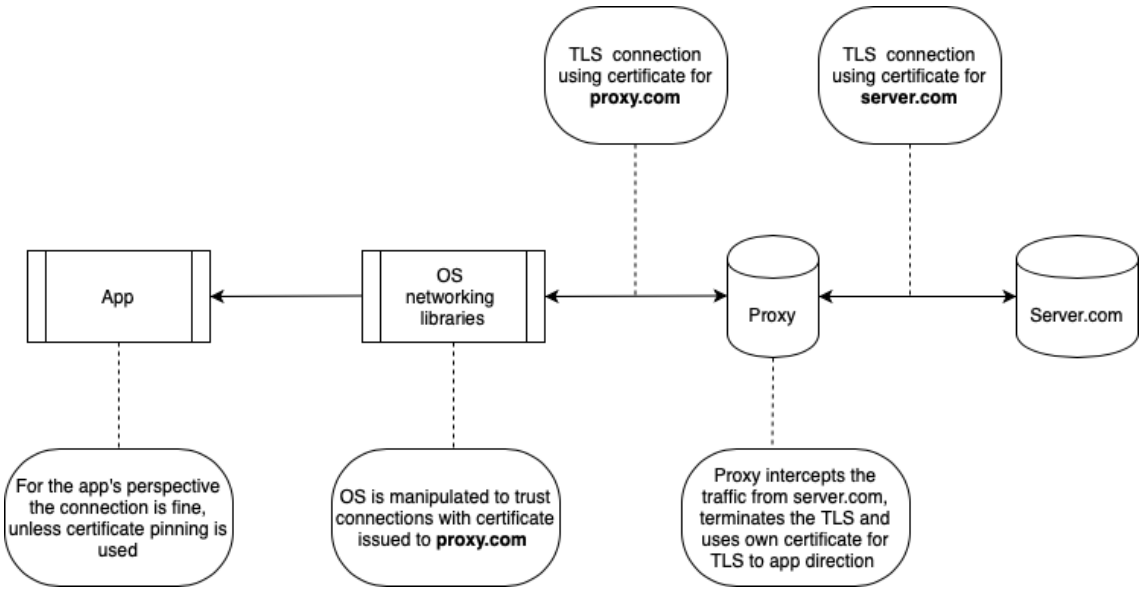


Figure 8 Traffic eavesdropping with a proxy and TLS termination

Certificate pinning

The main principle in the certificate pinning is that the client application ensures after successful TLS handshake that the presented certificate of the server is issued on a previously known or pre-configured public key. The assumed public key can be even hard-coded into the application if needed. Also, the client may be programmed to accept multiple alternative keys from a key set [72]. This allows the system, for example, to rotate the keys used.

Certificate pinning can also be dynamic in a way that when a trusted connection between the parties has been established, the next assumed public key can be sent to the client and then stored securely. When the certificate pinning is used the requirements of using secure storage and app shielding technologies become important. This is covered in later chapters of this thesis.

6.4 Authentication server

One obvious architecture for an authentication system is the client-server architecture. The server, which is called *authentication server* in this thesis, has various important purposes in this architecture. Example of a generic authentication server architecture is presented in Figure 9. Concepts and the main components of this architecture are presented in the following chapters.

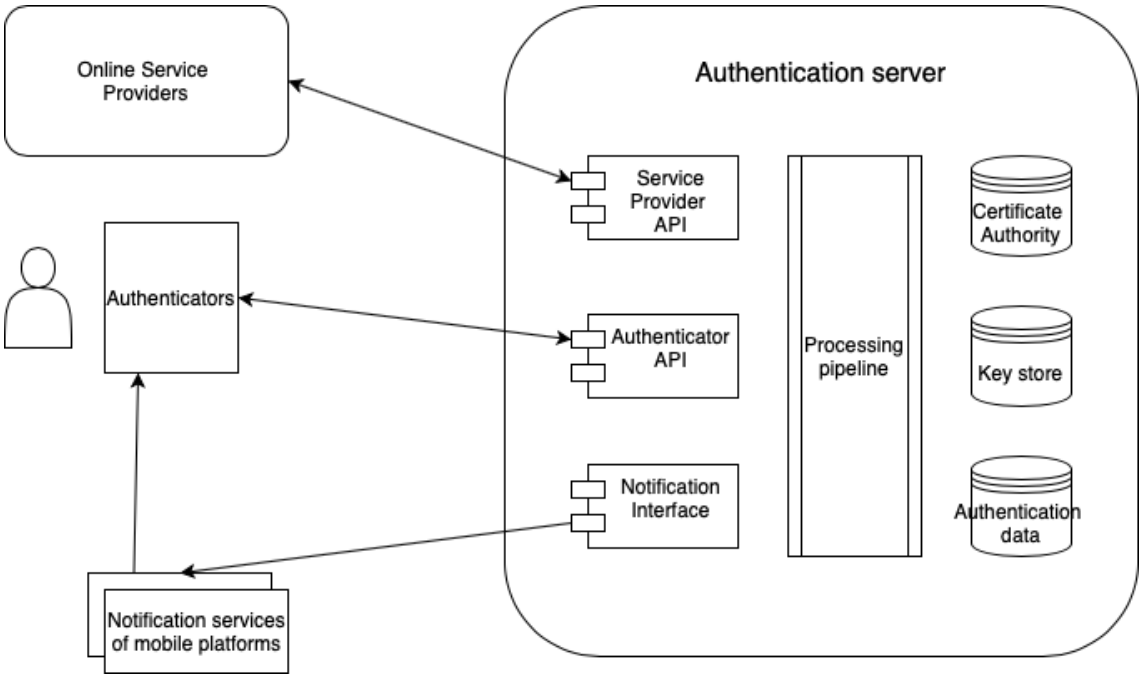


Figure 9 Generic authentication server architecture

6.4.1 Service Provider API

Especially in the large enterprise architectures it is common that the authentication services are centralized to a single system. For example, a bank can have arranged the authentication of the customers so that they can use a mobile app authenticator to login to daily banking services and to stock trading service that are two totally different software systems.

The aforementioned arrangement can be implemented by federated login concept. If we continue using the banking as example, the daily banking service application and the stock trading platform are *relying parties* for the authentication service that is the *identity provider.*

The relying parties need to communicate with the identity provider in order to handle the tasks related to the authentication. In this example architecture, the Service Provider API is the interface that is used by the relying parties (service providers). The Service Provider API has endpoints for managing the end-user identities and associations. Also, it has endpoint for managing authentication events.

It is beneficial for the Service Provider API, if it is designed to follow common standards regarding the interface specifications and the communication protocol. This is important because the standardized frameworks are typically well designed and reviewed regarding the security. Additionally, this may bring also cost reduction. Developers of the client applications may possibly utilize common purpose-built programming libraries instead of building all from the scratch.

For federated identity management, there are some useful frameworks and standards that can be used with the Service Provider API. For example, Security Assertion Markup Language (SAML) V2.0 was published year 2005 [73]. It is still commonly used by many organizations even it may be seen technically slightly outdated. However, newer generation standards like OpenID Connect (OIDC) [74] are currently preferred for the Service Provider API.

6.4.2 Authenticator API

The mobile app authenticators need various services through their life cycle in the client-server architecture. In this example architecture, the Authenticator API is the interface that supplies the services. The API can be structured various ways. One possible solution is to arrange the API into endpoints based on the use-cases they are related to. For example, fetching open authentication requests and submitting the signatures can be served from the authentication endpoint as they belong to the authentication use-case.

Registration endpoint

The authenticator app needs to connect to the server immediately when the required cryptographic keys (we assume that the PKC is used) are generated. The authenticator must register the generated public key to the server for later use. Possibly the authenticator and server exchange other keys, attributes or shared secrets too depending on the implementation details. The registration endpoint of the Authenticator API supports these requirements.

Authentication endpoint

The authentication endpoint is needed for the authentication flow. For example, the mobile authenticator app needs to fetch the ongoing authentication requests that are sent by the service providers. Secondly the committed authentication requests and the cryptographic signatures must be sent back to the server for

verification and further processing. Simplified generic authentication flow is presented in the Figure 10.

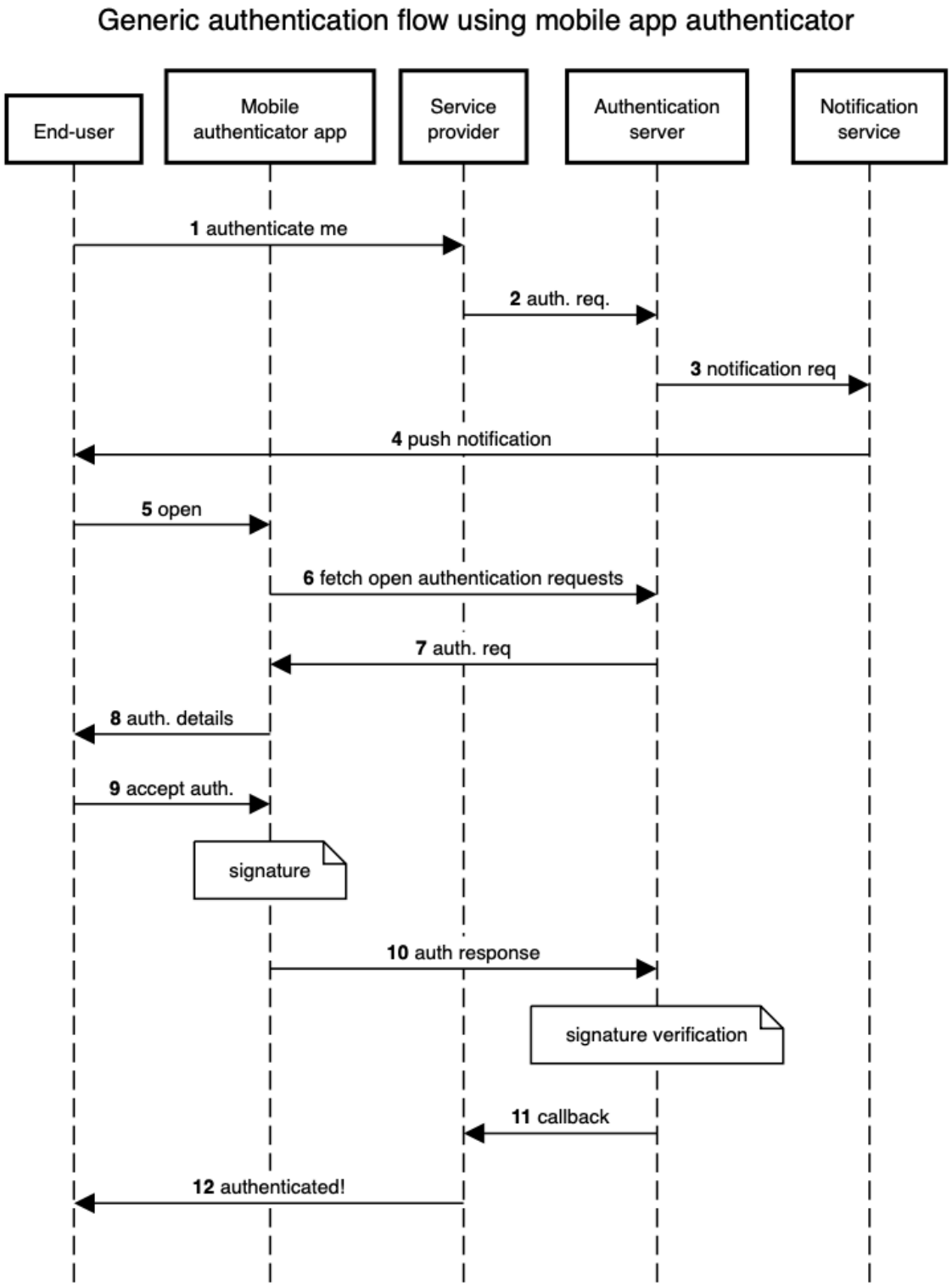


Figure 10 Generic authentication flow using mobile app authenticator

7 Challenge to bootstrap the user identity

Initialization of a mobile authenticator app securely is itself a challenging task. Vulnerability or weakness of any component in the mobile authenticator app ecosystem is a potential risk for fraudulent use or a data breach. The old cliché quote says that the chain is as strong as the weakest link in it. This saying can be applied to an authentication system where authentication events must leave an uninterrupted audit trail.

7.1 The authenticity of the authenticator

We have a generic problem about how to ensure that only valid client can be enrolled into an authentication system. A potential threat scenario to an authentication system is that a hacker with appropriate knowledge and tooling can listen the traffic and reverse engineer the communication protocol between clients and the server. By this information the hacker can either impersonate an existing client or deploy new fake clients into the system.

The solution against eavesdropping and theft of the confidential information is that the clients have a strong shared secret between them and the server. This shared secret is used to prove that the newly deployed client is valid and connected to a valid server counterpart. Additionally, the shared secret is used to generate encrypted messaging channel. In a nutshell, the shared secret is the root where the higher layers of security are based on.

The problem of the bootstrapping is, how the new client can have the secret already when it is generated and how the secret is stored securely. Traditionally, in the mobile phone systems like GSM or UMTS the mutual trust and confidentiality between the user equipment (UE) and home network (HN) has been built on a subscriber authentication key stored on a tamper resistant SIM-card. [75] Naturally the key generation must happen in security-controlled factory environment.

The SIM concept has similarities to hardware security keys: they also may have hardware backed shared secrets deployed already from the factory.

7.2 The root of trust

The root of trust problem domain is a major challenge for a mobile app authenticator. The low-level mechanisms that secure the subscriber in a mobile network are not generally available for the third-party mobile app makers. The root of trust must be based on something else.

In many mobile phone models, there exists a hardware backed isolated TEE (trusted execution environment). TEE is separated from the normal processing environment where the operating system and applications run [76]. As example of TEE, Apple has Secure Enclave [77] present in modern iPhones. Microprocessor vendor ARM has their TrustZone technology that can be utilized for the TEE implementations in mobile phone environments [78]. However, it is claimed that most of the smartphone and tablet contains a TEE today [76].

TEE ensures that the code and data inside of it are protected with regarding confidentiality and integrity. Client applications of the TEE receive services like storage layer security, secure isolated cryptographic operations and secure timing services.

It is important to note that the mobile app authenticator should generate the needed cryptographic keys inside a TEE. But possibilities to do so depend much on the APIs and the security mechanisms of the mobile operating systems as the developer of a third-party app does not have direct access to hardware. Fortunately, iOS of Apple and Android support the TEE through their key management systems.

It is notable that generating and using cryptographic keys in a TEE helps to enhance the application security during runtime. It also enhances security of the communication protocol between client and server. But it does not prevent generation of fake clients: there is nothing that could allow the server to distinct a fraudulent client app from a valid one (assuming that the fake client can talk to the server with the protocol required). TEE is just a mechanism that may help preventing the identity thefts in some scenarios after the app is installed.

8 Security of the mobile apps

As mentioned before, the mobile app platforms and ecosystems evolve currently with high speed, and the security has problems to follow. Fortunately, the open source security community, academic research and the security business players on the market try to minimize the gap vigorously. One example of this threat mitigation activity is the OWASP Foundation that publishes the common threat check-lists to arise awareness among the app developers and businesses.

OWASP Top 10 Mobile risks were 2016 [79]:

M1: Improper Platform Usage
M2: Insecure Data Storage
M3: Insecure Communication
M4: Insecure Authentication
M5: Insufficient Cryptography
M6: Insecure Authorization
M7: Client Code Quality
M8: Code Tampering

M9: Reverse Engineering
M10: Extraneous Functionality

In the following chapters, we discuss some of these aforementioned threats that the mobile applications are facing.

8.1 App shielding

Vulnerabilities in mobile environments may in some circumstances allow a hacker, foreign malicious app or malware to read the secrets of the mobile app authenticator from the runtime memory or from the persistent memory.

Another common attack against mobile apps is repackaging. In repackaging the malware writer alters a legitimate application to include malicious code and then publishes it to an app market or even injects the additional functionalities on-the-fly [80, 81]. So-called rooted or jailbroken operating systems are especially vulnerable for this approach.

The rooting (Android) or jailbreaking (iOS) is a method to alter the operating system so that the various security related restrictions for the apps are removed. Typically, the apps are let to run on root (administrator) user privileges. This removes the boundaries of the app isolation of the operating system. Root user privileges permit reading any file in the persistent storage no matter which app owns it. The motivation for the rooting or jailbreaking may be that the end-user can, for example, load unsigned applications from illegal sources outside app stores. This has traditionally been popular in China.

App shielding is generic term for mitigation methods against these aforementioned scenarios. App shielding typically includes methods like obfuscation of program code, white box cryptography and anti-tampering technologies. [82]

8.2 White box cryptography

When the white box cryptography is not applied, the cryptographic keys are located as sequential bytes in the memory space of an application during runtime. As stated before, the bytes can be read in some circumstances and this means theft of the confidential cryptographic secrets.

Purpose of the white box cryptography is to prevent the exposure of the cryptographic secrets [13]. To reach this goal, the keys are obfuscated, stored in a distributed format in various memory locations (as non-sequential form) and the obsolete redundant data may be added into the keys. Additionally, the application code that executes the cryptographic algorithms is obfuscated and modified so that it is extremely difficult for human to read. For example, typical AES symmetric key encryption and decryption contains execution of loop over certain index. This

can be transformed to “flattened” lookup table based execution that is more difficult to reverse engineer.

Implementations of white box cryptography can also protect the integrity of the executed binaries. The binaries can include multiple (even overlapping) hash functions that check during execution that the original code is not altered. Typically, when the integrity check shows positive finding about tampering, the application code may be forced to halt immediately to prevent the misuse.

One interesting question is, whether the white box cryptography brings solution for the root of trust. White box cryptography allows application developers to inject keys and cryptographic algorithms into the application binaries with high confidence that the secrets are not detectable with reverse engineering methods. However, the key is not unique for each application and in no way tied to the device. It is questionable that the generic key alone could be a root of trust.

8.3 Obfuscation

The term obfuscation means manipulation of a program in such way that it keeps the original functional capabilities, but the manipulated program code becomes non-understandable by human.

Boaz Barak et al. provided definition: *“Informally, an obfuscator O is an (efficient, probabilistic) ‘compiler’ that takes as input a program (or circuit) P and produces a new program O(P) that has the same functionality as P yet is ‘unintelligible’ in some sense.”* [83]

Obfuscation can be seen as more important treatment for the Android apps than for iOS apps. The Android apps are typically made with Java –technology that is easily decompiled and reverse engineered with static analysis.

8.4 Anti-tampering technologies

The application can be tampered with various mechanisms by a hacker who wants to first reverse engineer the authenticator app and perhaps design an attack against it. At the end, the goal is to make fraudulent payments or to get access to some critical systems. The publisher of the authenticator app wants that the app is running in a safe environment enjoying the sandboxing of the operating system and all other security features designed by the phone manufacturer.

8.4.1 Rooting and jailbreak detection

As stated before, the rooting or jailbreaking makes the running environment fundamentally unsafe. The rooting or jailbreaking must be detected in order to estimate the trust that can be given for the authentication events.

Rooting and jailbreaking can be detected in some reliability level by using various checks made on the running environment. False positives are possible as the accuracy of the detection vary, and so multiple checks must be used simultaneously. Most of the detection methods rely on the basic idea that the app should not reach certain resources that are normally protected by the running environment. For example, app should not be able to add or modify files in protected areas. Also, successful attempt to read system files that require root privileges reveals that the operating system has probably been tampered.

When the rooting or jailbreaking is detected during runtime the app may react to the situation by pre-defined way. Simplest method is to halt the app abruptly leaving no room to operate for the hacker.

8.4.2 Guarding

The app can be protected so that it calculates checksums over the application binaries during the running and so detects if some of the application fragments are changed. Typically, the calculations of the checksums have overlapping regions so that the application alteration without getting caught becomes very difficult. Desired reaction when the alteration of the binaries is detected may vary. The reaction can also be slightly delayed and so causing additional difficulty for the attacker to reverse engineer the logic. [84]

8.4.3 Encryption wrappers

In the white box cryptography, the basic idea is to prevent the secret information to be readable in clear text. Similarly, in using the encryption wrappers the principle is to prevent the software binaries to exist in cleartext format allowing unauthorized static analysis. With encryption wrapper, the software is normally encrypted and only decrypted dynamically at runtime. [84]

8.5 The reality about security of the mobile apps

The security level of the financial mobile apps has been under interest of the security consultant companies for many years (since invention of the mobile banking apps). The point of view is of course commercial and the motivation for the information gathering is to sell consultancy. But despite the background and biased opinions, the findings of the surveys are noteworthy. They conclude that banking apps are generally speaking insecure.

Accenture and NowSecure Lab explored 30 banking apps from North America in 2016 [85]. They downloaded the apps straight from the app stores and performed 780 tests for the apps. Both major mobile platforms were included (iOS & Android). Some of the findings are collected below.

- 60% of the apps were not obfuscated in Android.
- 13% of the apps did not use certificate validation in networking protocols.
- 33% of the apps used world-writable files

Another example of commercial research is the study made by joint effort of crypto & security consultancy vendors Inside Secure and UL [86]. They selected 19 apps made for consumer use. Apps were from various geographical markets. The study did not target to any specific size of banks and there were well established companies and new small challenger banks among the publishers of the apps.

The key finding from this study was that only 5% of the apps reached the security level that was used as benchmark, namely Visa & Mastercard security standards for mobile payment applications. Majority of the apps had very low security levels.

The requirements for the benchmark level were (in a nutshell) [86]:

- Code handling sensitive data and algorithms is developed in a language that compiles to processor native machine code (i.e. C/C++)
- Strong obfuscation of all critical code
- Anti-tamper protection of the application
- Cryptography protected by whitebox (or equivalent technology)
- No sensitive text visible in static analysis of code
- Network traffic encrypted using TLS 1.2 and downgrade not possible
- Certificate pinning applied to networking
- Strong device binding

An interesting research topic could be to study which were the reasons that led to the design decisions regarding the application security. The four-factor model could probably be used for this research. Unfortunately, we must leave this topic out from the scope of this thesis.

8.6 Summary about protecting technologies

The highest protection levels are achieved when multiple protecting technologies are used together and in synergy. It is possible to achieve a level where it is fairly unpractical and unbeneficial to try to arrange an attack against the app.

Unfortunately, the app shielding methods typically increase the size of the binaries and slow down the application execution as the control flow is altered. Also, the steps required to generate obfuscated and shielded binaries require extra effort from the developers. Debugging the errors in production environments may be extremely complex with shielded apps. These may slow down the motivation to adopt the shielding technologies for authenticator apps.

9 End-to-end security of the system

When the authentication system is based on the client-server model, there must be layers of trust all the way from the client device hardware layer to the application server backend and to the app publishing system of the mobile platform. This approach is described in the corporate security as an onion model [81]. Basic idea of the model is that the security of the system is based on logical layers. However, the onion model security can be challenged also. The system architecture does not always follow layered design and can include also single point of failures.

9.1 The viewpoint of an attacker

One possible way to examine the security of an authentication platform is to look from the viewpoint of the attacker. And the onion model may help to visualize this viewpoint. Attacks against mobile authenticator app ecosystem may be classified to several main categories:

- Attack against the user of the mobile authenticator app
- Attack against the execution environment of the mobile app
- Attack against the communication channel that the app is using when it is connecting to other clients or servers.
- Attack against the application server
- Attack against the mobile app development
- Attack against the publishing framework

All attacks must be taken into consideration when designing an authentication system architecture. The onion model can be challenged regarding the possibility of multiple simultaneous attacks. The model is based on assumption that the attacks are sequential.

9.2 The PIN lifecycle model

One commonly used concept for knowledge-based authentication factor is personal identification number (PIN). This number is traditionally used with smartcard-based credit cards but also for unlocking the SIM card (subscriber identification module) of a mobile phone. During last decade, the PIN concept is also adopted to mobile apps. App may require an authentication step before it is permitted to start, or app may require a step-up authentication [87] when user is doing a critical transaction that only he/she is entitled to.

Because of long history of PIN, lots of research and standards are related to banking sector. Same basic usage principles and dynamics can be applied to the mobile app authenticators as to the smart cards.

One noticeable standard for the PIN security is the VISA PIN security guideline [10]. It presents a reference model for the security considerations based on a PIN lifecycle. Many of these aspects are valuable for the mobile authenticator app context, too.

Process	**Process Description**	
PIN Creation	Generation of the PIN, card magnetic stripe personalization, ICC personalization (if applicable), load to issuer authorization systems.	
	PIN selection	Cardholder self-selection of PIN.
PIN Transmission	Any transmission of PINs: • between issuer approved PIN handling devices, • to and from card holders.	
PIN Storage	Protection of PIN-related data by issuers, issuer approved PIN handling devices and cardholders	
PIN Processing	All processing of PINs within a PIN-handling device.	
	Online PIN verification	Online verification of the cardholder PIN.
PIN handling device management	Deployment, usage and decommissioning of equipment used to process and store PINs.	
PIN related Key Management	Management of cryptographic keys for secure PIN creation, storage, processing, transmission and verification	
Cardholder authentication	Process by which the cardholder supplies credentials to a system to access PIN management functionality	
PIN advice	Notification of the PIN to the cardholder.	
PIN change	Cardholder or issuer re-selection of PIN.	
Additional PIN management functions	Any other functionality required by issuers to manage their PINs	

Note: reprinted from VISA Issuer PIN Security Guideline, 2010

9.2.1 PIN creation

The authenticator issuer may preset the PIN (like some banks do for the chip-based credit cards) or it can be decided and initialized by the end-user during the device enrollment process. The former approach is usually considered to be safer.

Research on PIN security has indicated that PINs selected by end user are weaker because humans tend to select common easily remembered number sequences or something that is semi-public information like their birthday. PIN length is also limited. Common practice is to allow down to 4 digits for the PIN length. Short PINs require only low cognitive effort from the end-user, but they are vulnerable to brute force attacks either by human or automated attacker.

Joseph Bonneau et al from University of Cambridge found out that guessing the PIN based on victim's birth date led to success once for 11 cases. If usage of the obvious number sequences like 1234 was prohibited, they gained access by guessing once for 18 cases [88].

9.2.2 PIN storage

In addition to the PIN guessing easiness issue described above, the PIN can also be stolen by a hacker or malware installed in the running environment of an authenticator app.

Mobile platform vendors fortunately have addressed the challenge to store secret items by supplying secure key-value storages like iOS Keychain and Android Keystore. These mechanisms provide basic safety against malicious apps that aim to steal the data from other app's static storage. Also, if the secure storages supplied by the mobile platforms work in co-operation with TEE environment, higher security levels are achieved.

9.2.3 PIN processing

The risk scenarios related to the PIN processing in the mobile device can be categorized to following categories: *theft from memory* and *theft from glass*

Theft from memory

Numerous times there have been vulnerabilities in mobile platforms (specially in Android operating system) that have allowed a mobile app to read foreign app's

address space. These vulnerabilities could be used to read the PIN from memory when it is typed by the end-user. The PIN alone is not of course useful information for an attacker, but when combining with other vulnerabilities it may lead to situation where foreign app or attacker is able to reach valuable information or activate remotely events in an app.

If the application platform supplies a protected end-to-end connectivity from touchpad screen to trusted execution environment, the attacker is probably not able to reach the protected data (PIN in this case) though runtime memory.

Theft from glass

In this scenario the attacker is able to intercept the character reading process either by getting information about the touch event coordinates or even more straightforward approach by setting an own keyboard overlay on the screen capturing the user's touching actions. Again, the PIN alone is only one component of the malicious orchestration targeting to a fraudulent transaction or something valuable.

PIN on glass: PIN entry on a touchscreen keypad integrated on a PCI-approved terminal.
PIN on COTS: PIN entry on the touchscreen of an off-the-shelf consumer smartphone or tablet connected to a PCI-certified card reader.
PIN on terminal: PIN entry on a physical, push-button keypad on a card terminal.

Reprinted from website: https://www.mobiletransaction.org/what-is-pin-on-glass/

The banking industry has been actively driving security in this context by publishing related standards. Payment Card Industry (PCI) has released security requirements standard for Software-based PIN Entry on COTS (commercial of the shelf) devices [89].

10 Visions about the future

The previous chapters of this thesis have presented some aspects of current and historical authentication mechanisms. In this chapter, we present concepts and trends that can have an effect to the next generation authentication solutions.

10.1 Computational offloading

The concept of computational offloading may become popular in the future among the authentication platforms. For example, Dejan Kovachev et.al stated in their survey [90] that the mobile applications will be developed in the future so that the heavy processing will be executed in the cloud.

Computational offloading may allow the authentication platforms to have features that are beyond computational capabilities of the current mobile devices. The concept of computational offloading means an arrangement where a computationally intensive task is moved from one execution environment to another one that is more suitable for the task. Typically, it may be better to process large amounts of data in a server environment than in a mobile device having limited CPU capacity and limited battery life.

Recently there has been rising interest to mobile edge computing paradigm (MEC). The rationale behind the MEC is to improve computational capabilities of a system by performing the intensive activities in the edge nodes on the mobile network. This brings benefits regarding latency, for example (when compared to traditional cloud computing).

Artificial intelligence and machine learning are examples of features that benefit from computational offloading and MEC. Machine learning algorithms may be useful for making decisions about authentication requests based on complex input data. How to provide authentication decisions based on multiple variables is not a new computational task. For example, the 3-D Secure [91] compliant access control server of a credit card issuer must take multiple conditions and variables into account when estimating the trust of a payment transaction.

10.2 Autonomous authentication

One challenge for the authentication has traditionally been the usability. Users have thought that authentication methods are clumsy and create too much cognitive load. Users probably would prefer that the system requiring authentication would authenticate them autonomously. In the best case, authentication would be performed without any effort from the user.

Modern biometric authentication methods have come very close to achieve the minimal effort from the user. Face recognition and fingerprint authentication on mobile phones require no or little cognitive effort to use. However, they fail occasionally in certain circumstances. For example, when the fingers are wet the fingerprint reader may fail to recognize the fingerprint.

One common solution for the effortless user authentication experience could be the *autonomous authentication*. The basic idea of the autonomous authentication is to make continuous observations about the end-user by the device that he/she is using. The observations, i.e. practically collected data, are analyzed with advanced statistical methods to gather a holistic trust score. The data that is collected for the analysis can be taken from various sources.

10.2.1 Mobile carrier data

The mobile operators have access to the data that the mobile phones receive or transmit when they are connected to the network. The online behavior of the end-

user can be analyzed, for example, by recording the DNS-queries sent from the device [92]. Also, the IP addresses that the device is connected to can be tracked.

The mobile carrier data can be utilized as one data source among others for the statistical analysis of the autonomous authentication. For example, if the device suddenly starts to send packets to IP addresses that are either known to be rogue addresses or are not recorded in the usage history of the end-user, the trust score can be reduced.

10.2.2 Behavioral biometric data

The uniqueness of the human physical actions has been researched for many decades. For example, the human keystroke patterns have been found to be useful for authenticating individual users using computer systems [93] [94]. In mobile phone environment, there is typically no physical keyboard, so something else must be looked to find patterns from the actions of the end-user.

It has been found that the user actions performed on the touch screens of the mobile phones vary and can be used for the analysis and authentication [95] [96]. Timing of the finger actions, finger pressure and the area under the finger pressure are typically examined.

10.2.3 Device sensor data

The smartphones are equipped today with various sensors like accelerometer, temperature sensor and inclination sensor. Furthermore, the smartphones have typically positioning capability based on GPS. Readings of these sensors can be used for input for the statistical analysis in order to find patterns from the behavior of the end user [92]. Anomalies in the sensor readings can be used to weaken the trust score.

10.2.4 Summary

The continuous collection of the data from the various sources and the continuous statistical analysis may provide way to *autonomous authentication*. This should be researched more in the future. Autonomous authentication may be the next successful authentication method. However, the concerns of the privacy may set a challenge for this development.

11 Acknowledgements

I want to thank my wife Katja for the long awaited (25 years) inspiration to start this thesis. Also, the coaching, deep knowledge and the arguments given by Professor Valtteri Niemi have been very crucial for my work.

12 References

[1] GSM Association, "The Mobile Economy 2020," GSMA Intelligence, 30 May 2020. [Online]. Available: https://www.gsma.com/mobileeconomy/wp-content/uploads/2020/03/GSMA_MobileEconomy2020_Global.pdf. [Accessed 30 May 2020].

[2] Statista Ltd., "Number of smartphone users worldwide from 2016 to 2021 (in billions)," 2019. [Online]. Available: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/. [Accessed 30 May 2020].

[3] S. Mahmood, B. Amen and R. Nami, "Mobile Application Security Platforms Survey," *International Journal of Computer Applications,* vol. 133, pp. 40-46, 2016.

[4] European Banking Authority, "Regulatory Technical Standards on Strong Customer Authentication and common and secure communication under Article 98 of Directive 2015/2366 (PSD2)," 2017. [Online]. Available: https://eba.europa.eu/sites/default/documents/files/documents/10180/1761863/314bd4d5-ccad-47f8-bb11-84933e863944/Final%20draft%20RTS%20on%20SCA%20and%20CSC%20under%20PSD2%20%28EBA-RTS-2017-02%29.pdf. [Accessed 30 May 2020].

[5] Citrix, "What is single sign-on (SSO?)," [Online]. Available: https://www.citrix.com/glossary/what-is-single-sign-on-sso.html. [Accessed 18 May 2020].

[6] Ubisecure inc, "What is Single Sign-On (SSO)?," [Online]. Available: https://www.ubisecure.com/single-sign-on/what-is-sso/. [Accessed 18 May 2020].

[7] OneLogin inc, "How single sign-on works," [Online]. Available: https://www.onelogin.com/learn/how-single-sign-on-works. [Accessed 18 May 2020].

[8] Okta, "Federated identity vs SSO," [Online]. Available: https://www.okta.com/identity-101/federated-identity-vs-sso/. [Accessed 18 May 2020].

[9] European Banking Authority, "Opinion of the European Banking Authority on the elements of strong customer authentication under PSD2," 2019. [Online]. Available: https://eba.europa.eu/sites/default/documents/files/documents/10180/2622242/4bf4e536-69a5-44a5-a685-de42e292ef78/EBA%20Opinion%20on%20SCA%20elements%20under%20PSD2%20.pd. [Accessed 30 May 2020].

[10] VISA, "VISA issuer PIN security guideline," 2010. [Online]. Available: https://usa.visa.com/dam/VCOM/download/merchants/visa-issuer-pin-security-guideline.pdf. [Accessed 30 May 2020].

[11] IBM, "Blockchain basics: Introduction to distributed ledgers," IBM, 2020. [Online]. Available: https://developer.ibm.com/technologies/blockchain/tutorials/cl-blockchain-basics-intro-bluemix-trs/. [Accessed 30 May 2020].

[12] Wikipedia, "Blockchain," [Online]. Available: https://en.wikipedia.org/wiki/Blockchain. [Accessed 30 May 2020].

[13] Rambus Technologies, "What is white box cryptography," 2019. [Online]. Available: https://www.rambus.com/blogs/what-is-white-box-cryptography/. [Accessed 30 May 2020].

[14] A. Salomaa, Public-key cryptography, Springer Science & Business Media, 2013.

[15] A. Carlisle and S. Lloyd, Understanding public-key infrastructure: concepts, standards, and deployment considerations, Sams Publishing, 1999.

[16] Fortune Business Insights, "IAM MARKET ANALYSIS - 2026," [Online]. Available: https://www.fortunebusinessinsights.com/industry-reports/identity-and-access-management-market-100373. [Accessed 5 April 2020].

[17] FIDO Alliance, "FIDO2: WebAuthn & CTAP industry standards," [Online]. Available: https://fidoalliance.org/fido2/. [Accessed 30 May 2020].

[18] J. Andress, The Basics of Information Security, Understanding the Fundamentals of Infosec in Theory and Practice, 2014.

[19] Feitian Technologies, "BioPass tokens," [Online]. Available: https://www.ftsafe.com/Products/FIDO/Bio. [Accessed 07 April 2020].

[20] Yubiko, "Yubikey," [Online]. Available: https://www.yubico.com/products/. [Accessed 07 April 2020].

[21] OASIS Consortium, "PKCS #11 Cryptographic Token Interface Base Specification Version 2.40," 2015.

[22] Mozilla, "NPAPI documentation," [Online]. Available: https://wiki.mozilla.org/NPAPI#NPAPI_Documentation. [Accessed 12 April 2020].

[23] Google, "NPAPI plugins," [Online]. Available: https://developer.chrome.com/apps/npapi. [Accessed 12 April 2020].

[24] Mozilla, "NPAPI plugins," [Online]. Available: https://support.mozilla.org/en-US/kb/npapi-plugins. [Accessed 12 April 2020].

[25] Microsoft, "NPAPI support is currently disabled," [Online]. Available: https://answers.microsoft.com/en-us/windows/forum/all/npapi-support-is-currently-disabled/a48ec865-4282-44f8-bcf1-81ad011aa473. [Accessed 12 April 2020].

[26] Fido Alliance, "WebAuthn," [Online]. Available: https://fidoalliance.org/fido2/fido2-web-authentication-webauthn/. [Accessed 12 April 2020].

[27] ScienceDirect, "Hardware Token," 2020. [Online]. Available: https://www.sciencedirect.com/topics/computer-science/hardware-token. [Accessed 30 May 2020].

[28] Internet Engineering Task Force, "TOTP: Time-Based One-Time Password Algorithm," 2011. [Online]. Available: https://tools.ietf.org/html/rfc6238. [Accessed 30 May 2020].

[29] RSA, "RSA SecurId hardware token specifications," 2015. [Online]. Available: https://www.rsa.com/content/dam/en/data-sheet/rsa-securid-hardware-tokens.pdf. [Accessed 30 May 2020].

[30] ETSI, "Universal Subscriber Identity Module (USIM) Application Toolkit (USAT) (3GPP TS 31.111 version 13.3.0 Release 13)," 2016. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/131100_131199/131111/13.03.00_60/ts_131111v130300p.pdf.

[31] S. Drimer, S. J. Murdoch and R. Anderson, "Optimised to Fail: Card Readers for Online Banking," 2009. [Online]. Available: https://murdoch.is/papers/fc09optimised.pdf. [Accessed 30 May 2020].

[32] Cryptomathic inc, "White Paper, Two-Factor Authentication for Banking," 2012. [Online]. Available: https://www.cryptomathic.com/hubfs/docs/cryptomathic_white_paper-2fa_for_banking.pdf. [Accessed 30 May 2020].

[33] J. Brown, W. Shipman and R. Vetter, "SMS: The short message service," *Computer,* January 2008.

[34] Telecommunication Engineering Centre, Ministry of communications and information technology, Department of telecommunications, Goverment of India, "STUDY PAPER ON SS7 Security," [Online]. Available: http://tec.gov.in/pdf/Studypaper/ss7%20security.pdf. [Accessed 30 May 2020].

[35] K. Hamandi, A. Chehab, I. Elhajj and A. Kayssi, "Android SMS Malware: Vulnerability and Mitigation," in *WAINA*, 2013.

[36] National Institute of Standards and Technology, "Digital Identity Guidelines," 2020. [Online]. Available: https://pages.nist.gov/800-63-3/sp800-63b.html. [Accessed 30 May 2020].

[37] A. Rabkin, "Personal knowledge questions for fallback authentication: Security questions in the era of Facebook," in *Symposium on Usable Privacy and Security*, 2008.

[38] R. L. German and S. K. Barber, "Current Biometric Adoption and Trends," The University of Texas Austin, 2016.

[39] Yubico, "Yubico blog," 2019. [Online]. Available: https://www.yubico.com/blog/yubico-reveals-first-biometric-yubikey-at-microsoft-ignite/. [Accessed 30 May 2020].

[40] Mastercard, "Mastercard Biometric Card FAQ," 2019. [Online]. Available: https://www.mastercard.us/content/dam/mccom/en-us/documents/biometric-card-merchant-faq.pdf. [Accessed 30 May 2020].

[41] Visa, "Fingerprint authentication moves from phones to payment cards," [Online]. Available: https://usa.visa.com/visa-everywhere/security/biometric-payment-card.html. [Accessed 30 May 2020].

[42] M. Davis and T. Ellis, "Memorandum RM-4122-ARPA: The RAND Tablet: A Man-Machine Graphical Communication Device. Aug. 1964," RAND Corporation, 2005. [Online]. Available: https://www.rand.org/content/dam/rand/pubs/research_memoranda/2005/RM4122.pdf. [Accessed 30 May 2020].

[43] Apple, "About Face ID advanced technology," 26 February 2020. [Online]. Available: https://support.apple.com/en-us/HT208108. [Accessed 2020].

[44] V. Blanz and T. Vetter, "Face recognition based on fitting a 3D morphable model," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 25, no. 9, 2003.

[45] S. Chen, A. Pande and P. Mohapatra, "Sensor-Assisted Facial Recognition: An Enhanced Bio- metric Authentication System for Smartphones," in *12th annual international conference on Mobile systems, applications, and services*, 2014.

[46] J. Bonneau, C. Herley, C. van Oorschot and F. Stajano, "The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes," in *Symposium on Security and Privacy*, San Francisco, 2912.

[47] J. Bonneau, C. Herley, C. van Oorschot and F. Stajano, "Passwords and the Evolution of Imperfect Authentication," *Communications of the ACM,* vol. 58, p. 78–87, July 2015.

[48] D. Forsberg, G. Horn, W.-D. Moeller and V. Niemi, LTE Security, John Wiley & Sons Ltd., 2010.

[49] Nok Nok Labs inc, "FOUR BARRIERS TO ADOPTING STRONG AUTHENTICATION," July 2014. [Online]. Available: https://www.noknok.com/wp-content/uploads/2017/10/4barrierswhitepaper_0.pdf. [Accessed 22 02 2020].

[50] European Union, "General Data Protection Regulation," 2016. [Online]. Available: https://gdpr-info.eu/. [Accessed 30 May 2020].

[51] K. Reese, T. Smith, J. Dutson, J. Armknecht, J. Cameron and K. Seamons, "A Usability Study of Five Two-Factor Authentication Methods," in *Proceedings of the Fifteenth Symposium on Usable Privacy and Security*, Santa Clara, U.S, 2019.

[52] N. Gunson, D. Marshall, H. Morton and M. Jack, "User perceptions of security and usability of single-factor and two-factor authentication in automated telephone banking," *Computers and security,* 2011.

[53] M. Just and D. Aspinall, "On the security and usability of dual credential authentication in UK online banking," in *International Conference for Internet Technology and Secured Transactions*, 2012.

[54] K. Krol, E. Philippou, E. De Cristofaro and M. A. Sasse, ""They brought in the horrible key ring thing!" Analysing the Usability of Two-Factor Authentication in UK Online Banking," University College London, 2015.

[55] J. Colnago, S. Devlin, M. Oates, C. Swoopes, L. Bauer, L. Cranor and N. Christin, ""It's not actually that horrible": Exploring Adoption of Two-Factor Authentication at a University," Carnegie Mellon University, University of California, Berkeley, 2018.

[56] Duo Security inc, "State of the auth - experiences and perceptions of multi-factor authentication," 2017. [Online]. Available: https://duo.com/assets/ebooks/state-of-the-auth.pdf. [Accessed 30 May 2020].

[57] R. Kakerow, "Low power design methodologies for mobile communication," in *Proceedings. IEEE International Conference on Computer Design: VLSI in Computers and Processors*, 2002.

[58] H. Atlam, A. Alenezi, A. Alharthi, R. Walters and G. Wills, "Integration of cloud computing with Internet of Things: Challenges and open issues," in *2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), pp. 670–675*, 2017.

[59] S. Haber and S. W. Stornetta, "How to Time-stamp a Digital Document," *Journal of Cryptology,* vol. 3, pp. 99-111, 1991.

[60] X. Wang, X. Zha, W. Ni, R. Liu, Y. Guo, X. Niu and K. Zheng, "Survey on blockchain for Internet of Things," *Computer Communications,* vol. 136, 2019.

[61] G. Zyskind, O. Nathan and A. Pentland, "Enigma: Decentralized Computation Platform with Guaranteed Privacy," 2015. [Online]. Available: https://arxiv.org/abs/1506.03471. [Accessed 30 May 2020].

[62] R. Merkle, "A Digital Signature Based on a Conventional Encryption Function," in *Advances in Cryptology*, 1988.

[63] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," [Online]. Available: https://bitcoin.org/bitcoin.pdf. [Accessed 30 May 2020].

[64] N. T. Courtois, P. Emirdag and D. A. Nagy, "“Could bitcoin transactions be 100x faster?”," in *11th International Conference on Security and Cryptography (SECRYPT)*, 2014.

[65] A. Chauhan, P. O. Malviya, M. Verma and S. T. Mor, "Blockchain and Scalability," in *IEEE International Conference on Software Quality Reliability and Security Companion*, 2018.

[66] J. Sen, "A Robust and Efficient Trust Management Scheme for Peer-to-Peer Networ," Innovation Lab, Tata Consultancy Services Ltd., 2010.

[67] S. Hoober, "Mobile Apps: Native, Hybrid, and WebViews," 2018. [Online]. Available: https://www.uxmatters.com/mt/archives/2018/08/mobile-apps-native-hybrid-and-webviews.php. [Accessed 30 May 2020].

[68] W3C, "Generic Sensors API," 2019. [Online]. Available: https://w3c.github.io/sensors/. [Accessed 30 May 2020].

[69] S. Sasidaran, "Survey on Native and Hybrid Mobile Application Development Tools," *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) ,* vol. 6, no. 9, 2017.

[70] Internet Engineering Task Force, "The Transport Layer Security (TLS) Protocol Version 1.3," 2018. [Online]. Available: https://tools.ietf.org/html/rfc8446. [Accessed 30 May 2020].

[71] J. Petters, "What is a proxy server," Varonis, 2019. [Online]. Available: https://www.varonis.com/blog/what-is-a-proxy-server/.

[72] Open Web Application Project, "Certificate and Public Key Pinning," [Online]. Available: https://owasp.org/www-community/controls/Certificate_and_Public_Key_Pinning. [Accessed 28 March 2020].

[73] OASIS standards consortium, "SAML Wiki," 2005. [Online]. Available: https://wiki.oasis-open.org/security/FrontPage. [Accessed 12 April 2020].

[74] OpenID Foundation, "OpenID Connect specifications," [Online]. Available: https://openid.net/connect/. [Accessed 12 April 2020].

[75] V. Niemi and K. Nyberg, UMTS Security, John Wiley & Sons, 2003.

[76] J.-E. Ekberg, K. Kostiainen and N. Asokan, "Trusted execution environments on mobile devices," in *2013 ACM SIGSAC conference on Computer & communications*, 2013.

[77] Apple, "Storing Keys in the Secure Enclave," [Online]. Available: https://developer.apple.com/documentation/security/certificate_key_and_trust_services/keys/storing_keys_in_the_secure_enclave. [Accessed 30 May 2020].

[78] ARM, "Development of TEE and Secure Monitor Code," [Online]. Available: https://www.arm.com/why-arm/technologies/trustzone-for-cortex-a/tee-and-smc. [Accessed 01 June 2020].

[79] OWASP Foundation, "OWASP Mobile Top 10," [Online]. Available: https://owasp.org/www-project-mobile-top-10/. [Accessed 9 March 2020].

[80] S. Chatterjee, K. Paul, R. Roy and A. Nath, "A Comprehensive Study on Security issues in Android Mobile Phone — Scope and Challenges," *International Journal of Innovative Research in Advanced Engineering (IJIRAE),* vol. 3, no. 3, 2016.

[81] T. Alagna and E. Chen, "Larstan's the Black Book on Corporate Security," in *Larstan's the Black Book on Corporate Security*, Larstan publishing inc., 2005, p. 76.

[82] Intertrust, "Application shielding for secure applications," [Online]. Available: https://www.intertrust.com/intertrustblog/application-shielding-for-secure-applications/. [Accessed 07 March 2020].

[83] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan and K. Yang, "On the (im)possibility of obfuscating programs," *Journal of the ACM,* vol. 59, no. 2, 2016.

[84] D. E. Bryant, J. M. Atallah and R. M. Stytz, "A SURVEY OF ANTI-TAMPER TECHNOLOGIES," Center for Education and Research in Information Assurance and Security, Purdue University, West Lafayette, IN 47907-2086, 2004.

[85] Accenture, "Mobile Banking Applications security challenges for banks," 2016. [Online]. Available: https://www.accenture.com/t20180223T145013Z__w__/us-en/_acnmedia/PDF-49/Accenture-Mobile-Banking-Apps-Security-Challenges-Banks.pdf. [Accessed 30 May 2020].

[86] Inside Secure and UL, "Whitepaper - The Wild West of Mobile Security," 2018. [Online]. Available: https://www.verimatrix.com/sites/default/files/White%20paper/Whitepaper-The%20Wild%20West%20of%20Mobile%20Security.pdf. [Accessed 9 March 2020].

[87] M. Salle, N. Liampotis, D. Vaghetti, C. Kanellopoulos, M. Linden, S. Memon, D. Hübner, A. Paolini, N. van Dijk, U. Stevanovic, M. Hardt and P. Solagna, "Guidelines on stepping up the authentication component in

AAIs implementing the AARC BPA," 2018. [Online]. Available: https://aarc-project.eu/wp-content/uploads/2018/05/AARC-G029_Guidelines-on-Step-Up-Authentication.pdf. [Accessed 30 May 2020].

[88] J. Bonneau, S. Preibusch and R. Anderson, "A birthday present every eleven wallets? The security of customer-chosen banking PINs," 2012.

[89] PCI Security Standards Council, "Software-based PIN Entry on COT," 2018. [Online]. Available: https://www.pcisecuritystandards.org/documents/SPoC_Security__Requirements_v1.0.pdf. [Accessed 30 May 2020].

[90] D. Kovachev and R. Klamma, "Beyond the client-server architectures: A survey of mobile cloud techniques," in *1st IEEE International Conference on Communications in China Workshops (ICCC)*, 2012.

[91] EMV, "3-D Secure Specifications," 2018. [Online]. Available: https://www.emvco.com/emv-technologies/3d-secure/. [Accessed 30 May 2020].

[92] M. Jakobsson, E. Shi, P. Golle and R. Chow, "Implicit authentication for mobile devices," in *Proceedings of the 4th USENIX conference on Hot topics in security (HotSec'09)*, 2009.

[93] S. Bleha, C. Slivinsky and B. Hussien, "Computer-access security systems using keystroke dynamics," *IEEE Transactions on PatternAnalysis and Machine Intelligence,* vol. 12, 1990.

[94] Y. Zhao, "Learning User Keystroke Patterns for Authentication," 2005.

[95] M. H. Wajeeh and S. M. Hameed, "USER AUTHENTICATION BASED ON TOUCH DYNAMICS OF PATTERN UNLOCK," *International Journal of Computer Science and Mobile Computing,* vol. 4, no. 5, 2015.

[96] P. S. Teh, N. Zhang, S.-Y. Tan, Q. Shi, K. How and R. Nawaz, "Strengthen user authentication on mobile devices by using user's touch dynamics pattern," *Journal of Ambient Intelligence and Humanized Computing,* 12 2019.

[97] S. Boonkrong and S. Vongsingthong, "A Survey on Smartphone Authentication," *Walailak Journal of Science and Technology,* vol. 12, pp. 1-19, 2015.

[98] Yubiko, "Yubiko Website," 2020. [Online]. Available: https://www.yubico.com. [Accessed 30 May 2020].