Masaryk University
Faculty of Informatics

Multi-Factor Authentication in Large Scale

Master's Thesis

Bc. Pavel Břoušek

Brno, Spring 2019

Masaryk University
Faculty of Informatics

Multi-Factor Authentication in Large Scale

Master's Thesis

Bc. Pavel Břoušek

Brno, Spring 2019

This is where a copy of the official signed thesis assignment and a copy of the Statement of an Author is located in the printed version of the document.

Declaration

Hereby I declare that this thesis is my original authorial work, which I have worked out on my own. All sources, references, and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Bc. Pavel Břoušek

Advisor: RNDr. Michal Procházka, Ph.D.

Acknowledgements

I wish to express sincere thanks to my supervisor, RNDr. Michal Procházka, Ph.D., for his valuable advice and guidance and for introducing me to the work on Unified login.

I am also grateful to Evan McElravy for proofreading and my girlfriend, Anna, for her endless support.

Abstract

Physical world activities have been migrating to the virtual world – this includes communication, banking, commerce and more. One of the requirements to make this migration happen is establishing a digital identity of each user. Identification of users would not be possible without proper authentication. Authentication with passwords, although common, faces a number of security issues, so new authentication methods have been developed to be combined into multi-factor authentication.

This work provides a comparison of various multi-factor authentication methods, which can be used for authentication of thousands of users over the internet. Some of the methods, which were identified as suitable, have been implemented and tested in the Masaryk University Unified login identity provider.

Keywords

SimpleSAMLphp, SAML, multi-factor authentication, password replacement, single sign-on, federated identity management, user authentication

Contents

List of Tables

List of Figures

Introduction

Physical world activities have been migrating to the virtual world – these include communication, banking, commerce and more. One of the requirements to make this migration happen is establishing a digital identity of each user. Digital identification of users has to deal with the dynamic nature of the Internet as well as numerous security threats. Identification of users would not be possible without proper authentication, which should be both resilient to attacks and user-friendly.

Passwords have been the most common user authentication method since they were introduced. They are simple to use and implement, but there are many security issues related to their usage. Many attacks have been developed to overcome password-based authentication. Because of that, new authentication methods have been developed in order to replace passwords. The new methods usually perform well from the security perspective, but some of their properties, such as cost or user-friendliness, do not make them suitable enough to replace passwords completely. Therefore, online services have been implementing multi-factor authentication, which usually combines password-based authentication with another mean of authentication, such as using a hardware security token or TOTP.

Another way to reduce password usage is not to perform user authentication at all, but to delegate it and rely on supplied identities from a trusted identity provider. This not only reduces the complexity of the service but also allows trustworthy user properties to be obtained from the identity provider, for example student status in the case of a university login. If a service wants to grant access only to students, it does not have to deal with paperwork, since the university identity provider supplies this information with every login.

Masaryk University Unified login serves as an identity provider for dozens of services, some of which allow users to perform sensitive operations such as financial transactions or private information access. Therefore it has to offer a reasonable level of security assurance for all users and a high level of security for more privileged users. Since multi-factor authentication is not yet implemented, it can be used to improve the security of users.

In this work, I evaluate various methods of authentication which can be used for multi-factor authentication on the web of an organization with thousands of users, such as Masaryk University. For each method, I identify its strengths and weaknesses and its suitability to be used for primary or secondary authentication. Furthermore, I select some methods which are suitable for implementation in the Masaryk University Unified login, and try them in the testing environment of Unified login.

1 Basic terms and concepts

Basic terms concerning digital identity and authentication are described in this chapter.

1.1 Identity

Identity in the physical world may be defined as an "exclusive perception of life, which is bound to a body". [1] It allows people to be identifiable in society or in a social group over time. The more practical view is that identity is a set of a person's attributes which uniquely distinguish this person from other people in a group.

"Digital identity refers to a representation of the identity of a person in digital environments." [2] In the digital world, a person's identity usually consists of a set of attributes, sometimes even only one unique identifier. Digital identity is therefore this digital representation of the attributes associated with a person.

1.1.1 Federated identity

"Federated identity allows users to link identity information between accounts without centrally storing personal information." [3] This practically means that a user can be authenticated by a single identity provider (IdP) and then access multiple service providers (SPs) without having to manage separate credentials for each of them. Users can see and control which services are linked to their identity and which attributes are shared with each of them. The result is a lower number of authentications and greater control over personal data.

1.2 Identity management

"Identity management means managing various partial identities (usually denoted by pseudonyms) of an individual person." [1] Identity management involves the management of personal attributes, including their storage, processing, disclosure, and disposal. Identity management technologies and solutions are mainly deployed within en-

terprises since their operation does not pay off unless the user base is large.

Identity management systems have to determine which subset of attributes (partial identity) is going to be provided to a certain service. "Identity assurance is concerned with the proper management of risks associated with identity management." [4]

1.3 User authentication

Authentication of users is crucial for identity management so that the identity can be trusted. "Authentication of an individual, in the context of identity management, is the process of establishing enough confidence in the fact that an alleged (partial) identity truly describes that individual." [2] A user has to authenticate by providing a piece of evidence to support the claimed identity. There are three categories and many types of credentials, with the most basic and common being passwords. Identification and authorization are also a precondition for authorization and access control.

1.3.1 Step-up authentication

"The process of moving up from one authentication level to the next level is termed step-up authentication." [5] After a user has logged in using lower level authentication, upon accessing more sensitive resources, the user is required to authenticate again using a higher level authentication. This scenario is implemented in some systems to balance the security of sensitive operations and user-friendliness.

1.3.2 Multi-factor authentication

Multi-factor authentication (MFA), in contrast with step-up authentication, requires the high-level authentication on each login. MFA is described as "a method of authentication which requires the user to have a combination of at least two out of the following three types of credentials:" [6]

- something you know (password, PIN)
- something you have (hardware token, one-time password)

- something you are (biometrics like a fingerprint)

"Single-factor authentication (SFA) uses one of these components as a means of verifying customer identity." [7] The most common example is a password or a PIN. "Two-factor authentication (2FA) potentially provides enhanced security by combining two different authentication components." Since for example passwords are easier to compromise remotely and tokens are easy to steal, their combination provides protection against both types of threats because an attacker would have to compromise both at the same time.

Two-factor authentication is also called two-step verification (2SV) or generally MFA.

The introduction of MFA brings the need to use single-purpose passwords for applications which cannot offer MFA. These passwords are usually called app passwords. "An app password is a long, randomly generated password that you provide only once instead of your regular password when signing in to an app or device that does not support two-step verification." [8]

1.4 Factors of authentication

1.4.1 Knowledge

Traditional passwords, as well as PIN codes, fall under the category of "something you know". Passwords and PIN codes are the most widespread method of user authentication. They offer several benefits including the ease of use, interoperability, and easy and cheap deployment. [9] From the security point of view, passwords are less than ideal. Passwords with sufficient entropy to sustain brute-force or dictionary offline attacks are hard to remember. Users often choose an identical password for many services, which lowers the password's security to the security of the least trustworthy of them. Attempts to increase the security of passwords by introducing password expiration or imposing strength requirements do not seem to be successful [10]. Nevertheless, no authentication method which would outperform passwords in all aspects has been introduced so far.

1.4.2 Ownership

Security hardware tokens such as RSA SecurID[1] or YubiKey[2] fall under the category of "something you have", together with more traditional authentication methods such as one-time passwords (OTPs) on a piece of paper. As of 2019, the number of authentication methods featuring mobile phones is growing – from the HMAC-based one-time password (HOTP) and time-based one-time password (TOTP), authenticator applications showing push notifications, up to FIDO[3] universal 2nd factor (U2F). Google even embedded a software security token in the Android operating system [11].

Security tokens can store secure passwords or use public key cryptography, so they are usually more secure than traditional passwords. Proving ownership of a token is also more tightly bound to a specific user. Although the token may be lost or stolen, it is considered easier to protect the hardware token than a password. The main downside of secure hardware tokens is their price; tokens cost tens of dollars each (as of 2019).

1.4.3 Inherence – biometrics

"Biometrics are based on the principle of measurable physiological or behavioral characteristics such as a fingerprint or a voice sample." [12] Fingerprints are the most common biometrics used for authentication. The main advantage of biometrics is that they truly authenticate a human user, not a piece of information or an item, and users have them always at the ready. But they also face a number of difficulties. Measurements of the same biometric are never the same and also the result of the comparison is not binary, so there needs to be an acceptance (or rejection) threshold, which implies the possibility of false positives and false negatives. Biometrics are not accessible to all people because of permanently or temporary missing or damaged body parts, but also weather might cause false rejection. Biometric data usually has insufficient entropy for cryptographic purposes and is also public;

1. https://www.rsa.com/en-us/products/rsa-securid-suite/rsa-securid-access/securid-software-tokens
2. https://www.yubico.com/products/yubikey-hardware/
3. https://fidoalliance.org/

people leave fingerprints on almost everything they touch, fakes are quite easy and cheap to create, and liveliness detection can usually be bypassed. Because of the limited number of body parts (for example fingers), there is a limited number of unique ”passwords” a person can use, and they cannot be revoked. Most of these difficulties can be solved by using a biometric locally to protect a strong cryptographic key.

2 Federated identity management

As we saw in section 1.1, digital identity consists of attributes associated with a user, stored in an identity management system in digital form. SPs rely on identity systems for user authentication. Organizations offer their identity management not only for internal use and in their own services but also for external SPs. This is referred to as federated identity management. [13]

”A federated identity management system consists of software components and protocols that handle the identity of individuals throughout their identity lifecycle.” [14] In a federated identity management system, we distinguish three types of entities – users, IdPs and SPs. ”The IdPs manage and provide user identities and potentially issue user credentials, and the SPs (also known as relying parties) are entities that provide services to users based on their identity (attributes).” [14] IdPs usually provide single sign-on (SSO) endpoints which are used by multiple SPs.

Two common authentication/authorization frameworks, supported by Unified login at the time of writing, are SAML and OpenId Connect.

2.1 Single sign-on

”Single sign-on is a concept to delegate the authentication of an End-User on an SP to a third party – the so-called IdP.” [15] As described in subsection 1.1.1, this lowers the number of credentials that have to be kept as well as the number of re-used passwords and passwords in general. It also gives more control over personal information to users and provides a better user experience.

2.2 Security Assertion Markup Language

”Security assertion markup language (SAML), which is based on XML, provides a framework for authentication and authorization in Web services.” [16] The current version at the time of writing (May 2019) is 2.0. SAML defines the authentication flow between a user, an IdP and an SP. Trust between the IdP and the SP is established by exchanging

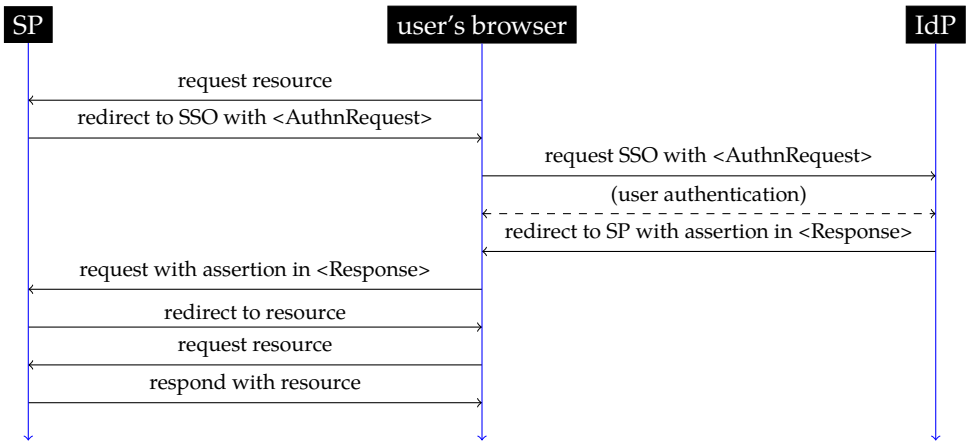

Figure 2.1: SAML 2.0 authentication flow

metadata and certificates with public keys. SAML messages are sent in the XML format, which is base64-encoded and signed. The messages are included in HTTP GET or POST fields. The user is redirected from the SP with an authentication request to the IdP, which shows a login page and authenticates the user. Upon successful authentication, an identity assertion with the appropriate set of personal attributes is encoded and included in an authentication response. The user is redirected back to the SP with this assertion, which is validated and acted upon. The authentication flow of SAML 2.0 is shown in Figure 2.1.

2.3 OAuth 2.0 and OpenId Connect

An alternative to SAML is OpenID Connect (OIDC), which is built on top of OAuth 2.0. "The OAuth 2.0 authorization framework enables a third-party application to obtain limited access to an HTTP service, either on behalf of a resource owner by orchestrating an approval interaction between the resource owner and the HTTP service or by allowing the third-party application to obtain access on its own behalf." [17] SP is referred to as the relying party (RP) and IdP is called an authorization server (AS).

"OIDC is the OAuth 2.0-based replacement for OpenID 2.0 (OpenID)." [15] According to "OpenID Connect FAQ and Q&As", while "SAML is an XML-based federation technology used in some enterprise

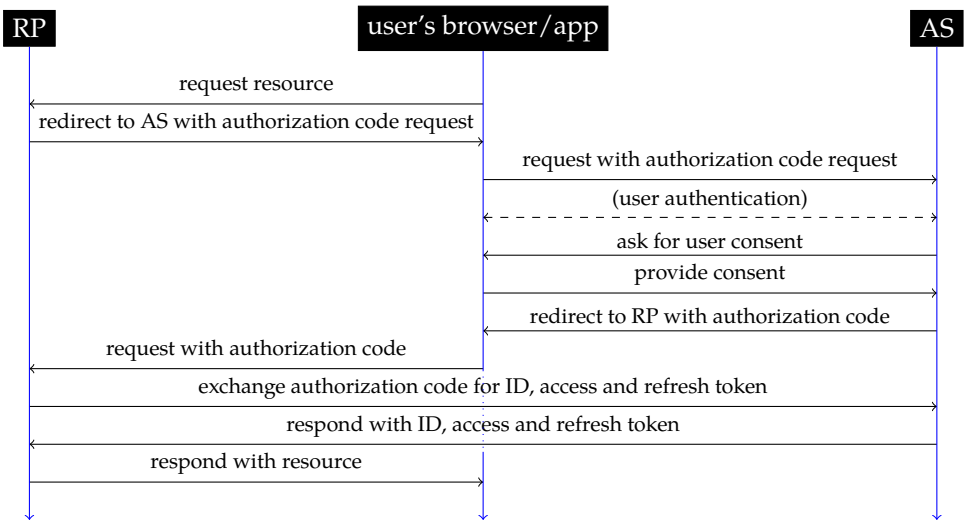

Figure 2.2: OIDC authorization code flow

and academic use cases", OpenId Connect "can satisfy these same use cases but with a simpler, JSON/REST based protocol" [18]. OIDC is newer than the strictly web-based SAML, so it was designed with native and mobile applications support already in mind. Both SAML and OIDC are currently being used and libraries for their usage being developed and maintained. OIDC supports three authentication flows – authorization code flow, implicit flow, and hybrid flow [19]. The authorization code flow, which is most similar to SAML, is shown in Figure 2.2.

2.4 REFEDS assurance framework

SAML and OIDC do not have built-in support for expressing identifier uniqueness, identity assurance, and attribute assurance. These can be implemented by specifying optional attributes or values, for example, by complying with the Research and Education FEDerations (REFEDS) group's assurance framework.

The REFEDS group is "a community of practitioners actively engaged in access and identity work. The aim of REFEDS is to exchange Identity Federation processes, practices, and policies and to discuss ways to facilitate inter-federation work." [20] This group has published

an optional extension to SAML and OIDC called REFEDS Assurance Framework [21]. It defines several values which an IdP may assign to users to express certain properties of the claimed identity. In the case of SAML, these values are placed in the `eduPersonAssurance` attribute. Values specified in version one of the framework valid for the Unified login (excluding guest users) are described in Table 2.1.

Cappuccino and Espresso are assurance profiles, which should also be presented for easier treatment on the SP. The assurance profiles are described in Table 2.2.

Table 2.1: REFEDS assurance framework values

Value	Description
`https://refeds.org/assurance`	conformance to definitions
`$P/ID/unique`	user identifier is never re-assigned and represents a single person who can be contacted
`$P/ID/eppn-unique-no-reassign`	eduPersonPrincipalName is never re-assigned and represents a single person who can be contacted
`$P/IAP/local-enterprise`	the user is / would be allowed to access internal systems
`$P/ATP/ePA-1m`	affiliation attributes are updated at least monthly
`$P/ATP/ePA-1d`	affiliation attributes are updated at least daily
`$P/IAP/low`	for example self-asserted identity together with a verified e-mail address
`$P/IAP/medium`	for example remote ID check and live video
`$P/IAP/high`	user presented genuine identity document and extra steps were taken to protect the account
`$P/profile/cappuccino`	assurance profile, see Table 2.2
`$P/profile/espresso`	assurance profile, see Table 2.2
$P = `https://refeds.org/assurance`	

Table 2.2: REFEDS assurance profiles

Value	Cappuccino	Espresso
`https://refeds.org/assurance`	✖	✖
`$P/ID/unique`	✖	✖
`$P/ID/eppn-unique-no-reassign`		
`$P/ID/eppn-unique-reassign-1y`		
`$P/IAP/low`	✖	✖
`$P/IAP/medium`	✖	✖
`$P/IAP/high`		✖
`$P/IAP/local-enterprise`	✖	✖
`$P/ATP/ePA-1m`	(✖)	(✖)
`$P/ATP/ePA-1d`		

$P = `https://refeds.org/assurance`,
✖ = required, (✖) = conditionally optional

3 Multi-factor authentication solutions evaluation

This chapter contains an evaluation of several web-compatible authentication methods, which are candidates for SFA or MFA in organizations with tens of thousands of users. In general, these need to be secure, inexpensive for most users, and usable by all. I assume that the setup is done correctly, so I do not evaluate their susceptibility to erroneous deployment. My evaluation criteria and assessment method are followed by descriptions of each authentication method and their evaluations. The evaluation is summarized in section 3.5.

3.1 Evaluation criteria

I chose several criteria to evaluate authentication methods based on their properties, partly inspired by the paper *The quest to replace passwords* [9]. I take into account the difficulty of user compromise, which is divided into difficulties of cautious or irresponsible user compromise, remote or local. The authentication secret or token should be difficult to give away and should not link identities between services. In enterprises, it is considered beneficial if the method is standardized or at least open-source and can be used without third-party dependencies. From the user point of view, it should be easy to use, and it should be reasonably easy and cost-effective to deploy.

3.1.1 Difficulty of user compromise

In the case of Unified login, not all users are allowed to perform sensitive operations like financial transactions. These users might not consider the protection of this account important [22]. Good authentication methods should therefore be as resilient as possible in the face of irresponsible users (those who for example choose weak passwords). On the other hand, it has also to offer a sufficient level of security when required (for example for the privileged users).

I take into account resistance to various kinds of attacks:

- traffic capture (sniffing)

- man-in-the-middle attack
- covert observation
- recording attack
- phishing

Malware is not taken into account since the presence of malware usually breaks the authentication scheme. I assign levels low, medium, and high according to the criteria listed in Table 3.1.

Table 3.1: User compromise evaluation criteria

Evaluation	Description
● high	active real-time attack required, brute-force not possible, requires physical theft
⊙ medium	active non-real-time attack or brute-force possible
○ low	passive attack or fast brute-force possible

Apart from security against attackers, it is also beneficial if the authentication method makes it difficult for users to give away their credentials (such as telling somebody their password). The evaluation of this is described in Table 3.2.

Table 3.2: Self-sabotage resistance evaluation criteria

Evaluation	Description
● high	Users have to lend their device
⊙ medium	Users have to lend an item or create a physical copy
○ low	Users simply give away the secret information

3.1.2 Privacy

One of the downsides of passwords it that when a password is used for multiple services, a leak from one of them compromises the others as well. In case of passwords, it is up to users not to share the same password for multiple services – which is probably only possible

using a password manager. Some authentication methods solve this by default, often preserving unlinkability of the authenticator device or application between services. Levels assigned to the privacy criterion are listed in Table 3.3.

Table 3.3: Privacy evaluation criteria

Evaluation	Description
● high	Info stored in the database is by default unique per service
⊙ medium	Info stored in the database may be unique per service with extra effort
○ low	Info stored in the database cannot be unique per service

3.1.3 Vendor-independence

The authentication method should not require paid or proprietary software. It should not depend on an external server or service, so that the availability of the IdP is not dependent on a third party. Standard-compliant open-source software hosted in-house is preferred. The assigned levels are listed in Table 3.4.

Table 3.4: Vendor-independence evaluation criteria

Evaluation	Description
● high	open source implementation exists or it is based on an open standard
⊙ medium	single vendor implementation or limited to specific vendors
○ low	closed source, limited to a single vendor

3.1.4 Impact of server compromise

It is considered beneficial if the compromise of a server (for example a database leak) does not result in the attacker being able to impersonate any user, even to the affected service. This security requirement is not

as strong as the previous ones, because I expect the server and database to be fairly well protected in the considered environment. I do not take into account the authentication method's resistance to improper deployment, because I assume that in a large organization the setup will be done properly.

The assigned levels are listed in Table 3.5.

Table 3.5: Server compromise evaluation criteria

Evaluation	Description
● high	database leak does not result in any impersonation
⊙ medium	database leak results in privacy violation or impersonation for irresponsible users
○ low	database leak results in unlimited impersonation

3.1.5 Usability

It is important that authentication methods are easy to use because of the large user base. The following properties make an authentication method more difficult to use:

- the need to remember something
- the need to carry something
- the need for physical actions
- the need to use a specific body part
- the need to install specific software
- lengthy learning curve
- lengthy logging in
- frequent false rejections

It is required that the method be accessible to handicapped users – those without a screen, without a standard keyboard, etc. – so that

most users are able to use the authentication method. Usability levels are described in Table 3.6.

Table 3.6: Usability evaluation criteria

Evaluation	Description
● high	No action or single button press is required
⊙ medium	Either typing, switching apps, or body part engagement is required
○ low	A combination of actions is required

3.1.6 Ease of deployment

Because of the large number of users, the average cost per user (including both money and time) should be negligible. The authentication method should require a reasonably simple setup which works for all users. Maintenance (excluding password resets) should require limited human resources. The assigned levels are listed in Table 3.7.

Table 3.7: Ease of deployment evaluation criteria

Evaluation	Description
● high	Existing component installation and simple configuration is required
⊙ medium	Installation of several components or advanced configuration or customization is required
○ low	Custom implementation and specific hardware is required

3.2 Authentication by knowledge

Passwords are by far the most common authentication by ”something you know” on the web. A PIN alone does not offer a sufficient level of security (entropy). So-called ”security questions” are considered bad practice, although they are still very common, because the answers are usually public (like mother’s maiden name), and they can usually be

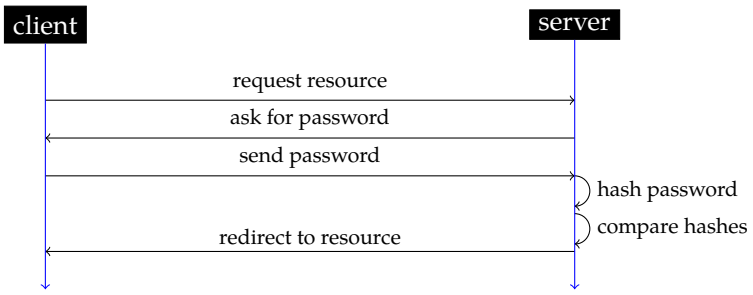

Figure 3.1: Password authentication

used to reset the password. There are also other schemes, for example using pictures and requiring users to click on certain picture(s) from a set or to click on certain points in the picture. They are usually considered similarly secure as a PIN, so they are used only locally or together with another security measure.

3.2.1 Passwords

Passwords do not perform well in the security categories. Even if a password has sufficient entropy, is not re-used, and is protected by hashing and salting, ideally with a memory-hard function, it is still sent over the network, so it may be eavesdropped, stolen from a password manager, or obtained by brute-force from a leaked database. If we consider malware, there are further possibilities to obtain passwords, for example using key loggers. And since the offered security drops with an irresponsible user, I assign the level medium to cautious user compromise and the level low to the others. Passwords can be easily given away; this implies the level low in self-sabotage resistance. Re-used passwords represent a privacy violation, granting the level medium in privacy. The strengths of passwords are vendor-independence and the ease of deployment, in which categories they get level high. On server compromise, passwords can always be obtained, although with nontrivial computing power, so they get the level medium. Usability of passwords is quite good, as text input is usually not a limitation, but in the ideal case, users have to remember dozens of unique, random passwords, so I assign the level medium in usability.

3.3 Authentication by ownership

Security hardware tokens are commonly used for 2FA, although they can be used as a primary authentication method as well. The reasons not to do so are that even a weak password still raises the bar for an attacker, and also if the token is not protected by a PIN (or equivalent), it would be easy to impersonate physically proximate users just by stealing and using the token.

3.3.1 Shared secret based

Password managers

Password managers basically turn authentication by "something you know" into "something you have". Passwords are stored in a local or remote database, usually protected by a master password. This (mostly) increases security in comparison to just using passwords – a password manager can be usually trusted more than several services, to which security might not be of great importance. The most significant benefit is the possibility of using a strong and unique password for each service, with no changes required on the server side. But password managers are not an improvement in all respects – in the case of a vulnerability in the manager itself (or the underlying operating system), all passwords might be leaked or lost at once. In case of synchronization between devices, the passwords have to be somehow shared with a remote server. Some services try to force users into using password managers by setting high password strength requirements, such as 16 or more characters, but this does not prohibit users from copying the password on a piece of paper, and better alternatives exist.

HOTP and TOTP

An improvement over static passwords are HOTP and TOTP. Client and server have a shared secret key, and they also share a counter (HOTP) or clock (TOTP). The shared secret is transmitted over the network only during registration. On login, only the result of a one-way function of the shared secret and the counter or time is sent. HOTP

is computed according to the following formula:

$$\text{HOTP}(K, C) = \text{Truncate}(\text{HMAC-SHA-1}(K, C))$$

The truncate operation results in a number in decimal format of 6 to 10 characters. "TOTP is the time-based variant of this algorithm, where a value T, derived from a time reference and a time step, replaces the counter C in the HOTP computation. TOTP implementations may use HMAC-SHA-256 or HMAC-SHA-512 functions, based on SHA-256 or SHA-512 hash functions, instead of the HMAC-SHA-1 function that has been specified for the HOTP computation." [23] So TOTP has the following formula:

$$\text{TOTP}(K, T) = \text{Truncate}(\text{HMAC-SHA-256}(K, T))$$

TOTP is usually preferred over HOTP, because the codes are guaranteed to be short-lived, it uses a stronger hash function, does not allow denial of service, and it is easier to handle skipped codes. In HOTP, if a user generates several codes without using them, the server has to generate all of them and find the matching one in order to increment the counter accordingly. In TOTP this is not needed, the clocks just have to be in sync, which is usually the case with both servers and smartphones being automatically synchronized with time servers. To allow a little difference between client and server clock or to allow more time for typing, the server may accept the preceding and subsequent TOTP but does not have to search for the correct counter. This adjustment has to be implemented with caution because it increases usability at the expense of lowering security by extending the attacker's window of opportunity.

Both HOTP and TOTP only transmit a product of a one-way function, so the shared secret and counter are believed to be impossible to obtain remotely. Because of the increasing counter or time included in the computation, old codes cannot be reused. Therefore, I assign the level high to remote compromise resistance as well as both cautious and irresponsible user compromise resistance. Local compromise is easier: an attacker with temporary access may alter the system time or increase the counter to obtain a code valid in the future, which can be later used to log in and in case of HOTP to cause a denial of service. Because the attack requires getting the user's unlocked device, this results in the level medium in local compromise resistance. Giving away

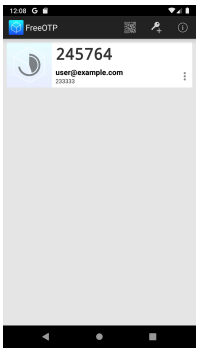

Figure 3.2: TOTP code in FreeOTP Authenticator

the shared secret is usually only possible during registration or by lending the user's device, which yields the level high in self-sabotage resistance. Privacy is not an issue since the shared secret is different for each registration. Both protocols are RFC standards, so highly vendor-independent. The weakness is the low resistance to server compromise because the shared secret has to be stored unhashed. Usability is similar to passwords, either typing or switching apps is required, so I assign the level medium. Both HOTP and TOTP are easy to deploy because of their simplicity and the number of available implementations.

Yubico OTP

Yubico OTPs[1] are 44-character long strings – the first 12 characters being the Public ID of the YubiKey device and the remaining 32 characters making up a unique code. Yubico OTP is not dependent on time, only on counters (same as with HOTP), and if a counter value lower than the one already seen is encountered, the OTP is rejected.

Yubico OTP is supported by YubiKey hardware tokens with secure storage, so the secret is almost impossible to extract. It offers a high level of security for cautious users and against remote attacks. I grant only medium in irresponsible user compromise, since a generated code is valid for any service, unlike HOTP or TOTP. Local compromise is probably even easier, because only the token has to be stolen or used. Users might be more likely to lend a token instead of their mo-

1. `https://developers.yubico.com/OTP/`

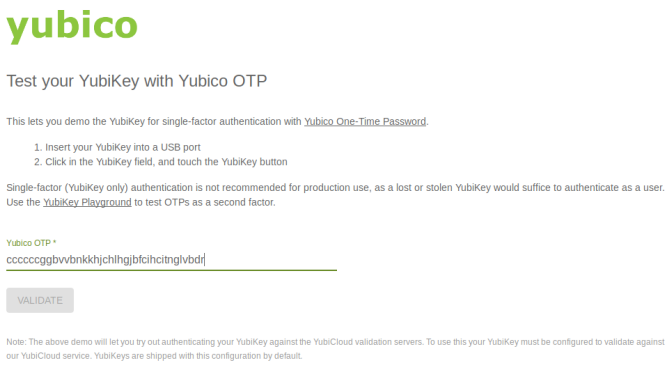

Figure 3.3: Yubico OTP demo web page

bile phone, so I assign medium in self-sabotage resistance. Privacy is only mediocre because the public ID of the YubiKey is the same for all services, so unlinkability is only possible using multiple tokens. This solution comes from a single vendor, but libraries are available, and the validation server may be hosted in-house, so I assign the medium level. Server compromise resistance is high because the database only contains public IDs. In case of an in-house server, the impact is higher, because symmetric keys are used. Usability is also high because logging in requires only touching a button. Deployment is easy with the libraries and plugins being available.

3.3.2 Challenge-response based

The shared-secret based methods prevent reusing OTPs by rejecting old codes based on the creation time or increasing counter value, but they do not prevent the pre-generation of future codes. This risk can be mitigated by using a challenge-response mechanism, where the server generates a fresh random challenge and asks the client to include it in the login request, ideally after performing a client-side computation, for example, hashing, encryption, or signing. The basic scheme is shown in Figure 3.4. This scheme makes it impossible to pre-generate valid codes since the challenge is set to expire. On the other hand, it

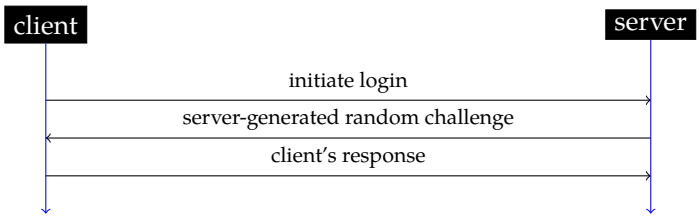

Figure 3.4: Basic scheme of challenge-response authentication

increases the load on the server, because the challenges have to be stored.

SMS OTP

One of the most basic and common methods of this kind are SMS OTPs – typically a 4- to 8-digit code is sent via text message to the user's phone number and has to be copied or typed in. They are commonly used for 2FA and step-up authentication by large organizations including banks, although they face a number of security issues, and their use has been discouraged by NIST since 2017 [24]. There also used to be a service which allowed the use of SMS OTPs for user authentication called (Twitter) Digits[2], which was discontinued in September 2017 in favor of Firebase phone auth[3]. SMS authentication requires setting up text message delivery or using a third-party service, which brings additional costs. There are several methods to bypass it, including phone number hijack, eavesdropping, misusing SMS read permission, or reading the code from a notification on a locked phone. This method also has privacy issues, since the use of same phone number not only links the user across services but also reveals private contact information. On the other hand, SMS OTPs are still valid for phone number verification, although extra steps have to be taken to mitigate virtual numbers and online services for receiving SMS messages such as Receive SMS online[4].

2. https://web.archive.org/web/20170825215701/https://get.digits.com/ and https://web.archive.org/web/20180424012843/http://get.digits.com:80/blog/digits-is-now-shut-down
3. https://firebase.google.com/docs/auth/android/phone-auth
4. https://www.receivesmsonline.net/

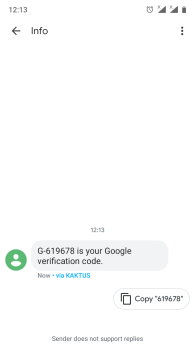

Figure 3.5: Example of a SMS OTP

Even for a cautious user, the SMS OTP may be compromised, so I assign medium for cautious users and low for the others. Remote compromise is limited but still possible, so I assign the level medium. Local compromise is even easier than with TOTP; the text message might simply be read from a locked phone's display in some cases. Self-sabotage in the form of forwarding the OTP is only possible in real time and only once, so I assign the medium level. Privacy is low because the phone number is shared with all of the services and we can assume that most users cannot use a unique one for each service. This privacy violation also increases the impact of server compromise, which itself does not allow impersonation, but still gets only medium for violating privacy. Usability is similar to TOTP so I assign medium. Deployment is complex and requires setting up text message delivery or using a third-party service, so I assign medium to vendor-independence and low to the ease of deployment.

Login notifications

An improvement over text messages are push notifications sent to a company's (authenticator) app. Since it is possible to use device storage and other capabilities, secret or public key cryptography may be involved together with an entropy-wise stronger challenge. It is also more difficult for other apps or malware to access the codes. On the other hand, compatibility is worse than with SMS because users have to use smartphones and an app has to be created for each supported operating system. The notification delivery itself requires either a

vendor-dependent setup or the involvement of a third party service, which is not desired for a security-related use case.

For login notifications, I assign high for cautious users and remote attacks and medium for irresponsible users and local attacks. Notifications are more difficult to hijack than SMS, and the notification cannot be tapped from a locked screen. Notifications cannot be forwarded in any way, so self-sabotage resistance is high. Linkability is not an issue because each service has to provide its own application, so I assign the level high to privacy, although different privacy concerns might arise from installing these applications on a personal device. This method is highly vendor-dependent, and deployment is difficult, so this yields low levels. On the server side, there is only an identifier for pushing a notification into the device, so the compromise impact is low. Usability is excellent, because the notification has to be just tapped or confirmed.

U2F

As hardware security tokens started to become more common and the need for MFA began to rise, several standards regarding 2FA were introduced under the FIDO2 framework. The most important of these is probably U2F, which defines a challenge-response based protocol for registration and for authentication, which is shown in Figure 3.6. Security devices which are U2F-compliant (such as a YubiKey) can be used with any service which supports U2F. Assuming a secure channel (HTTPS/TLS with verified origin) and no malware on the client side, the protocol is designed to prevent man-in-the-middle attack (TLS channel ID), phishing (origin), tracking multiple registrations of the same device (key handle and app id), and device cloning (counter). The standard itself limits the effectiveness of these protections to malware-free clients and remote attacks – theft of the token still allows impersonation, if used for SFA or if the password is also compromised.

Remote attack resistance is high by design, even for incautious users. Local compromise has been put out of scope by the standard, which is designed to prevent more common online attacks, and is very easy. Self-sabotage is possible by lending the key but is not permanent, so I assign the medium level. Privacy is ensured by key handles and

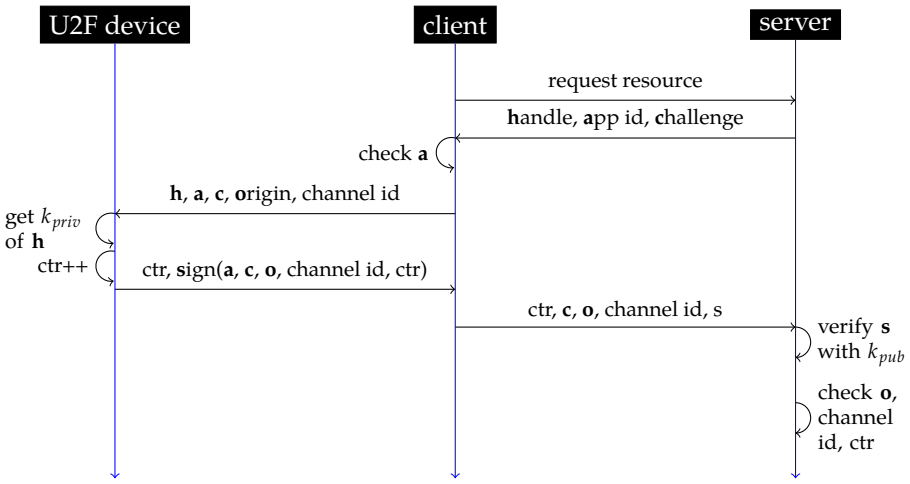

Figure 3.6: Universal second factor authentication

app ids. U2F is described in a vendor-independent standard. Because the server only stores public keys (or private keys encrypted by a symmetric key present only on the device), a database leak does not allow any impersonation, even to the breached service. Usage is usually as simple as pressing a button, so usability is high. Deployment is also easy, there are libraries for both clients and servers available, as well as two browser APIs which can be used, FIDO U2F API[5] and Web Authentication (webauthn)[6].

TiQR

TiQR is an authentication system based on challenge-response. The challenge in the form of a QR code is scanned with a mobile application. The user also has to confirm every login with a PIN, so this authentication is a combination of ”something you have” and ”something you know”. The authentication flow is shown in Figure 3.7. It is quite user-friendly since users only have to remember the PIN code and carry their mobile phone. The downsides are that an internet connection is required on the phone, and also when logging in from a mobile phone, users have to switch apps.

5. https://fidoalliance.org/specs/fido-u2f-v1.0-nfc-bt-amendment-20150514/fido-u2f-javascript-api.html
6. https://www.w3.org/TR/webauthn/

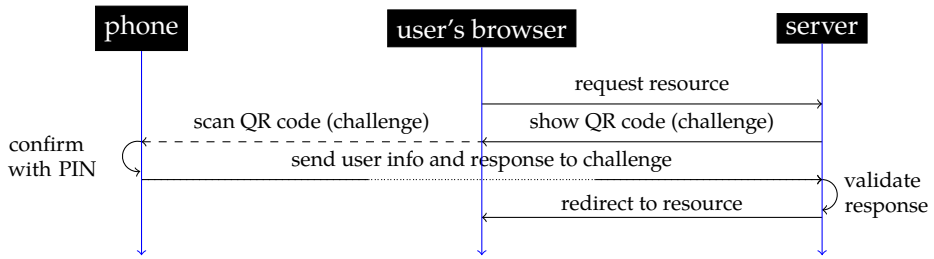

Figure 3.7: tiqr authentication

Because of the combination of a challenge-response based authenticator application and a PIN, this authentication scheme is the most robust one from the security standpoint in this comparison. It should be noted that the security of other methods is also improved when used as a second factor or if they were combined with a PIN. Privacy is medium because the same ID is used with all services. I assign vendor-independent as low, because there is only one implementation from the creators, and because for better user experience (the TiQR app has to be opened to scan the QR code), push notifications have to be set up, and also the TiQR app is only available for Android and iPhone as of May 2019. The server does not store a shared key, so server compromise impact is low. Usability is the lowest in this comparison, because of all the steps. The setup is a little bit more complicated than with other methods such as TOTP or Yubico OTP, but there are plugins or modules for the most common use cases, for example, a SimpleSAMLphp module.

3.4 Authentication with biometrics

Biometrics are never used plain over the internet because of the difficulties and privacy issues which their usage brings. They are usually used to unlock a secret (cryptographic key) or a token holding such a secret. Such usage would already provide 2FA, but since it is not very much resistant to local attackers, it is still beneficial to combine it with a password or a PIN.

The variety of biometrics used in practice is limited by the commonly available hardware. In the case of mobile phones, the most common are fingerprint readers. Other biometrics are also possible, such

as face scan (using regular or dedicated cameras) or voice recognition (using built-in microphones). On mobile phones, these biometrics can be used to unlock the phone itself or confirm an action inside vendor-specific apps. Apps can hold a secret (for example an encryption key) and use the operating system API to require fingerprint authentication. The security of this solution may still be lowered by the user by registering somebody else's fingerprints or using a finger-resembling token instead.[7]

For the combination of fingerprint and login notifications, I assign high levels to all types of compromise resistance, self-sabotage resistance, and privacy. The downside is still the dependence on mobile operating system vendors and the need to create separate apps, which results in low level in vendor-independence and ease of deployment. I consider the use of fingerprints to be slightly more demanding than just pressing a button, so I assign medium in usability.

Another possibility for incorporating biometrics in authentication is the usage of FIDO U2F devices, some of which offer extra security through biometric protection.[8] The U2F standard defines a way for devices to prove their manufacturer and device family (attestation), so an application may check whether the U2F device used for authentication supports biometrics, or even enforce it.

The combination of biometrics and U2F yields the best result of this comparison, although it is not suitable for the whole user base because of the cost and limited availability – not all U2F devices support fingerprints.

3.5 Evaluation summary

My evaluation is summarized in Table 3.8. There is no single authentication method which would replace passwords and improve in all aspects, so MFA seems appropriate for strengthening security.

Authentication by "something you know" alone does not offer a sufficient level of security. It has to be either combined with some form of a token into MFA or only used to prevent local compromise of a token, like with PINs on credit cards.

7. `https://www.nanotips.com/products/taps-touchscreen-sticker`
8. for example Kensington VeriMark Fingerprint Key

Authentication with tokens or authenticator apps of various kinds are usually secure against remote compromise but easier to compromise locally. To improve, they can be combined with a PIN or a password (like TiQR), or they can be protected with biometric authentication, for example, fingerprint. Authentication text messages should be abandoned because they are not secure and also more difficult and costly to deploy.

Biometric authentication cannot be used over the internet, but it can be conveniently used to protect authentication notifications and hardware security tokens.

The combination of a password and a fingerprint-protected U2F device or login notifications gives the best results at a higher cost, so it is suitable for protecting sensitive information and financial transactions. For regular users, methods such as TOTP, Yubico OTP or U2F offer extra security with little additional costs.

Table 3.8: Authentication methods evaluation

Web auth. method	SFA	MFA		cautious user compromise diff.	irresponsible user compromise diff.	remote user compromise resistance	local user compromise resistance	self-sabotage resistance	privacy	vendor-independence	server compromise resistance	usability	ease of deployment
		gen.	MU										
password	✔	✖	✖	⊙	○	○	○	○	⊙	●	⊙	⊙	●
HOTP	◙	◙	⊗	●	●	●	⊙	●	●	●	○	⊙	●
TOTP	✔	✔	✔	●	●	●	⊙	●	●	●	○	⊙	●
Yubico OTP	✔	✔	✔	●	⊙	●	○	⊙	●	⊙	●	●	●
SMS OTP	◙	◙	⊗	⊙	○	⊙	○	⊙	○	⊙	⊙	⊙	○
notifications	✔	✔	⊗	●	⊙	●	⊙	●	●	○	●	●	○
U2F (UAF)	✔	✔	✔	●	●	●	○	⊙	●	●	●	●	●
TiQR	✔	✔	✔	●	●	●	●	●	⊙	○	●	○	⊙
fingerprint+notif.	✔	✔	✔	●	●	●	●	●	●	○	●	⊙	○
fingerprint+U2F	✔	✔	✔	●	●	●	●	●	●	●	●	⊙	●

✔ = supported, ✓ = viable, ◙ = deprecated, ⊗ = not supported, ✖ = not available, ● = high, ⊙ = medium, ○ = low

4 Environment

This chapter contains a description of the environment of Unified login for which I implemented multi-factor authentication and the components which I built on.

4.1 Unified login

Masaryk University's Unified login[1] serves as an IdP for a number of services with thousands of users, including Inet MU[2], Microsoft (MS) Office 365[3] and the WebKredit application of the Accommodation and Catering Services[4]. SPs obtain not only the user's personal university number (UČO) but also other attributes (like student status) based on the service's requirements.

Several use cases are possible with alternative authentication methods. Multi-factor authentication should be used to allow better security for all users in general. For privileged users (for example those who perform financial transactions), a security token may be preregistered and enforced to prevent impersonation and its consequences. MFA can be forced generally for selected users, but it may also be requested by SPs. This may result in step-up authentication, where technically only the second factor authentication is performed when accessing a sensitive resource. MFA also brings the possibility of allowing federated identities (for example social networks) for logging in because they can be combined with a trusted second factor – a preregistered security token.

Multi-factor authentication also brings some requirements. A website which allows users to setup MFA and register devices has to be created. In the case of implementing preregistered security tokens for privileged users, a procedure has to be established, and the user interface has to support it. There also needs to be a way of dealing

1. `https://id.muni.cz/`
2. the economic and administrative system of Masaryk University (MU), `https://inet.muni.cz/`
3. accounts of students, employees and graduates, `https://o365.muni.cz/`
4. SKM MU, `https://www.skm.muni.cz/`

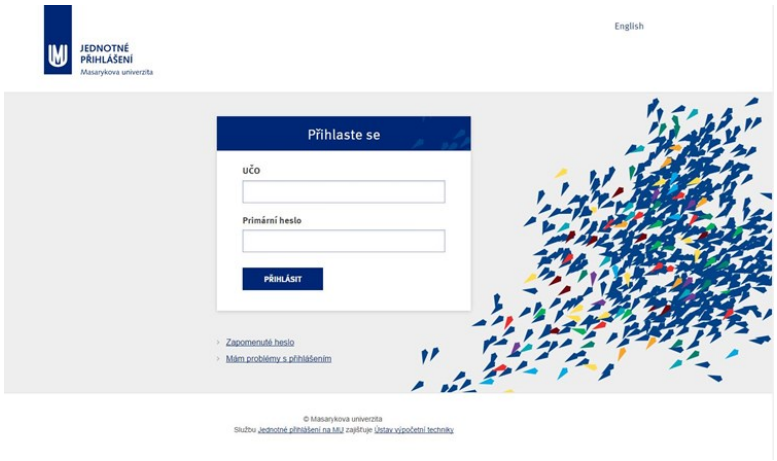

Figure 4.1: Login page with muni theme applied

with lost tokens, because recovery is not as easy as sending a password reset link.

4.2 Technologies

The heart of the Unified login is SimpleSAMLphp[5], an application written in PHP that deals with SAML-based authentication. SimpleSAMLphp and its dependencies are managed by Composer[6].

SimpleSAMLphp follows the SAML protocol; in this case it is configured as an IdP. Its behavior can be modified and extended with modules, which are collections of PHP code together with HTML templates, CSS styles, and Javascript. Modules can define classes, authentication sources, themes, and authentication processing filters.

The login page can be customized using themes, which override default HTML, CSS, Javascript, and PHP files. Unified login uses a custom theme with the university design, which is shown in Figure 4.1.

Modules can also define alternative login methods (authentication sources), which can be used for primary authentication. Modules may also define authentication processing filters, which are executed after the user has authenticated. These filters can alter user attributes or

5. `https://simplesamlphp.org/`
6. dependency manager for PHP, `https://getcomposer.org/`

even display extra web pages, for example, to perform MFA or ask for user consent.

4.3 Testing environment

For testing purposes, there is a separate instance of Unified login, which I used to try several modules providing MFA and when developing modules authswitcher and authapi for controlling MFA on a per-user basis.

5 Development

The implementation of multi-factor authentication in the Unified login and the configuration of modules authswitcher and authapi, which I developed, are described in this chapter.

5.1 Authentication modules for SimpleSAMLphp

My goal was to implement several MFA methods in SimpleSAMLphp. Since there are some modules providing authentication methods available, these should be used instead of implementing everything from scratch.

As described in section 4.2, these modules define authentication sources or authentication processing filters. Authentication sources can only be used as a primary authentication method; most of them offer some form of password-based or federated authentication. But some modules define authentication processing filters, which are executed after a user is authenticated, providing MFA.

The limitation of SimpleSAMLphp and these modules is that the configuration is global and cannot be changed per user. Therefore I had to create a module which is able to run authentication processing filters based on each user's settings. This module is called authswitcher.

Listing 5.1: authswitcher MFA filter

```
1 => [
 'class' => 'authswitcher:SwitchAuth',
 'configs' => [
  'yubikey:OTP' => [
   'api_client_id' => '12345', // Yubico API client ID
   'api_key' => 'abcdefghchijklmnopqrstuvwxyz', // Yubico API key
   'key_id_attribute' => 'yubikey',
   'abort_if_missing' => true,
   'assurance_attribute' => 'yubikeyAssurance',
  ],
  'simpletotp:2fa' => [
   'secret_attr' => 'totp_secret',
   'enforce_2fa' => true,
  ],
 ],
],
```

5.2 authswitcher

The module's purpose is to optionally run one of several predefined authentication processing filters, based on user-specific settings. The MFA itself happens in the filters. The authswitcher module defines an authentication processing filter, whose configuration includes the configurations of the MFA filters, as shown in Listing 5.1.

Listing 5.2: authswitcher configuration

```
<?php
/**
 * This file is part of the authswitcher module.
 */

$config = [
 'dataAdapter' => 'sspmod_authapi_DbDataAdapter',
];
```

The authswitcher module configuration consists of a data adapter, which is a class implementing an interface defined by the authswitcher module. An example configuration is shown in Listing 5.2. The data adapter interface makes the authswitcher module independent of the storage. I also created a small module authapi with a data adapter implementation which reads data from an SQL database (for example MySQL or Oracle), which is described in section 5.5.

5.3 REFEDS assurance framework

In Unified login, most of the REFEDS assurance framework's values may be assigned statically but, for example, the level of identity proofing and credential issuance quality, based on which one of the assurance profiles may also be issued, need to be in sync with the type of authentication performed. For this purpose, the authswitcher module contains an authentication processing filter, which adds the high assurance level and the espresso profile when multi-factor authentication is performed, or medium assurance level and the cappuccino profile otherwise. Its usage is simple, as shown in Listing 5.3.

Listing 5.3: authswitcher filter for REFEDS

```
15 => [
 'class' => 'authswitcher:Refeds',
],
```

5.4 Extending authswitcher

For adding an MFA method, a class has to be created with a name in the form `aswAuthFilterMethod_*modulename*_*filtername*` which extends the `sspmod_authswitcher_AuthFilterMethod`. For example, a helper class for a filter named `bar` of a module named `foo` would look like the one in Listing 5.4. This helper class may perform some preparation before the filter is executed, for example, insert a shared secret in a user attribute for the filter to use.

Listing 5.4: Adding a MFA method helper class

```
<?php
class aswAuthFilterMethod_foo_bar
    extends sspmod_authswitcher_AuthFilterMethod {
 /* ... */
}
```

Data can be obtained by any implementation of the `sspmod_authswitcher_DataAdapter` interface, which is shown in Listing 5.5.

Listing 5.5: DataAdapter interface

```
<?php
interface sspmod_authswitcher_DataAdapter {
    /** The constructor */
    function __construct();
    /** Return an array of sspmod_authswitcher_MethodParams */
    function getMethodsActiveForUidAndFactor($uid, $factor);
    /** Test whether the user can login via SFA, MFA or both. */
    function getMFAForUid($uid);
}
```

5.5 authapi

The authapi module provides an implementation of the `sspmod_authswitcher_DataAdapter` interface which uses an SQL database. Similarly to authswitcher, the module contains a configuration template, which is shown in Listing 5.6.

Listing 5.6: authapi configuration template

```
<?php
/**
 * This file is part of the authapi module.
 */

$config = array(
```

```
    'dsn' => 'mysql:dbname=foobar;host=127.0.0.1',
       // change to match your database settings
    'user' => 'foo',
       // change to database username
    'pass' => 'bar',
       // change to database password
    //'mfa_table' => 'auth_method_setting', // default
);
```

5.6 Deployment

The installation of both modules is quite simple since they have no extra dependencies, other than SimpleSAMLphp and the MFA modules. They can be installed via composer, by cloning from git, or by copying the files.

Configuration of the authentication processing filters from Listing 5.1 can be placed globally in `config/config.php`, for each IdP in `metadata/saml20-idp-hosted.php`, or for each SP in `metadata/saml20-sp-remote.php`.

Configuration of the data adapter from Listing 5.2 has to be placed in a new file called `config/module_authswitcher.php`, which can be created by copying the template from `config-templates/module_authswitcher.php`.

Similarly, configuration with database credentials from Listing 5.6 has to be placed in a new file called `config/module_authapi.php`, which can be created by copying the template from `config-templates/module_authapi.php`.

5.7 Testing

I tested MFA using authswitcher together with SimpleTOTP[1] for TOTP and the YubiKey module[2] for Yubico OTP. The authapi module was used to access the database, which I prefilled with my shared secret for TOTP and the public ID of my YubiKey for Yubico OTP. After authentication, either a screen asking for TOTP (Figure 5.1) or Yubico OTP (Figure 5.2) is shown.

1. `https://github.com/aidan-/SimpleTOTP`
2. `https://github.com/simplesamlphp/simplesamlphp-module-yubikey`

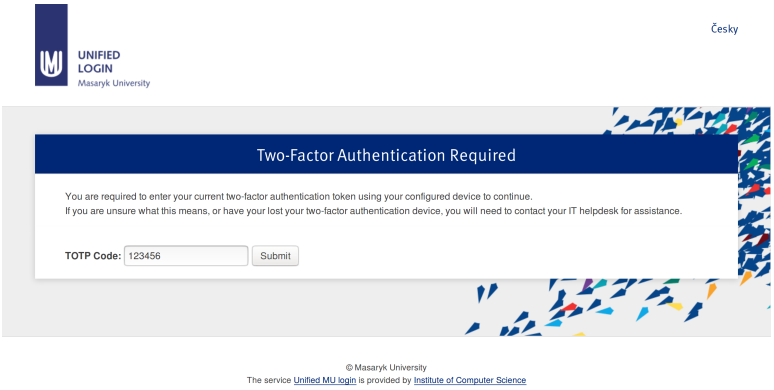

Figure 5.1: Login page asking for TOTP

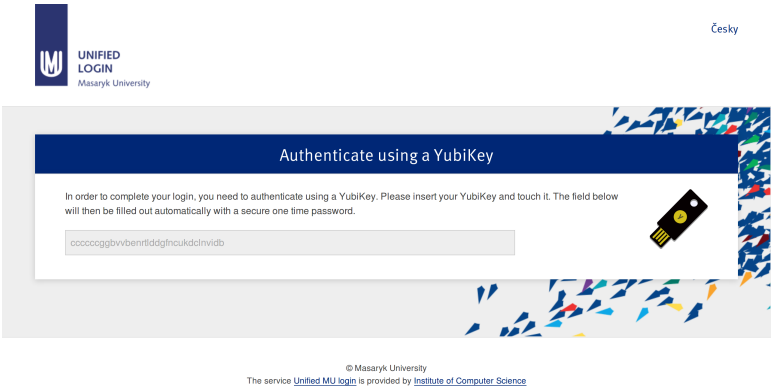

Figure 5.2: Login page asking for Yubico OTP

5.8 Next steps

The authswitcher module is ready for production to provide MFA. The authapi module provides the necessary data from a database, and it also includes an API for retrieving and modifying users' settings.

A graphical user interface has to be created. For general MFA, each user has to be able to turn MFA on and register TOTP devices, YubiKeys, etc. The user interface should also allow, for example, top management to preregister tokens for privileged users or users logging in with external identities. After the user interface is ready, MFA should be available for all Unified login users.

6 Conclusion

This work provides a comparison of several authentication methods suitable for multi-factor authentication on the web in large organizations. The list is not exhaustive, but given the evaluation criteria provided here, new authentication methods can be evaluated and compared with these results.

It has been shown that passwords do not perform well from the security perspective, but none of the other authentication methods can replace them without increasing cost or demands on users, or worsening availability or usability. From this it followed that multi-factor authentication is highly recommended, because some combinations of authentication methods eliminate most of the weaknesses. I identified several methods which are suitable for multi-factor authentication in large organizations.

I demonstrated that it is possible to introduce multi-factor authentication into a large organization with thousands of users, such as Masaryk University. I implemented two-factor authentication using TOTP and Yubico OTP in the Unified login identity provider. For switching between authentication methods on a per-user basis, I developed the module authswitcher for SimpleSAMLphp, which can be used not only in the Unified login but also in other organizations. Two-factor authentication using TOTP and Yubico OTP has been tested in the testing environment of Unified login.

When a graphical user interface is created to set up multi-factor authentication and register devices, the new authentication methods should be available to all users of Unified login. If new authentication methods emerge, such as WebAuthn, they can be added to the authswitcher configuration.

Bibliography

[1] A. Pfitzmann and M. Hansen, *A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management*, v0.34, Aug. 2010. Available from: http://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.34.pdf (visited on 04/04/2019).

[2] K. Rannenberg, D. Royer, and A. Deuker, Eds., *The future of identity in the information society, Challenges and opportunities*. Berlin Heidelberg: Springer-Verlag Berlin Heidelberg, 2009, xvi, 508, ISBN: 978-3-540-88480-4.

[3] Liberty Alliance Project, "Business benefits of federated identity", Apr. 2003. Available from: http://www.projectliberty.org/liberty/content/download/382/2705/file/LAP%20Business%20Benefits%20White%20Paper%20v4.pdf (visited on 04/19/2019).

[4] Y. Beres, A. Baldwin, M. C. Mont, and S. Shiu, "On identity assurance in the presence of federated identity management systems", in *Proceedings of the 2007 ACM Workshop on Digital Identity Management*, ser. DIM '07, Fairfax, Virginia, USA: ACM, 2007, pp. 27–35, ISBN: 978-1-59593-889-3. DOI: 10.1145/1314403.1314409. Available from: http://doi.acm.org/10.1145/1314403.1314409 (visited on 04/17/2019).

[5] P. A. Ashley, S. R. Muppidi, and M. Vandenwauver, "Method and system for stepping up to certificate-based authentication without breaking an existing ssl session", Patent US 7,395,424 B2, Jul. 1, 2008. Available from: http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO1&Sect2=HITOFF&p=1&u=/netahtml/PTO/srchnum.html&r=1&f=G&l=50&d=PALL&s1=7395424.PN. (visited on 04/04/2019).

[6] E. Kennedy and C. Millard, "Data security and multi-factor authentication: Analysis of requirements under eu law and in selected eu member states", *Computer Law & Security Review: The International Journal of Technology Law and Practice*, vol. 32, pp. 91–110, 2016, ISSN: 0267-3649. Available from: http://ezproxy.muni.cz/login?url=https://search.ebscohost.com/login.

aspx?direct=true&AuthType=ip,cookie,uid&db=edselp&AN=S0267364915001697&lang=cs&site=eds-live&scope=site.

[7] 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 & Security*, vol. 30, no. 4, pp. 208–220, 2011, ISSN: 0167-4048. Available from: http://ezproxy.muni.cz/login?url=https://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,cookie,uid&db=edselp&AN=S0167404810001148&lang=cs&site=eds-live&scope=site (visited on 04/20/2019).

[8] Microsoft. (Nov. 20, 2018). Using app passwords with apps that don't support two-step verification, Available from: https://support.microsoft.com/en-us/help/12409/microsoft-account-app-passwords-and-two-step-verification (visited on 04/04/2019).

[9] J. Bonneau, C. Herley, P. C. v. Oorschot, and F. Stajano, "The quest to replace passwords, A framework for comparative evaluation of web authentication schemes", University of Cambridge, Computer Laboratory, Tech. Rep. UCAM-CL-TR-817, Mar. 2012. Available from: https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-817.pdf (visited on 04/11/2019).

[10] S. Chiasson and P. van Oorschot, "Quantifying the security advantage of password expiration policies", *Designs, Codes, and Cryptography*, vol. 77, no. 2-3, pp. 401–408, 2015, ISSN: 15737586. Available from: http://ezproxy.muni.cz/login?url=https://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,cookie,uid&db=edselc&AN=edselc.2-52.0-84942991130&lang=cs&site=eds-live&scope=site (visited on 04/12/2019).

[11] A. Birgisson, "The ultimate account security is now in your pocket", *The Keyword*, Apr. 10, 2019. Available from: https://blog.google/technology/safety-security/your-android-phone-is-a-security-key/ (visited on 04/12/2019).

[12] "Security of biometric authentication systems", *International Conference on Computer Information Systems and Industrial Management Applications (CISIM) 2010*, p. 19, 2010, ISSN: 978-1-4244-7817-0. Available from: http://ezproxy.muni.cz/login?url=https://search.ebscohost.com/login.aspx?direct=true&

AuthType=ip,cookie,uid&db=edseee&AN=edseee.5643698&lang=cs&site=eds-live&scope=site (visited on 04/17/2019).

[13] A. Bhargav-Spantzel, A. C. Squicciarini, and E. Bertino, "Establishing and protecting digital identity in federation systems", *Journal of Computer Security*, vol. 14, no. 3, pp. 269–300, 2006, ISSN: 0926227X. Available from: http://ezproxy.muni.cz/login?url=https://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,cookie,uid&db=a9h&AN=21349011&lang=cs&site=eds-live&scope=site (visited on 04/19/2019).

[14] A. Bhargav-Spantzel, J. Camenisch, T. Gross, and D. Sommer, "User centricity: A taxonomy and open issues", *Journal of Computer Security*, vol. 15, no. 5, pp. 493–527, 2007, ISSN: 0926227X. Available from: http://ezproxy.muni.cz/login?url=https://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,cookie,uid&db=a9h&AN=25850726&lang=cs&site=eds-live&scope=site (visited on 04/04/2019).

[15] C. Mainka, V. Mladenov, J. Schwenk, and T. Wich, "SoK: Single Sign-On Security — An Evaluation of OpenID Connect", *2017 IEEE European Symposium on Security and Privacy*, p. 251, 2017, ISSN: 978-1-5090-5762-7. Available from: http://ezproxy.muni.cz/login?url=https://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,cookie,uid&db=edseee&AN=edseee.7961984&lang=cs&site=eds-live&scope=site (visited on 04/17/2019).

[16] J. Rapoza, "SAML unlocks door to Web services", *EWeek*, vol. 19, no. 49, p. 62, 2002, ISSN: 15306283. Available from: http://ezproxy.muni.cz/login?url=https://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,cookie,uid&db=a9h&AN=8646574&lang=cs&site=eds-live&scope=site (visited on 04/17/2019).

[17] RFC Editor, "The OAuth 2.0 Authorization Framework", RFC 6749, Oct. 2012. Available from: https://www.rfc-editor.org/rfc/rfc6749.txt (visited on 04/17/2019).

[18] OpenID, "OpenID Connect FAQ and Q&As", *WELCOME TO OPENID CONNECT*, 2019. Available from: https://openid.net/connect/faq/ (visited on 05/15/2019).

[19] N. Sakimura, J. Bradley, M. Jones, B. de Medeiros, and C. Mortimore, "OpenID Connect Core 1.0 incorporating errata set 1",

Tech. Rep., Nov. 8, 2014. Available from: https://openid.net/specs/openid-connect-core-1_0.html (visited on 04/19/2019).

[20] REFEDS. (2019). REFEDS, Available from: https://refeds.org/ (visited on 05/06/2019).

[21] ——, "Refeds assurance framework ver 1.0", Tech. Rep., Oct. 11, 2018. Available from: https://wiki.refeds.org/x/HYBRAg (visited on 05/06/2019).

[22] P. KADLECOVÁ, *Motivace uživatelů používat bezpečná hesla*, Bachelor thesis, 2011. Available from: https://is.muni.cz/th/jrfkb/ (visited on 04/22/2019).

[23] RFC Editor, "TOTP: Time-Based One-Time Password Algorithm", RFC 6238, May 2011. Available from: https://tools.ietf.org/html/rfc6238 (visited on 05/08/2019).

[24] P. A. Grassi, J. L. Fenton, E. M. Newton, R. A. Perlner, A. R. Regenscheid, W. E. Burr, and J. P. Richer, *NIST Special Publication 800-63B: Digital identity guidelines, Authentication and lifecycle management*, National Institute of Standards and Technology, Jun. 2017. Available from: https://doi.org/10.6028/NIST.SP.800-63b (visited on 04/12/2019).

A List of electronic attachments

This attachment describes the content of the archive file available from the thesis repository of Masaryk University.

A.1 simplesamlphp-1.16.3

- SimpleSAMLphp in version 1.16.3, for which `authswitcher` was developed
- obtained from GitHub by
  ```
  git clone https://github.com/simplesamlphp/simplesamlphp.git
  && cd simplesamlphp
  && git checkout v1.16.3
  ```
- to be extracted into `simplesamlphp`
- dependencies have to be installed with
  `php composer.phar install` or `composer install`

A.2 simplesamlphp-1.16.3-installed

- SimpleSAMLphp in version 1.16.3, for which `authswitcher` was developed, with dependencies already installed
- obtained from GitHub by
  ```
  git clone https://github.com/simplesamlphp/simplesamlphp.git
  && cd simplesamlphp
  && git checkout v1.16.3
  && composer install
  ```
- to be extracted into `simplesamlphp`

A.3 authswitcher

- module for SimpleSAMLphp
- setup is described in `README.md`

A.4 authapi

- module for SimpleSAMLphp
- setup is described in `README.md`

Acronyms

2FA two-factor authentication. 5, 21, 25, 27, 29

2SV two-step verification. 5

AS authorization server. 10

HOTP HMAC-based one-time password. 6, 21–23, 32

IdP identity provider. 3, 9, 10, 12, 17, 33, 34

MFA multi-factor authentication. 4, 5, 15, 27, 30, 33, 35, 37–40, 42

MS Microsoft. 33

MU Masaryk University. 33

OIDC OpenID Connect. 10–12

OTP one-time password. 6, 23–25, 29, 31, 32, 40, 43

REFEDS Research and Education FEDerations. 11, 12

RP relying party. 10

SAML security assertion markup language. 9–12, 34

SFA single-factor authentication. 5, 15, 27

SP service provider. 3, 9, 10, 12, 33

SSO single sign-on. 9

TOTP time-based one-time password. 6, 21–23, 26, 29, 31, 32, 42, 43

U2F universal 2^{nd} factor. 6, 27, 28, 30, 31

UČO personal university number. 33