Multi-factor Authentication

Overview paper about state-of-the-art multi-factor authentication systems and mechanisms

Version 1.0, 17.08.2016

Andreas Fitzek – andreas.fitzek@egiz.gv.at

Summary: This document provides an overview of two-factor authentication. It presents state-of-the-art authentication mechanisms and frameworks, analyses implementation variations and discusses the agile authentication, the next generation of authentication services.
# Table of contents

1 Introduction ............................................................................................................................ 3

2 Multifactor Authentication ................................................................................................... 5
   2.1 Knowledge ....................................................................................................................... 6
   2.2 Possession ......................................................................................................................... 6
   2.3 Inherence ......................................................................................................................... 8
   2.4 Combination .................................................................................................................... 9

3 Standards & Frameworks .................................................................................................... 10
   3.1 NIST Electronic Authentication Guideline ................................................................. 10
   3.2 ISO/IEC 29115 ................................................................................................................. 11
   3.3 Authentication protocols ............................................................................................. 12

4 Implementation Services .................................................................................................... 15
   4.1 Shared account ............................................................................................................ 15
   4.2 Second factor support ................................................................................................. 15
   4.3 Combined authentication .......................................................................................... 15
   4.4 Privately hosted ............................................................................................................. 15

5 Agile Authentication ........................................................................................................... 16

6 Summary ................................................................................................................................ 19
1 Introduction

The modern world is connected through the Internet which is used for a great deal of activities such as reading the news, interacting in a social network, making a wire transfer, or accessing our health data. Many online services need users to log into the service in order to augment the functionality or to provide the functionality at all. How do online services determine the identity of the current user? The user has to authenticate herself against the system. We define authentication as “The process of establishing confidence in the identity of users or information systems.” [NISTEAG] The fast growth in the number of online services provokes an increasing number of various authentications the user needs to control.

Classic authentication is implemented by a shared secret between the system and the user. The most common form of this authentication is the combination of username and password. Only the user and the system know the user’s password. The user enters her password, it is transferred to the system and the system verifies the password. From this authentication the local authentication model can be generated. The local authentication model consists of three actors. The claimant is the user that claims a certain identity. The transport channel needs to protect the secret information during transfer, and the verifier, is the system, that also knows the password and verifies its correctness.

Because of the fast growth in the number of online services, users need to remember an increasing amount of username and password combinations. Users tend to reuse passwords for multiple services, and choose simple and easy to remember passwords. To reduce the amount of username and password combinations users have to remember, the concept of identity providers was developed. In this concept online services share accounts. An identity provider is an online service, that authenticates users and vouches to other online services for the identity of the user. The external authentication model is an extension of the local authentication model. The claimant and the transport channel keep their roles, but the verifier is externalized from the online service. Another online service verifies the username and password combination. The verifier transmits an assertion to the relying party to vouch for the
identity of the user. The relying party is the actual service, that wanted the user to be authenticated.

With external authentication, authentication for multiple online services can be done with one set of credentials. This increases the value for attackers when the credentials can be stolen. These credentials needs to be strengthened. If the authentication only relies on a single credential, like a secret password, the authentication is called single-factor authentication. To strengthen the authentication, multi-factor authentication was developed. Multi-factor authentication is an authentication where at least two credentials from different authentication categories have to be provided to the verifier. In general, authentication categories are something a user knows, something a user has, and something a user is.
2 Multifactor Authentication

During an authentication the user has to provide credentials, that builds up the confidence in the claimed identity of the user. Most electronic systems still rely on one such a credential, typically a password. The problem with passwords was discussed lengthy in other works already [PASSWORD]. The common problem is that the passwords are not complex enough to withstand brute force attacks. Another problem is that many users reuse passwords at multiple electronic systems. This multiplies the impact of a lost or leaked password. The password is something the user and the electronic system both know. By telling the system the password, the user proofs, that she knows the password. If the authentication solely relies on a single credential, the authentication is a single factor authentication. Because of the growing computational power to run brute force attacks, the rise of malware and key loggers, and in general computer attacks, the confidence in authentication solely based on a single authentication factor shrinks constantly.

To strengthen the confidence in the authentication, multiple credentials should be used during an authentication. This is called multifactor authentication. Multifactor authentication is recommended by global authentication standards, if the authentication should receive a higher assurance level, like the NIST EAG [NISTEAG], the ISO 29115 [ISO29115] and also the new eIDAS regulation in the EU [EIDAS], which was aligned to some extent to the ISO standard.

Multi-factor authentication requires the user to provide multiple credentials from different authentication types. In general, there are three categories of authentication types: something the user knows (knowledge), something the user is (inherence), and something the user has (possession). We discuss these categories later on. Also new categories are developed. Examples are e.g. somebody you know [FOURTH].

By requesting multiple authentication factors the confidence in the authentication process is increased. It provides higher protection against attacks that target the collection of the credentials, or the impersonation of users. As an example a key logger is enough to capture a password. If the password is the only authentication factor, the attacker has enough information to access the account. Multifactor authentication provides another layer of security. In addition to the knowledge factor (in example the password) another factor is requested. The user might have to proof the possession of another device. This could be implemented as an SMS-TAN. In this case the user receives an SMS with a OTP. This OTP is unique for each authentication process and has to be entered additionally to the password. An attacker needs to gain access to the device used to enter the password and the device that receives the SMS to gain access to the credentials. By requesting multiple authentication
factors an attack becomes more and more complicated. A potential attacker needs to compromise more and more devices.

But the combination of these authentication factors is critical to protect the user against attacks and to actually increase the assurance in the identity. For example, if a multi factor authentication is in place as combination of a password and a SMS-TAN. A user has to be aware, that a malware that is located on the mobile device, that receives the SMS-TAN, might also have the ability to capture the password as it is entered. If the user uses this device to log into the system, the malware is able to capture the password and can forward the SMS-TAN messages to an attacker. This idea can be extended at will, by adding new authentication factors and new authentication methods. These authentication factors might become very complex, especially if a biometric authentication is used. [BIOREVIEW] There also exist many token based authentication systems, that can be integrated into authentication systems. [RFC 2289, RFC 4226, RFC 6238]

Currently three categories of authentication factor are widely accepted. In the following sections we describe these categories and their advantages and disadvantages.

2.1 Knowledge
A knowledge based authentication factor means, that an entity shares a secret with another entity. An appropriate way to prove the knowledge of this shared secret has to implemented to successfully authenticate with a knowledge based factor.

The simplest and common implementation of this concept is a password. An entity knows a password and tells the authenticating entity the password as a prove.

2.2 Possession
A possession based authentication factor means, that an entity can prove that she is in possession of something. The prove of this possession is mostly done using cryptography.

One-Time-Passwords (OTPs) are a famous example of possession based authentication. The oldest example for these passwords are printed TAN lists. With these lists a system and the user share the same list of TAN codes. The user usually possesses a hard copy of the TAN list. The systems ask the user to enter the TAN with a specific number, if the user enters the correct code he proves the possession of the TAN list.

Electronic forms of these lists were developed, these are so called HMAC-based One-time Password Algorithm (HOTP) [RFC 4226]. These algorithms can produce codes, which are 6-8 numerical digits long, based on a secret key and an additional input.
An HMAC of the secret key and the input is calculated and the result is truncated to generate a readable TAN code. Instead of sharing a regenerated list of TAN codes, a system can share such a secret key with a device of the user. The device is able to generate a TAN code based on the shared secret and either on a counter, which is incremented at each use, or based on the current timestamp (Time-based OTP) [RFC 6238], or a random code generated by the system for each authentication request.

The premise for HOTPs is that it is hard to steal the secret key from the user’s device, and to use the secret key on the user’s device. If this premise can hold, the valid entry of a TAN code, proves the possession of the user device.

A prominent example of a TOTP device is the Google Authenticator application\(^1\). The Google Authenticator is available for Android, IOS and BlackBerry. It stores the secret information in the applications private storage on android devices. This protects these secret keys from access as long as the device is not rooted. If the device is rooted, the secret key database can be extracted and copied. This might be sufficient if the devices are managed inside a company via device management and monitored by system administrators. For a broad rollout like an e-Government component this will not be sufficient, since the system cannot determine if the actual device used for storing the secret key is compromised.

To achieve a better protection of these secret keys, hardware tokens should be used. One example of such a token is the Yubikey. These tokens are designed to store and calculate HOTP codes. These tokens protect the secret keys, from extraction and can only be used if the token is connected to the device. Therefore, it is a much better prove of possession, in this case of the token.

All HOTP based solution target the generation of human readable codes, which could be entered into a user interface. But also other proves of possession can be achieved. Many modern smartphones have the ability to bind cryptographic keys to a hardware module integrated into the smartphone. This binding ensures that a cryptographic key cannot be extracted from the device. This fact can be used to prove the possession of the device. To prove the possession of the device the user has to prove that she can use the cryptographic key. This is commonly done, by using the cryptographic key to create a signature. The resulting signature is usually too complex and too long to be transferred to the authentication system manually. But the device can transfer the digital signature directly via a network connection, this requires the device to be actually connected to the authentication system.

\(^1\) https://play.google.com/store/apps/details?id=com.google.android.apps.authenticator2&hl=de
If the smartphone is under complete control of an attacker, the key cannot be extracted by means of software, but potentially used by this attacker. To further reduce the risk, multiple smart devices can be combined to work together. For example, a smartphone and a smartwatch could both know one part of a multi part secret. Only if both devices work together, a valid digital signature could be created. These devices can be connected locally via network, or Bluetooth, or near field communication (NFC). This would require an attacker to gain access to both devices. It strongly depends on the concrete security requirements, how this possession should be implemented.

A practical problem is the huge diversity of devices and platforms, which is available. And therefore the different security mechanisms available on these platforms differ.

In an e-Government context all devices have to be supported, because a basic requirement of e-Government is that everybody has the right to participate. The fallback authentication mechanisms to include everybody is going to be the SMS based TAN. We assume that everybody with a phone has the ability to receive SMS messages. Considering the Austrian e-Government, server-based implementations of the citizen card (also known as mobile phone signature), should therefore define certain minimum criteria for supported platforms. A risk analysis of these criteria should define the potential risks of smartphone based factors. Multiple factors with different levels of security should be available to users. Depending on the available platform and available devices the system should detect the most secure factor available and deploy it to the user. Factors should be renewed in certain intervals and at each renewal the evaluation of the factor should be done again. If the security of a certain factor drops below the minimum criteria defined, because of new developments, the factor cannot be renewed again. This allows for an automatic transition of the user to a more secure factor.

2.3 Inherence

An inherence based authentication factor means, that an entity can prove that she inherits some unique characteristic. An example for such a characteristic is a fingerprint.

In general, biometric authentication is done using heuristics. A pattern is recorded from the entity and this pattern is matched against a known pattern. If the patterns match, the entity is authenticated.

A problem arises during the transfer of the recorded pattern. It has to be ensured, that the recorded pattern is actually recorded now from a trusted hardware (fingerprint scanner, iris scanner or webcam). For example, a server tries to authenticate a user based on his fingerprint. Therefore, a web service is developed which accepts an image of the user’s fingerprint and allows access to the server if
the fingerprint in the image matches a fingerprint in a database. An attacker does not need access to the actual finger, but only to an image of the fingerprint. The attacker does not have to fool the fingerprint scanner, but just post an image to the server. The server cannot determine if the provided image was actually recorded from a fingerprint scanner. To overcome this issue, the actual authentication against the server is leveraged to a cryptographic key. The server only receives a digital signature created with a secret key, where the client ensures, that the cryptographic key can only be used if biometric authentication was successful on the client.

Another problem arises if the scanner can be tricked into providing a false positive. For example, a face recognition might be tricked, by presenting a picture of the victim to the camera, or a fingerprint reader could be tricked to accept a 3d printed fingerprint of the victim.

Once a biometric characteristic of a user is reconstructed it cannot be changed. This is a major drawback: if a user recognizes, that a password is not secret anymore, she can simply change it, but she cannot change her biometric characteristic.

2.4 Combination

When combining multiple factor types into one factor, very strong and usable authentication factors can be created. As an example, the mobile phone of users can be used to act as a proof of possession. To achieve this, a cryptographic key is bound to the device. To perform a successful authentication, the server sends a random number to the phone and the phone has to respond with a message, which contains the encrypted and digitally signed random number, with respect to the devices cryptographic key. If an attack gains access to the device, the attacker would be also able to use the cryptographic key. To strengthen this factor, the cryptographic key could be protected by a passphrase. Only if the correct passphrase is entered the cryptographic key could be used. If this passphrase is a simple four-digit code an attacker could easily brute force the code, when she has access to the device. From a user’s point of view, a more complex passphrase is going to decrease the usability and the acceptance of the factor. One possible way to overcome this problem and keep using the four-digit PIN code, is to not have to PIN code unlock the cryptographic key, but to include the PIN code in the response to the server. In this case only the server can perform the verification of the correct PIN. If an attacker tries to brute force the PIN code, the server would detect multiple false tries and could flag the device as not trusted.

2 https://www.ccc.de/de/updates/2014/ursel
3 Standards & Frameworks

In the definition of authentication, we find the confidence in the identity. Service provides grant access to users based on the “confidence in the identity”. Therefore, based on the risk of granting access to a wrong user the service provider needs to communicate to the authentication system how confident the system needs to be for a successful authentication. To compare the levels of confidence from different authentication systems, we need more detailed definitions of these levels. The following standards define these levels.

3.1 NIST Electronic Authentication Guideline

The U.S. National Institute of Standards and Technology (NIST) defines a framework with more detailed specifications on authentication in the Electronic Authentication Guideline (EAG) [NISTEAG]. The EAG defines a complete authentication model, and the needed processes in this model to perform an authentication. The model defined by the EAG stretches from the initial registration process, to the credential management, to the actual authentication process. It defines requirements for all parts and communications of the model. Based on these requirements the EAG defines assurance levels, which are used to value the “confidence in the identity”. The levels range from one to four.

- **Level 1:**
  This level provides the lowest value of confidence. It requires no identity proofing, but the authentication has to use a secure authentication protocol and the proof of the user that she controls or possesses the credential. But there are no cryptographic methods that block offline attacks against the credential.

- **Level 2:**
  This level provides remote single factor authentication. It requires basic identity proofing, by presenting identifying information. This level has to resist against online guessing, replay, session hijacking, and eavesdropping attacks.

- **Level 3:**
  This level provides remote multi-factor authentication. At this level the identity proofing requires verification of the identifying information. It requires a proof of possession of tokens by cryptographic means.

- **Level 4:**
  This level provides remote multi-factor authentication. At this level the identity proofing requires in-person identity proofing. The authentication is similar to level 3 but the tokens have to be hardware tokens, that are validated at Federal Information Processing Standard (FIPS) 140-2 Level 2 or higher.
3.2 ISO/IEC 29115

The International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC) also define levels of confidence in the identity in the standard ISO/IEC 29115 [ISO29115]. Like the NIST EAG, ISO/IEC 29115 also defines four levels of assurance (LoA):

- **Level 1 - Low:**
  Little or no confidence in the claimed or asserted identity.
  The identity at this level does not require any identity proofing methods. This level does not require any cryptographic measures to protect the authentication credentials. It therefore provides only a minimum level of assurance in the claimed identity. Examples for this level are an authentication with username and password, or an access control via a device’s media access control (MAC) address.

- **Level 2 - Medium:**
  Some confidence in the claimed or asserted identity.
  This level requires identity information from an authoritative source, but no verification of the identity information has to be done. At this level authentication credentials have to be protected against offline attacks. The usage of the authentication credentials has to be protected by a secure authentication protocol, that protects the credentials from eavesdropping and online guessing attacks. Single-factor authentication is still acceptable for this level.

- **Level 3 - High:**
  High confidence in the claimed or asserted identity.
  This level requires identity information from an authoritative source that is verified. In this level authentication shall be based on multifactor authentication. Secret information of the authentication credentials shall be protected by cryptographical means during transit and at rest.

- **Level 4 - very High:**
  Very high confidence in the claimed or asserted identity.
  This level requires identity information from multiple authoritative sources that is verified and in case of a human entity and in-person witness. This method for identity proofing cannot be performed remotely. The same requirements apply to secret information as in LoA 3, with the addition that the information has to be stored in a tamper-resistant hardware. Furthermore, all sensitive data during an authentication protocol shall be protected by cryptographical means during transit and at rest.

Higher LoAs can always be used to fulfil requirements of lower LoAs as well.
3.3 Authentication protocols

With NIST EAG and ISO/IEC 29115 two standards are available, that provide a scale to measure the confidence in authentications. For applications to communicate their authentication parameters to authentication systems and to receive the authentication result, standardized communication protocols are needed. There are several standardized authentication protocols defined. In general, all these protocols build upon a similar authentication model. This model consists of three entities: the identity provider (IdP), the service provider (SP) and the end user. The IdP performs the actual authentication and assures the SP with a certain degree of confidence that the end user's identity is the claimed identity. The SP sends an authentication request to the IdP. The IdP runs the actual authentication procedure with the end user. The result of the authentication is sent back to the SP as authentication response.

In this section we discuss the most common and widely used authentication protocols and only the current version of each authentication protocol.

1. **Security Assertion Markup Language 2.0 (SAML2) [SAML2]**
   The Security Assertion Markup Language 2.0 (SAML2) is an XML based authentication protocol developed by the Organization for the Advancement of Structured Information Standards (OASIS). SAML2 is an open standard and therefore accessible for everybody. It is designed to identify a user for a SP through an IdP.
   SAML2 defines XML messages for authentication requests and authentication responses. Furthermore, SAML2 defines metadata formats for SPs and IdPs. The metadata describes all kinds of information of the SP or the IdP including the features supported, the used signing keys, the protocol endpoints and so on. SAML2 defines three methods for transferring the messages from the IdP to the SP and back.
   SAML2 defines support for many advanced features, like single-sign-on (SSO), single-log-out (SLO), and automatic service discovery.

2. **The OAuth 2.0 Authorization Framework (OAuth2) [RFC6749]**
   OAuth2 is defined in RFC 6749. The purpose of OAuth2 is to allow an end user to authorize an application to access a resource on behalf of the end user. The OAuth2 protocol is based on JavaScript Object Notation (JSON).
   In OAuth2 the end user authenticates with an authorization server. The end user provides the client application with an authorization grant. With this authorization grant the client application requests an access token from the authorization server. With this access token the client application requests the protected resource from the resource server. The resource server can validate the access token and decides if access to the protected resource is granted or not. OAuth2 is less complex than SAML2, OAuth2 does not define all advanced features that are available for SAML2.
3. **OpenID Connect [OIDC]**
   OpenID Connect is an authentication protocol that is built on top of OAuth2. OpenID is used to identify an end user for a SP through an IdP. OpenID consists of multiple specification documents similar to SAML2 and is able to provide a comparable feature set, include single-sign-on, single-logout and discovery.

4. **Simple Authentication and Security Layer (SASL) [RFC4422]**
   SASL defines an interface that decouples authentication mechanisms from application protocols [RFC4422]. It defines abstract messages, which have to be defined for each application protocol. With these messages, the communication partners perform authentication using one of the authentication mechanisms. These mechanisms are identified by names, which are consistent across the different application protocols. These mechanisms are to be registered with the Internet Assigned Numbers Authority (IANA).

5. **Fast Identity Online (FIDO)**
   FIDO is an international alliance that develops specifications for online authentication. FIDO defines two main concepts, which are the Universal Authentication Framework (UAF) [FIDOUAF] and the Universal Second Factor (U2F) [FIDOU2F].
   
   a. **The UAF** is designed to replace password based authentications of entities against a remote system. It provides the abstract concept of authenticators. These authenticators can create asymmetric cryptographic keys, which are used to digitally sign authentication requests from a remote system. The authenticators protect these keys. These keys can only be used if the user is able to unlock the authenticator, by some action. This might be a password, or a biometric characteristic. Only when the user is able to unlock the key in the authenticator, the authenticator can use the key to digitally sign the authentication request from the remote system and therefore authenticate the user against the remote system. UAF knows two kinds of authenticators. There are authenticators, which can only act as a second factor. These can only strengthen the authentication of the user, but these authenticators cannot identify the user. The second kind of authenticators, can be used for both use cases. They can authenticate and identify a user.

   b. **The U2F** is designed to augment an existing username password based authentication process, by adding a second factor of authentication. U2F provides standardized communication messages, and transport bindings, to register, authenticate and deregister a U2F security token. U2F also works with authenticators, in a similar way as UAF does. To perform an authentication against a remote system a cryptographic key is used to digitally sign an authentication request. Access to this
cryptographic key to create the signature is restricted by any form the vendor of the authenticator implements. Again this can be the simple possession of a token, or a biometric characteristic like a fingerprint. Once access to the cryptographic key is granted by the authenticator, the cryptographic key is used to sign the authentication request, which then authenticates the user.
4 Implementation Services

Multifactor authentication becomes more and more important for modern systems. The website [https://twofactorauth.org/](https://twofactorauth.org/) provides an overview of online services, that support multifactor authentication for their accounts. The integration of multifactor authentication is still costly. Therefore, services appeared that allow externalization of multifactor authentication. Different kinds of these services exist.

4.1 Shared account

One method to externalize multifactor authentication is to use a preexisting account that already supports multifactor authentication. The most famous examples of this are Google and Facebook. Google and Facebook both offer the use of their authentication infrastructure. Both services can be integrated using OpenID connect. During an authentication the user performs a login with her Google or Facebook account. The result of the authentication is transmitted to the integrating application. Both accounts support the use of multifactor authentication. The downside of this approach is, that it cannot guarantee the use of multifactor authentication during the login procedure.

4.2 Second factor support

To simply augment to preexisting authentication, supporting services can be used. One example of such a service is the YubiKey Cloud. It provides a web based interface to perform YubiKey authentication, without detailed knowledge about the YubiKey functions. This web based interface can be used to bind an existing user account to a specific YubiKey token.

4.3 Combined authentication

Another possibility to externalize authentication is to use a well implemented identity provider (broker), that combines different authentication sources and other external providers. One example of such a service is auth0. It integrates many external accounts like Google and Facebook, and also integrated factors like TouchID, a biometric authentication system. The integration into an existing system is done using OpenID connect or the OAuth2 protocol.

4.4 Privately hosted

Another possibility is to still externalize the authentication, but to a self-hosted service. This self-hosted service acts as an identity provider. The most famous open source implementation is Shibboleth. To support multifactor authentication, Shibboleth provides the MFA interface. This allows to implement and integrate multiple authentication factors into a Shibboleth instance.
5 Agile Authentication

There are various problems that arise with multi factor authentication. Multifactor authentication can cost more than simple authentication with a password. It may require more infrastructure, it may require special hardware for each user, development costs or even cost per authentication, for example the sending of an SMS.

Another problem is the very heterogeneous user environment. Users may use different operating systems, or client’s software. Users may have access to different hardware devices, like card readers, NFC readers, or biometric readers. Users may also have different smartphones (IOS, Android, Windows Phone, Blackberry, etc.) with different capabilities. Authentication systems should be able to consider these different environments and try to take full advantage of the different possibilities of these different environments.

Another problem may arise in the integration of multifactor authentication into existing systems.

Agile authentication aims to help authentication providers to improve their services. Current research creates authentication systems, based on the described technologies and standards in section 3.3, that provide flexible multi-factor authentication, that try to overcome the above problems or reduce them as much as possible.

The core idea is, that one main IdP aggregates assertions from other IdPs. The main IdP implements some sort of policy, that needs to be fulfilled by collecting assertions, to authenticate the user. Every other IdP runs an individual authentication factor. The main IdP controls to which IdP it directs the user to.
To allow a simple integration of this concept into an existing solution, standardized authentication protocols are used for the communication between the service provider and the main IdP, as well as for the main IdP and the different factor IdPs.

Since the authentication factors to be used for an authentication request are determined by the policy that is executed at the main IdP, the authentication factors to be used can be exchanged by changing the policy. With the policy making the decision on which authentication factors are to be used, the policy can take into account the environment available to the user. And therefore it can take full advantage of the possibilities that are available to the user.

The first publication for this kind of concept we found is from Kaji et al. [IDPPROXY]. It describes an authentication system, where an IdP proxy accepts authentication requests from SPs. To actually perform the authentication, the IdP proxy sends authentication requests to other IdP servers. The system uses assurance levels based on the probability of false acceptance. Each IdP that runs a specific authentication method has a defined probability of false acceptance. When a SP requests an authentication, the requested assurance level is mapped to a probability of false acceptance. The IdP proxy runs multiple authentications through several other IdP and sums up the probability of false acceptance of these IdPs until the requested assurance level is at least equivalent to the accumulated probability of false acceptance.

Multi Factor Authentication as a Service from Shah et al. [MFAAAS] describes an abstract system for authentication. It describes an architecture, that consists of two main parts. A server component (MFAS) and a client component (MFAP). The MFAS is a server application, that provides endpoints for authentication protocols. These endpoints are used by SPs to trigger authentication. As part of the authentication requests delivered to these endpoints the SPs can request an assurance level for the
authentication. The system maps this assurance level to an authentication policy. The MFAS knows multiple authentication modules, or servers. The policy defines the requests authentication modules, that needs to be executed to fulfill the authentication request. The MFAP is a client application, that is executed in the context of the user. MFAP discovers the available authentication factors to the user and collaborates with the MFAS to select and perform the authentication. The different authentication servers can have authentication modules running in the MFAP to perform an authentication.
6 Summary

Authentication is an important topic for modern online applications. Authentication is a complex issue. Many applications tend to externalize the authentication process itself to specialized online services, called identity providers (IdP). These IdPs provide an assertion with the user’s identity based on a certain level of assurance to the application.

Authentication standards like the NIST EAG and the ISO 29115 define authentication assurance levels, including features and requirements to fulfill these levels. Higher authentication levels require multifactor authentication. Currently three categories of authentication factors exist. These are possession (for example a secure token), knowledge (for example a password) and inherence (for example a fingerprint).

The technological development is very fast paced. This also includes new attack vectors against existing authentication methods and new technologies, that can be used to create new authentication methods. IdP need to adapt quickly to such changes, because potentially many applications are influenced by the current state of the IdP’s authentication methods. IdP’s again can utilize the concept of externalizing the authentication within their own system to create a highly flexible and adaptable agile authentication system. The main IdP executes different authentication methods which are connected by standardized authentication protocols. This design provides the possibility to exchange authentication methods quickly and without downtime.
## Document History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Author(s)</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.9</td>
<td>16.08.2016</td>
<td>Andreas Fitzek</td>
<td>Initial Version</td>
</tr>
<tr>
<td>1.0</td>
<td>17.08.2016</td>
<td>Arne Tauber</td>
<td>Review</td>
</tr>
</tbody>
</table>
Bibliography


[EIDAS] eIDAS-Verordnung EU Nr.910/2014


[BIOREVIEW] Biometric Authentication: A Review, Debnath Bhattacharyya et. al, 2009

[RFC2289] A One-Time Password System, N. Haller et.al., 1998

[RFC4226] HOTP: An HMAC-Based One-Time Password Algorithm, D. M’Raihi et. al, 2005

[RFC6238] TOTP: Time-Based One-Time Password Algorithm, D. M’Raihi et.al., 2011

[SAML2] Security Assertion Markup Language (SAML) V2.0 Technical Overview, OASIS


[RFC4422] Simple Authentication and Security Layer (SASL), A. Melnikov et.al., 2006

[FIDO2AF] FIDO UAF Architectural Overview, FIDO Alliance, 2014

[FIDO2U2F] Universal 2nd Factor (U2F) Overview, FIDO Alliance, 2015

[IDPPROXY] IdP proxy for combined authentication based on multiple IdPs, Kaji et al., 2010