A privacy-preserving authentication service using mobile devices

Mihai Togan[1,2] · Bogdan Chifor[1] · Ionuţ Florea[1] · George Gugulea[1]

[1]certSIGN, Bucharest, Romania
{mihai.togan | bogdan.chifor | ionut.florea | george.gugulea}@certsign.ro

[2]Military Technical Academy, Computer Science Dept., Bucharest, Romania
mihai.togan@mta.ro

Abstract

In this paper we discuss about authentication services using mobile devices, QR codes and attribute-based cryptography (ABC). The scope is to provide a secure device-to-service authentication once the human-to-device authorization is established. In this regard, we will leverage the ubiquity of mobile devices (such as smartphones) to act as authentication gateways between the user and the desired services. The paper proposes a privacy preserving attribute-based access control protocol to realize the authentication of the user to a restricted service using his mobile device. We employ the FIDO UAF authentication process that takes part between the local device and the online service. In addition, we extend the authentication mechanism by combining FIDO protocols and privacy-preserving attribute-based access control targeting trusted execution environment (TEE), thus allowing users to cryptographically prove distinct ID attributes instead of disclosing their complete identity.

1 Introduction

The smartphone market growth in recent years has been a technology success story. Chipset and increasingly powerful dedicated software platforms development has fuelled advances in phones design and ability beyond what could have been imagined just a few decades ago. The security of mobile devices deals with the same issues conventional computer security deals with: Confidentiality, Integrity, Authenticity, Non-Repudiation or Availability. Nowadays smartphones have enough processing power and third-party encryption applications could be easily available to end-users. The benefits of advanced security technologies such as cryptography, digital signatures and public key infrastructures are incontestable in the context of their use on desktop systems with applications and services such as data privacy, e-commerce, e-government, e-administration or online banking. Access to web sites from the mobile devices is a common practice therefore the same protection as when accessing the web resources from a computer shall be implemented regarding:

- Identity of the user
- Security of the connection
- Roles and access control

Mobile devices are updating the paradigm in information security. They are evolving to become the second factor in the access control scenarios (something that user has), as their ubiquity allows them to be considered as a de facto element that user always carries with him and that it is trusted in performing several security operations.

In this paper we propose the usage of the user's smartphone as the user authentication factor when the he tries to access a web portal from a desktop computer. The aim is to achieve the user authentication and authorization on a restricted area using the user's credentials stored on his mobile device.

Our contribution is twofold. Firstly, we show a general framework for a QR-based authentication service. The QR codes are used for the user's credentials transfer between his mobile device and the online service. The paper presents the involved entities and the main flow of the protocol that is used for authentication and access to the restricted web area. A PKI implementation of this general model is also described. Secondly, we propose a privacy-preserving authentication schema obtained by a combination of FIDO and U-Prove technologies. The paper proposes an extension of FIDO where attribute related information is stored in a custom way inside of FIDO extensions. Thus, FIDO may become a more flexible authentication framework which can be plugged with attribute-based protocols like U-Prove. The paper proposes a privacy preserving authentication scheme which uses FIDO as a bootstrap protocol for U-Prove authentication. Credentials transfer from the mobile device to the authentication server is realized by using QR codes. Two real-life use-cases are also proposed:

i) Students and professors access campus resources based on their attributes (registered student or professor).

ii) Access to age restricted content, e.g. rated movies, age restricted on-line shopping (alcohol, tobacco, etc.) presenting only reliable information regarding their age.

In both situations the user is using his mobile device to present his attributes. The verifiers require only a specific identity attribute in order to grant the individual access to a resource.

The rest of the paper is organised as follows. In section 2 we describe the QR-based authentication general framework. The involved protocol uses the user's mobile phone as an authentication factor for QR codes. A PKI implementation of the protocol is also presented. Section 3 briefly describes U-Prove and FIDO authentication technologies and proposes a combination of the two in order to create a privacy-preserving authentication protocol. Two use-case scenarios are presented in section 4. Finally, section 5 outlines our conclusions

2 QR-based Authentication

In this section we describe a type of user authentication service, which is based on the usage of the QR codes and the user's mobile phone. The involved entities are the user's mobile phone, user desktop computer (desktop), the Service Provider and the QR Authenticator. The aim is to complete the authentication and authorization for the user on the web application supplied by Service Provider. During the authentication process, the user's credentials stored in his smartphone are used. A temporary identity transfer is realized from the user's mobile phone to his desktop by using QR codes. Mainly, the Service Provider sends a temporary identifier to the user desktop in the form of a QR code, the smartphone scans it and after processing it sends an authentication data to the QR authentication server. Within this process the mobile and the desktop (or any other device) are coupled and associated as belonging to the same user. The user wanting to access a particular resource must be authorized by the Service Provider. The latter must receive the proper attributes (credentials or security assertions) from the QR Authenticator to authorize the user access.

The authentication flow is depicted in Fig. 1 and consists in the following steps:

1. The user opens the web portal in the browser from his desktop and starts an authentication process.
2. The Service Provider generates a random token associated with the current user session and sends a QR generation request to the QR Authenticator module for this session token. The Service Provider must not disclose any user session related information in order to mitigate vector attacks like session hijacking. The Service Provider request may contain a policy which describes the requested attributes in order for the user to be authorized.
3. The QR Authenticator verifies that the QR generation request is from a trusted entity, generates a QR code. It contains a nonce, the received session token and the expiration time. The QR code is saved in an internal database of QR Authenticator. The QR Authenticator sends the generated code to the Service Provider.
4. The Service Provider generates a QR code image using the information received from the QR Authenticator and sends it to the user browser.
5. The user opens the mobile application and scans the QR code image displayed in the browser. The mobile application decodes the QR code image and starts an authentication process with the QR Authenticator.
6. The QR Authenticator verifies the correctness of the received QR code and expiration time. The requested user access policy is sent to the Service Provider.
7. The Service Provider verifies the access policy and authorizes the user.
8. The Service Provider client-side component polls periodically the server in order to find the authorization status.

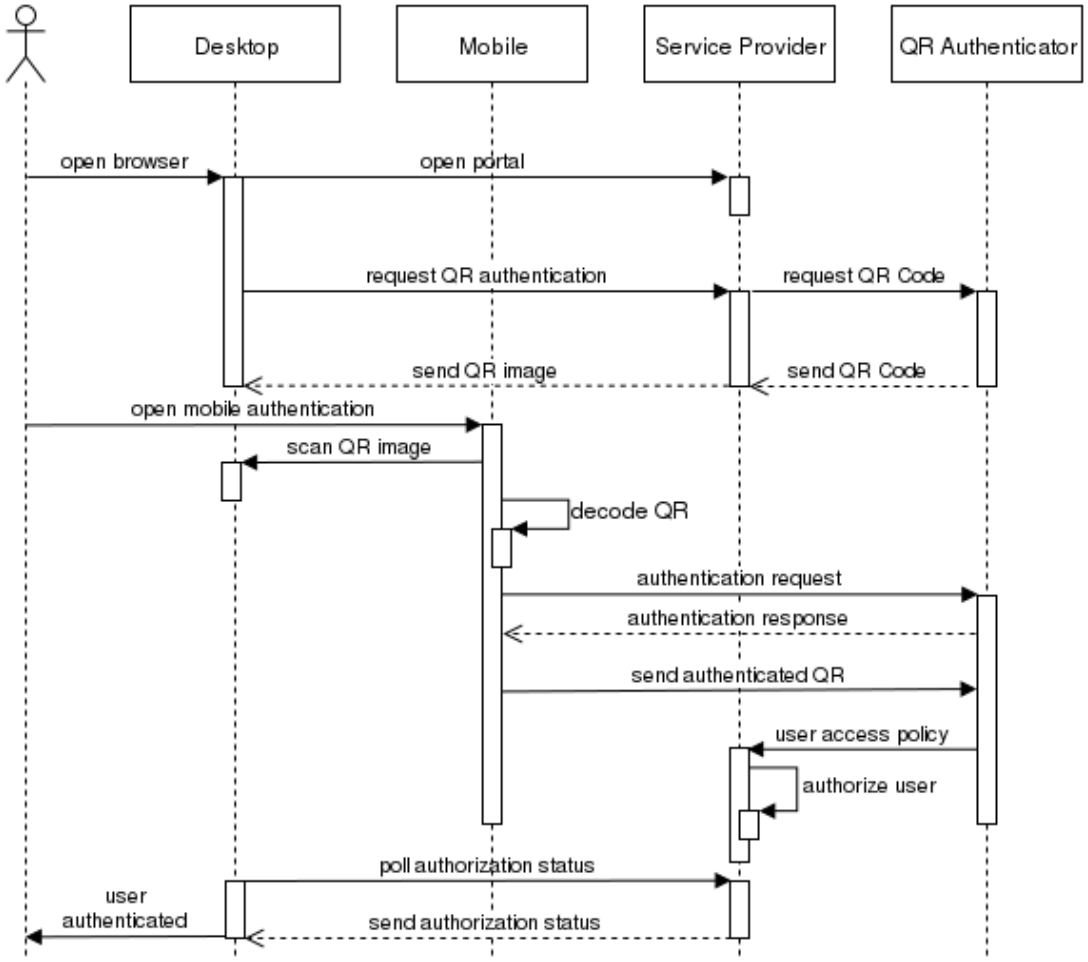

Fig. 1: QR-based authentication flow sequence (from [GFR+16])

The authentication process depicted in Fig. 1 is generic. The mobile application could authenticate to the QR server by means of PKI mechanisms, mutual TLS, FIDO protocols, attribute-based cryptographic protocols (like U-Prove [PaZa13]), or other methods. The mobile application may send the QR code to the QR Authenticator server through an authenticated and secure channel. A trust relation is established between the Service Provider and the QR Authenticator by using a shared secret. All the messages exchanged between the two server entities may contain a HMAC signature (HMAC-SHA512) or the two entities may use a mutual TLS communication.

The authentication of the user could be achieved with privacy-preserving if the Service Provider and QR Authenticator are located within separated trust domains. In this case, the QR Authenticator must be also a trusted service for the user.

2.1 QR-based Authentication Using PKI

In this section we describe a PKI implementation of the general QR-based authentication model described in the previous section. The usage of PKI to authenticate the user to QR Authenticator involves that the identity of the user needs to be provided by using digital certificates that are stored on the user smartphone. The device provides secure mechanisms for protecting the private key: secure key storage (Android KeyStore, hardware secure element) and password protected access. This allows the implementation of two factor authentication mechanism when the user needs to access the digital certificate:

- The smartphone (something the user has)

- The password to access the certificate from the smartphone (something the user knows)

When the user accesses the web site from the desktop, the mobile phone storing his digital certificate and private key is used. He must also have a digital signature application. The digital signatures and certificate validation are required to complete the authentication process carried by the user's smartphone and QR Authenticator. Step 5 from the user authentication process presented above unfolds with the following operations:

i) The user signs the QR code content using his private key and digital certificate stored on the mobile phone. A standard structure (like [Hous09]), or a custom one, could be used to envelope the signature and the content.

ii) The phone sends the signed content and his certificate to the QR Authenticator server using a separated connection.

iii) The server verifies the digital signature, the content and the digital certificate of the user. If the verification succeeds, the user is granted an access token which will be sent to the Service Provider.

The Service Provider allows the user to access the web resource using the rest of the flow presented in the QR-based authentication sequence. A Certificate authority issuing and managing the lifecycle of the digital certificates will be required in this implementation.

3 Privacy-preserving Authentication

To create a privacy preserving authentication protocol we analyse two authentication technologies, U-Prove [Paqu13] and FIDO [BaHH14], and propose a combination of the two. U-Prove offers privacy by disclosing only the required attributes of the users during the authentication process while FIDO has an important impact on user-experience as it employs a no password policy.

3.1 U-Prove

In this section we give a short introduction of U-Prove technology. It will be later used to extend the FIDO authentication capabilities. U-Prove is a cryptographic protocol, open-sourced by Microsoft, which provides the user's privacy by disclosing only the required attributes while interacting with an online verifier [PaZa13]. U-Prove architecture is divided in three separate entities: the user (also known as prover), the issuer and the verifier. In this paper, the user module is implemented as an Android application module. The main U-Prove artefact is an attribute container called token, which is signed by the issuer. The token has a public key and an associated private key which can be extended by means of a U-Prove device. The U-Prove device must be a trusted computing platform, but due to smartphone manufacturer restrictions regarding the execution of third-party code in the TEE. We chose to implement the U-Prove device logic in C programming language in order to have a portable solution. The U-Prove device software module is interfaced with the rest of the application by using Open-TEE [MDNA15] [OpTe16] and the Global Platform specifications [GpTe16].

U-Prove protocol provides unlinkability and untraceability, even if the issuer and the verifier collude, by means of blind signatures and zero-knowledge verification protocols. Concerning the U-Prove token structure, it is divided in two separate sections: Token Information (TI) and Prover Information (PI). The TI section is encoded by the issuer and disclosed during the presentation protocol, holding some application specific information like validity period or other metadata.

The U-Prove issuing protocol engages the user and the issuer which exchange three cryptographic messages. After the issuance protocol is completed the user generates the token. The structure of the issuance protocol messages along with the precomputation tasks executed on the user side are described in [PaZa13].

The U-Prove presentation process consists in a cryptographic protocol between the user and the verifier, during which the first one discloses a series of application specific attributes by means of a zero-knowledge proof protocol. The presentation has the purpose of proving the disclosed attributes validity and the user's private key ownership, which, used in conjunction with a verifier issued random challenge (cryptographic nonce) prevents replay attacks. The prover sends to the verifier a set of disclosed attributes and a presentation proof obtained by applying the token private key to the non-disclosed attributes and other data structures.

The main sub-components of the U-Prove presentation protocol are the proof generation and the proof verification. The proof generation sub-protocol consists in a series of four messages exchanged between the user and the device, ending with the user proof generation. The generated proof is then used in the other sub-protocol which consists in two messages exchanged between the user and the verifier. The structure of the proof generation messages is explained in [PaZa13].

Regarding the implementation, the issuer and the verifier modules are server-side components, implemented as web-application and interfaced with the smartphone modules through REST API. The U-Prove artefacts are serialized by following the Microsoft U-Prove WS-Trust Profile [Paqu11][BCP+16]. For both the server-side and client-side components the Microsoft Java implementation of U-Prove was used and deployed in a Tomcat container (on the server side) and in a Android application (on the user side). The deployment of the U-Prove architecture is depicted in Fig. 2. We have used the architecture presented in [BCD+14] as model for the deployment scenario.

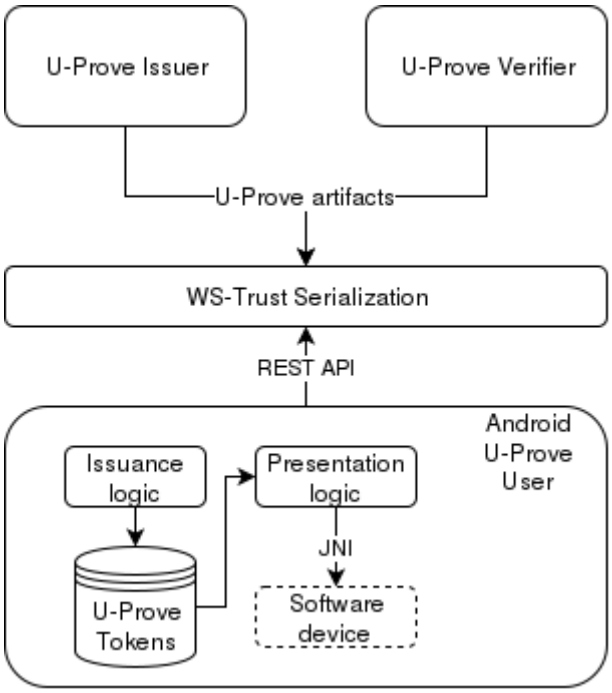

Fig. 2: U-Prove architecture modules

The U-Prove attributes are stored as blob in the Android application database by the Issuer module and retrieved by the Presentation module when necessary. The communication

between the Presentation module and the software device module is realized by means of a Java Native Interface (JNI).

3.2 FIDO

FIDO is a passwordless authentication protocol which can be used as first-factor (UAF) or as second-factor (U2F) element in the authentication process. The FIDO authentication gains traction because it is supported by important Internet and hardware companies which are part of the FIDO Alliance.

The main FIDO entities are the FIDO server, the FIDO client and the FIDO authenticator which is usually a trusted hardware device. In our work the FIDO authenticator was implemented as a software module, part of an Android application. The messages that are exchanged between the client and the server are divided in the following processes: registration, authentication and deregistration. During the registration process, an RSA key pair is generated, and the public-key is uploaded on the server-side along with a username. This is used in the authentication process where the user unlocks the authenticator private key and executes a FIDO challenge-response protocol. For unlocking the authenticator private key we used an Android smartphone equipped with a fingerprint reader. The FIDO messages structure and the communication flow are explained in [BaHH14]. They include also extensions that allow customization of the protocol.

3.3 FIDO Attribute-based Authentication

Even tough FIDO and U-Prove can be used for authentication and authorization, their combination has both advantages and drawbacks. The main advantage of using FIDO in combination with U-Prove is an improved security layer on the server side, where users can be authenticated or authorized based on their attributes. The most important drawback is losing two important U-Prove security features: untraceability and unlinkability. FIDO is not an accountless authentication protocol, thus every user action takes place in a session which is linked with a FIDO username.
Another challenge of integrating the two protocols is the number of messages required by U-Prove versus the simple challenge-response structure of FIDO.

In this section we propose an extended version of FIDO where attribute related information is stored in FIDO extensions. Thus FIDO becomes a flexible authentication framework which can be plugged with multiple cryptographic protocols. In our scenario the U-Prove token issuance takes place outside of the FIDO context and is employed only during the authentication process. When the user tries to access an unauthorized web resource, the FIDO server replies with an *AuthenticationRequest* message as described in [BaHH14]. The FIDO server transmits to the client the required U-Prove attributes through a FIDO extension as described in the following data structures.

```
Dictionary AuthenticationRequest {
        required OperationHeader header;
        required ServerChallenge challenge;
        Transaction[ ] transaction;
        required Policy policy;
}
Dictionary Policy {
```

```
        required MatchCriteria[ ][ ] accepted;
        MatchCriteria disallowed;
}
Dictionary Extension {
        required DOMString id;              /* Bind to 'U-Prove - attribute' */
        required DOMString data;            /* Required attribute encoded as base64 */
        required boolean fail_if_unknown;   /* Bind to true */
}
```

After the required attributes reach the software authenticator module through FIDO ASM, the U-Prove proof is generated and sent back to the server serialized in a FIDO extension which is encapsulated in the FIDO *AuthenticatorSignAssertion* structure.

```
Dictionary AuthenticatorSignAssertion {
        required DOMString assertionScheme;
        required DOMString assertion;
        Extension[ ] exts;      /*Serialized U-Prove proof*/
}
```

The latter structure is embedded in the FIDO *AuthenticationResponse* dictionary which is processed by the server. The attribute based FIDO authentication is figured out in Fig. 3.

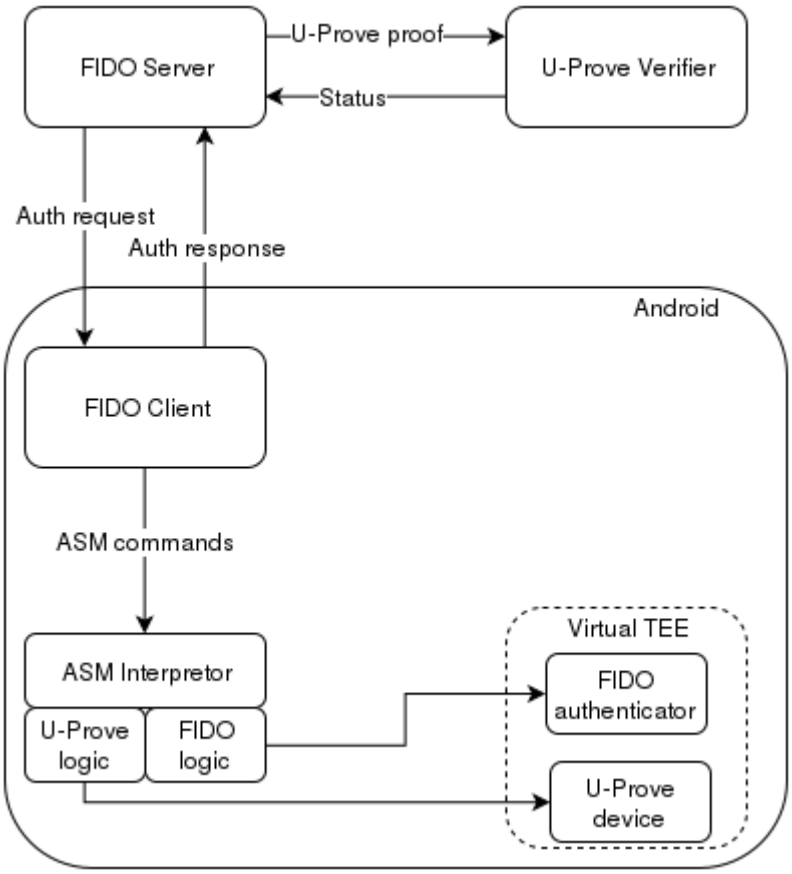

Fig. 3: Attribute based FIDO authentication process

In this implementation U-Prove does not handle replay attack mitigation, because this feature is already managed by FIDO (by signing the server generated challenge).

3.4 FIDO Authentication with Privacy-preserving

The U-Prove extensions over FIDO presented in the previous chapter do not employ the unlinkability and untraceability properties. The main advantage of using U-Prove over raw attributes in the previous context is the possibility to verify the attributes validity by executing the U-Prove presentation protocol. Another advantage for the attribute extended FIDO is abstracting the U-Prove proof generation process by using FIDO ASM commands. Therefore in an implementation similar to ours, the FIDO client does not have to contain any attribute processing logic.

To overcome the security drawbacks of the previous authentication scheme, we propose a combination of FIDO and U-Prove in order to obtain a privacy-preserving protocol. Thus FIDO becomes a bootstrap protocol for U-Prove authentication. For this implementation the FIDO and U-Prove logic are separated and used in a sequential manner. In this security scheme, employing FIDO brings a standardized user-to-device and device-to-service authentication while providing user-experience for unlocking the FIDO private key (by using biometrics or other authenticators). The usage of U-Prove after the FIDO authentication, protects the user‘s actions against linking with his real-world identity and assures the web service that the client is an authorized person (by having a registered account or other relevant attributes like proper age).

In this scenario the user does not trust the FIDO server, which can trace the accessed resources and link the FIDO account with the user's real-world identity.

After the user creates an account through the FIDO registration protocol, he authenticates to the server using the FIDO authentication protocol as detailed in [BaHH14]. The particularity of this scheme is the fact that FIDO server contains a U-Prove issuer and verifier implementation which handles session authentication tokens. After the FIDO *AuthenticationResponse* is verified successfully by the server, the FIDO client is issued a U-Prove token which states that it is authenticated. The session token has encoded in the TI field the session expiration time. This timestamp is expressed in hours, thus the user's session is protected by a k-anonymity scheme (it is assumed that multiple users login in an hour and all the users will have the same session expiration timestamp).

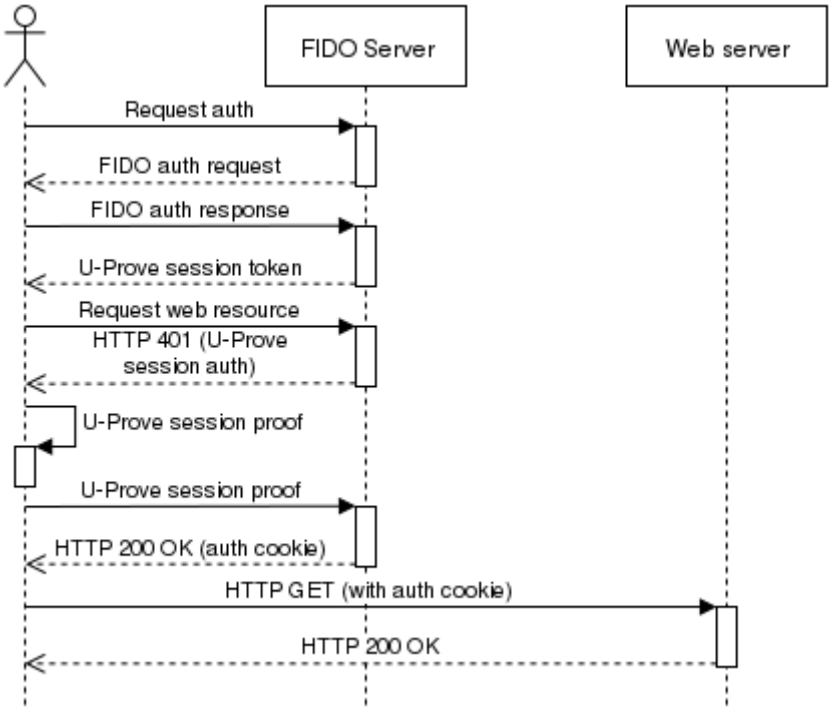

Fig. 4: Privacy-preserving FIDO authentication process

When the user tries to access a protected web resource, he engages in U-Prove presentation protocol with the web server. During this protocol the user discloses the fact that is a legitimate FIDO authenticated user and the session expiration time. Thus the web server cannot make a connection between the current HTTP request and the user's FIDO account. If the accessed resource requires disclosing other attributes, a U-Prove presentation protocol takes place. This privacy-preserving FIDO authentication scheme is presented in Fig. 4.

In the authentication sequence presented so far in this chapter, the user employs his smartphone to access a protected resource: both the client-side FIDO and U-Prove software stacks are part of an Android application. To consider the scenario where a user employs his desktop or another device to access a protected resource, we used QR authentication combined with privacy-preserving FIDO in order to transfer the trust from the smartphone to the other device. After the FIDO authentication takes place on the smartphone, the user is being issued a session token. When trying to access the protected resource from his desktop, the user is shown a QR code which contains HTTP session information and a server generated nonce. After scanning, the QR encoded data is signed by the smartphone U-Prove stack in the session token presentation protocol. After the session token attributes are verified, the desktop web client is granted access as described in section 2 of this paper. Thus the FIDO username used in the initial authentication cannot be linked with the resources accessed from the desktop device.

4 Use-cases

An important real-life use-case for the authentication methods proposed in this paper is the access of age restricted content: rated movies or age restricted on-line shopping (alcohol, tobacco etc.). From the security point of view this scenario requires the disclosure of relevant information regarding age, while preserving a person's anonymity.

If the desktop is used to access an age restricted web resource then the user's smartphone, which runs the security stack described in this paper, is used for authentication and authorization.

For this security scenario we employ the QR code authentication combined with U-Prove. The QR scanning is used to transfer the trust from the user's smartphone to his desktop. When a restricted web resource is accessed, a QR code containing web session and service provider information is displayed. After this step the mobile application uses the session information in a HTTP header when engages in a U-Prove presentation protocol with the service provider for disclosing the age related information. If the user's attribute is validated then the web session is mark as authorized and the user can access the restricted resource. For similar use cases which require an account, the privacy preserving FIDO method could be used.

Another common use-case which could employ the authentication methods described in this paper is accessing the campus resources by registered professors or students, based on their attributes. For this scenario U-Prove is used for managing the user's attributes and QR is used to transfer the trust from the user's smartphone to other devices. For this security scenario the user is being issued a U-Prove token which states that he is registered to the professor or student on a university. The token issuance is realized through the authentication stack described in this paper. When the user tries to access a restricted campus resource, he uses his smartphone to execute a U-Prove presentation protocol in order to disclose the fact that is a legitimate student or professor. If attribute verification is successful, the user is granted access to the resource, which could be an e-learning platform or another university service. If another device than the smartphone is used, then the QR code authentication transfer is used in the same way as the age restricted scenario.

5 Conclusion

This work addressed the authentication services using the mobile devices as authentication factor. In the first part of the paper we described a generic QR-based authentication framework. The involved entities and the main flow of the protocol have been presented. The framework is based on the user’s identity stored on his smartphone (as the user’s authentication factor) and a dedicated QR Authenticator component (as the user’s identity verifier). An identity transfer has been realized from the user’s smartphone to his desktop by using QR codes. The identity of the user is intended to be known only to the QR Authenticator but not to Service Provider. In this case, the model may provide also the user privacy-preserving if the QR Authenticator is a trusted party for the user and it not reveals the user’s identity to the Service Provider. A PKI implementation for the protocol was proposed in the paper.

In the second part of the paper we proposed an extension of the FIDO protocol in such a way to be combined with the attribute-based protocols like U-Prove. The main idea was to achieve the privacy-preserving for a FIDO already authenticated user. Firstly, we introduced the usage, in a custom way, of the FIDO extensions to carries attributes related information. In this way, FIDO becomes a more flexible authentication framework which can be plugged with multiple cryptographic protocols. Taking into consideration that FIDO security scheme does not take advantage of U-Prove unlinkability and untraceability properties, we proposed a privacy preserving authentication scheme which uses FIDO as a bootstrap protocol for U-Prove authentication. The FIDO and U-Prove protocols create a flexible security framework where both the user-to-device and the device-to-service authentication and authorization are provided. The usage of FIDO has a great impact on the user-experience because no passwords are employed. The usage of privacy-preserving attribute protocol ensures the user anonymity.

Taking into consideration these aspects the authentication methods presented in this paper could enable the development of multiple security scenarios which will be easily adopted by end-users.

The main direction for our future work would be implementing the U-Prove device module and the FIDO authenticator component in a custom TEE equipped hardware, in order to overcome the impossibility of running third-party code in a smartphone. Regarding the software stack developed so far, more testing would be required along with the development of an abstraction layer which could help integrating it into third-party applications. In order to integrate some authentication methods presented in this paper in a broader context, we will consider adapting them in order to comply with the specifications of FI-WARE security generic enabler [FIWA12]

Acknowledgements

This work has been partially funded by the European Commission in part of the ReCRED project (Horizon H2020 Framework Programme of the European Union under GA number 653417).

References

[BaHH14] Balfanz, Dirk and Hill, Brad and Hodges, Jeff, „FIDO UAF Protocol Specification v1.0", FIDO Alliance Proposed Standard, December 2014.

[BCD+14] Bichsel, Patrik et al., „Architecture for Attribute-based Credential Technologies - Final Version", ABC4Trust Project, Deliverable D2.2, 2014.

[BCP+16] Bianchi, Giuseppe et al., „Specification and Initial Design of the ABAC Infrastructure", ReCRED Project, Deliverable D5.1, 2016.

[FIWA12] FI-WARE Consortium, „FI-WARE GE Open Specifications (Security Chapter)", FI-WARE Project, Deliverable D8.1.1.b, 2012.

[GFR+16] Gugulea, George et al., „Description of DCA protocols and technology support", ReCRED Project, Deliverable D3.1, 2016.

[GpTe16] Global Platform TEE specifications, http://globalplatform.org/specificationsdevice.asp.

[Hous09] Housley, Russell, „Cryptographic Message Syntax (CMS)", RFC 5652 Internet Standard, 2009.

[MDNA15] McGillion, Brian and Dettenborn, Tanel and Nyman, Thomas and Asokan, N., „Open-TEE - An Open Virtual Trusted Execution Environment", In the 14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2015, Helsinki, Finland, August 20-22, 2015.

[OpTe16] Open-TEE Project, https://open-tee.github.io/.

[Paqu11] Paquin, Christian, „U-Prove WS-Trust Profile V1.0", 2011.

[Paqu13] Christian Paquin, „U-Prove Technology Overview V1.1 (Revision 2)", Microsoft, April 2013.

[PaZa13] Paquin, Christian and Zaverucha, Greg, „U-Prove Cryptographic Specification V1.1 (Revision 3)", December 2013.

Index

Authentication, mobile authentication, QR based authentication, FIDO, U-Prove, attributes based authentication, privacy preserving authentication.