ReCRED

From Real-world Identities to Privacy-preserving and Attribute-based CREDentials for Device-centric Access Control

Makes your digital life **safe** and definitely **easy**!

www.recred.eu

European Commission | Horizon 2020 European Union funding for Research & Innovation

FIDO in a nutshell

The password overloading problem

The password overloading is an already well-known problem that emerged outside IT professional people. As the mobile devices number increased so much in the last years, the number of people that have access to Internet increased as well. For most of the people, the Internet is represented by the web browser; but the reality is that it consists of a distributed collection of services such that every person that possess a smart-phone device has access to one or more services like text chat, e-mail, music, video, news or Internet banking, and I enumerated just a few of them. Although some of the services are free and do not require the user to authenticate, most of them made the user account mandatory. The world of the user accounts is dominated by the most simple authentication mechanism: user and password. As the number of accounts for one individual user increased, the number of passwords that he has to remember also increased. As the user tried to solve his "I don't remember my password" problem, he found solutions that just changed the problem into a security one: either he simplified the passwords to easily remember them, or repeated the same password with different services, or tried to write it in a "secure" place, being it a paper or a password manager, or all of them, he just exposed his accounts to hackers and made them less secure.

What is basically a secure mechanism, the password, became a very insecure one. If we take it individually, a password still offers a good security, if it respects some rules: it is long enough (e.g. 20 characters), it is complex enough (no dictionary words, alternate characters, etc.), is not too aged (e.g. less than 1 year), it is not reused, etc.; but if we take a step behind and take a global view to the world of the passwords, we encounter

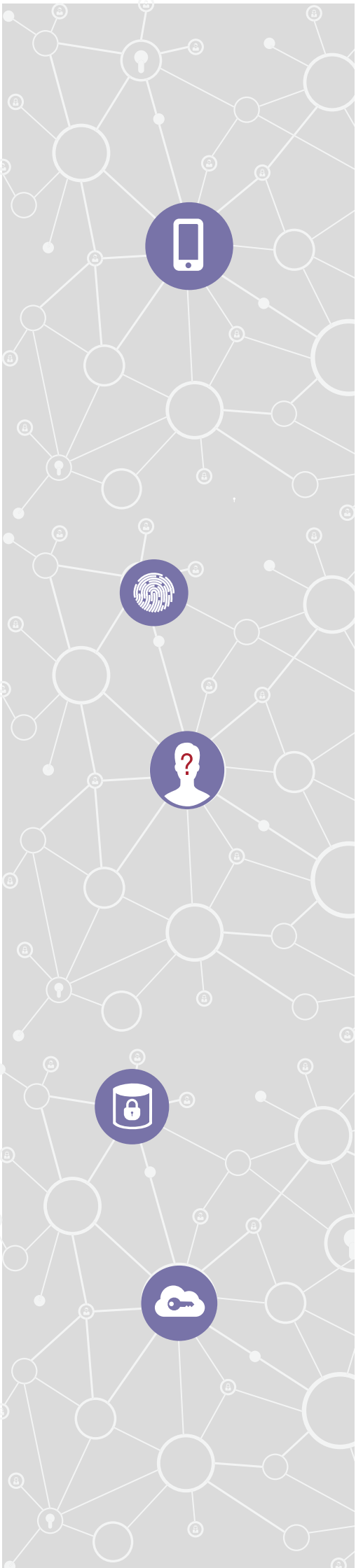
the problems that I mentioned above and it became a serious problem.

PKI came with a solution to the weak security passwords. Even if the math behind PKI: asymmetric cryptography is proven to be very secure, much more secure than passwords, the management of the keys still pose some problems and there are some well known attacks on the Certification Authorities that resulted in compromising the CA and in the end issuing malicious digital certificates.

The mechanism of using a pair of keys: one that is public (not secret) and is used for encryption and one that is private (only known by the user) and is used for decryption is proven to be very secure. The algorithms behind are available since 70', e.g. RSA algorithm, but they were never adopted by the mass of the end user, due to its lack of usability; securing the private key and the fact the user identity is bound to the public key in a digital certificate that is public for everyone to know, are just a few of the problems that PKI is facing.

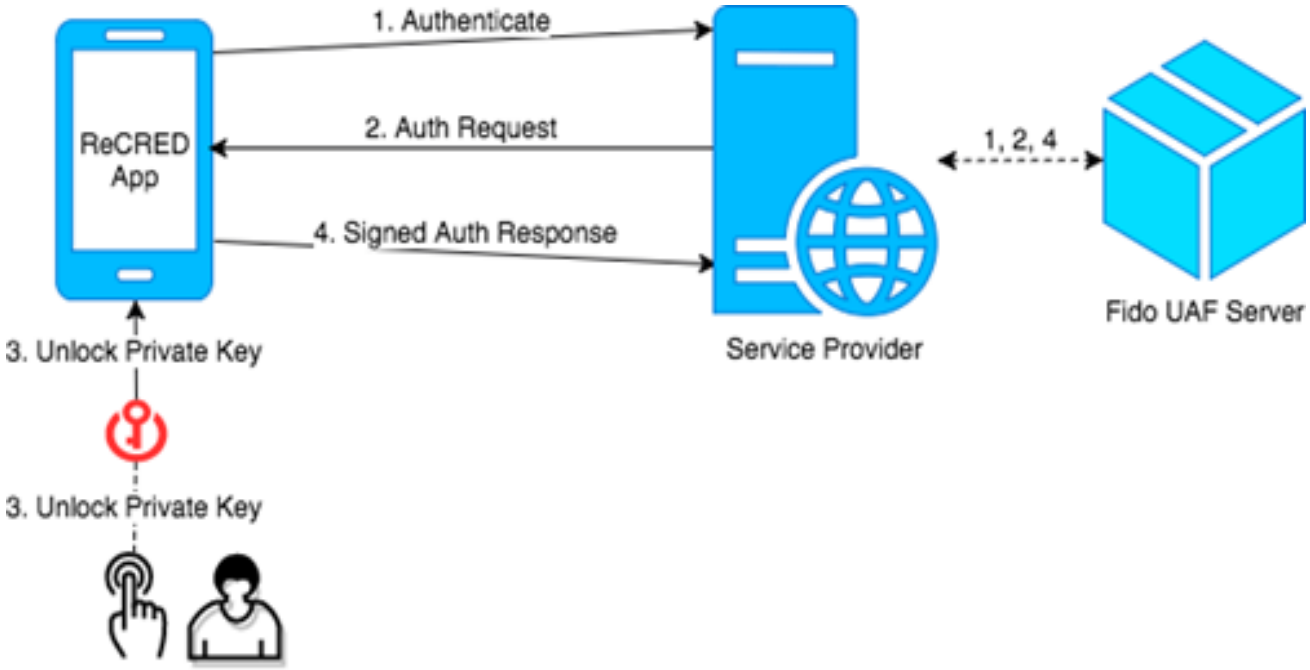
The FIDO Alliance came up with a solution to the existing passwords problem, by taking advantage of the asymmetric cryptography, but with a different approach than PKI. The FIDO protocols use the same mechanism of challenge/response authentication, but do not bound the public key to a digital certificate. The protocols do not intend to establish user identity and that process is left to external mechanisms; in fact, FIDO protocol solves the privacy problem by issuing similar FIDO Authenticator tokens to a big number of users (e.g. 100.000), such that even if the FIDO Authenticator token manufacturer is able to track the customer that bought it, will not be able to differentiate which one is the user that he tracks.

FIDO consists of two authentication protocols, called FIDO U2F and FIDO UAF, representing "Universal Second Factor" (U2F) and "Universal Authentication Framework" (UAF) respectively, first with the role of enhancing the authentication of the end-user to services by adding a second factor authentication to existing user accounts without changing the first authentication mechanism (e.g. username and password), and the other protocol is in fact a framework through which services can build up an authentication mechanism using the FIDO UAF protocol.



The FIDO UAF components

There are four main components in a basic FIDO UAF architecture:



- Service Provider - that contains the specific business services that the user wants to access
- Authentication service - that is a specialized FIDO UAF server (like the one we implemented in ReCRED)
- user-device - usually represented by a mobile phone or an Internet browser that is FIDO UAF aware
- User - which is required to unlock the private key and give his consensus to authenticate.

On the user device, there are three components that FIDO UAF stack is comprised of, plus the application itself:

- The user-agent that can be supplied through a native application of through a browser that understands FIDO protocols.
- FIDO UAF Client that checks if the user agent is authorized (for native applications checking if the developer certificate is registered with the FIDO UAF Server, or for web applications checking if the URL is registered with the FIDO UAF Server), determines if the users' UAF Authenticators match the UAF Server policy and relaying information received from the UAF Server to the UAF ASM.
- FIDO UAF Authenticator that is the FIDO interface to the private key storage
- Authentication Specific Module (ASM) that is an interface between the user-agent and the FIDO UAF Authenticator.

On the client side, the roles of each component are very well defined and the communication with the server side takes place through the user-agent only. On the

server side, the Service Provider does not implement and neither interprets FIDO UAF protocol messages, but forwards them to the FIDO UAF Server; therefore, it acts as a proxy.

How FIDO works

Both FIDO protocols, UAF and U2F, are challenge-response based authentication mechanisms and consist of server sending a challenge to the client, and the client signing the challenge and sending it back to the server. This is a very simplified overview of the protocols; below we will detail the UAF protocol.

1. In the first step the user initiates a request to the Service Provider by using the user-agent. The user-agent can be a web browser, or in our ReCRED project, an Android application called ReCRED App. The request is a normal HTTPs request to a user-owned resource that resides on the Service Provider. From the authentication point of view, the Service Provider is a relying party (RP), because it relies on an external component to realize the authentication. The RP just acts as a proxy requesting a new authentication challenge from the server and routing it back to the user agent.

2. FIDO UAF Server generates a challenge that is a random unique value and adds it to the authentication request, alongside the authenticator policy, and sends it back to the RP. The authenticator policy is comprised of some specific attributes that the FIDO Authenticator must have in order to be accepted by the server. If the wrong authenticator is selected, the signature from the authentication response will not be validated and the authentication will fail. The RP sends the authentication request to the FIDO Client through the user-agent. The FIDO Client interrogates each available authenticator through the ASM and selects the one that is appropriate to the policy that the FIDO UAF Server requires (e.g. one that requires user fingerprint authorization).

3. The authentication request is sent to the ASM that calls the Sign function on the selected Authenticator. Upon data-to-be-signed is received (data-to-be-signed is comprised by the UAF Server identity "appID" who generated the request, the identity of the user agent who requested authentication "facetID", communication channel information "channelBinding" – hash of the UAF Server TLS certificate and the challenge generated by the UAF Server) , the authenticator triggers the user input to unlock the private key. Therefore, a pop-up is presented and the user is asked to authenticate to the secure credential storage, using one of the registered methods. Usually, on Android platforms, the user enrolls one or more of his fingerprints, a PIN/password or a swipe pattern that are used to unlock the phone (the user authenticates to the phone). The same mechanism and the same credentials are used by the FIDO UAF client stack to unlock the secure storage that holds the private key used to sign the data received through the ASM. The user must use the credential specified (e.g. put his fingerprint on the fingerprint sensor) and the data-to-be-signed is signed inside the Authenticator.

4. The signature is returned to the FIDO UAF Client through the ASM. The ASM constructs the Authentication Response structure that will further arrive to the user-agent. The user-agent (ReCRED App) will send this structure to the Service Provider, as a response to the Authentication Request. The Service Provider will send the authentication response to the FIDO UAF Server that will verify the signature validity and the policy correctness. Upon a successful verification, the result is sent back to the Service Provider that returns it to the user agent.

FIDO UAF Operations

We described before the authentication operation which presumes that the user is initially enrolled with the Service Provider and subsequently with the FIDO UAF Server. There are three operations that the FIDO UAF Server defines:

- Registration Operation that has the purpose of enrolling the end-user and his user-device with a specific Service Provider.
- Authentication Operation that we described in the previous paragraph.
- Deregistration Operation that has the purpose of deleting a registered user and a registered device from the Service Provider database and also delete the key material from the user device as well.

FIDO UAF Security Measures

In order to mitigate threats that target the user, the relying party computing environment and also the supply chain (including the vendors of FIDO components), the protocol uses a multitude of security measures, the most important being the following:

FIDO Metadata Service (FIDO Authenticator "PKI" Architecture)

The entire "PKI" architecture is based on the concept of a Metadata Statement which is a "certificate" describing the technical characteristics of the FIDO Authenticator

and containing the vendor attestation certificates. The workflow around this FIDO Authenticator metadata statement is the following: a FIDO Authenticator vendor produces the metadata statement describing the characteristics of an authenticator, which is submitted to the FIDO Alliance as part of the certification process that distributes it through the UAF Metadata Service. The FIDO relying party configures its' registration policy to allow only the users with UAF Authenticators that match certain characteristics to register. After the client processes and sends a registration response to the server (this response contains the manufacturer and model information of the authenticator "AAID"), a signature is made with the attestation private key, ensuring that the key is generated with a certified authenticator that meets the registration policy.

User Unllinkability

To protect the protocol conversation such that any two relying parties cannot link the conversation to one user, FIDO UAF uses unique authentication keys and authenticator class attestation. Unique authentication keys imply that an authentication key is specific and unique to the following tuple: FIDO authenticator, user, relying party. Authenticator class attestation implies that hardware based FIDO authenticators support the authenticator attestation using a shared attestation certificate and that each relying party receives regular updates of the trust store through the Metadata Service.

Signature Algorithms

The FIDO eco-system allows the usage of multiple signature algorithms and signature encoding formats as a failover mechanism in case a particular signature algorithm isn`t considered secure anymore. The signature algorithms and encoding formats are:

- RAW SECP256R1 ECDSA with SHA-256: An ECDSA signature on the NIST secp256r1 curve which must have raw R and S buffers, encoded in big-endian order;
- DER SECP256R1 ECDSA with SHA-256: DER encoded ECDSA signature on the NIST secp256r1 curve
- RAW RSA SSA PSS with SHA-256: RSASSA-PSS signature must have raw S buffers encoded in big-endian order
- DER RSA SSA PSS with SHA-256
- RAW SECP256K1 ECDSA with SHA-256
- DER SECP256K1 ECDSA with SHA-256


ReCRED


ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
UNIVERSITY OF PIRAEUS


EXUS


institute
iMdea
networks


cnit


Cyprus
University of
Technology


Telefónica
Telefónica I+D


THE
PRODUCTIZERS


certSIGN


wedia


verizon


upcom
Bringing business and IT together


Baker
McKenzie.