# A User-Centric Identity Management Framework based on the W3C Verifiable Credentials and the FIDO Universal Authentication Framework

Romain LABORDE, Arnaud OGLAZA, Samer WAZAN, François BARRERE, Abdelmalek BENZEKRI,
University Toulouse III Paul Sabatier
{Romain.Laborde, Arnaud.Oglaza, Samer-Ahmad.Wazan, Francois.Barrere, Abdelmalek.Benzekri}@irit.fr

David W. CHADWICK
University of Kent
d.w.chadwick@kent.ac.uk

Rémi VENANT
University of Le Mans
Remi.Venant@univ-lemans.fr

***Abstract*— We present a user-centric and decentralized digital identity system that allows anyone to easily benefit from an enriched digital identity made of multi-purpose and multi-origin attributes. It increases usability by the elimination of user passwords. It also makes this digital identity highly trustworthy both for the user (in terms of privacy and sovereignty) and the service provider who requires highly certified information about the user being enrolled to and/or authenticated on its services. We built our system based on the Universal Authentication Framework specified by the FIDO Alliance and the data model proposed by the W3C Verifiable Credentials WG. The whole system has been implemented in a banking scenario.**

***Keywords—Federated Identity Management, FIDO UAF, Verifiable credentials***

## I. Introduction

Registration and authentication processes on websites have been significantly simplified by federated identity management systems, which allow single sign-on. A central actor, the Identity Provider (IdP), centralizes all the identity attributes of each of its users and provides them with a single authentication process they can use to identify and authenticate to any service on the internet that is federated with the IdP. Based on web-based standard protocols (e.g., SAML [1], OpenID Connect [2]), this process replaces the tedious task of manually declaring an identity by registering at each service provider (SP), with an automatic exchange of identity information between the IdP and the SP. Identity federation also simplifies user authentication in a password based world, since the user only authenticates to the IdP, thereby reducing the number of credentials that need to be remembered.

However, today's federated identity management (FIM) systems have a significant structural weakness, namely, the placement of the IdP at the centre of the identity ecosystem. First, the trust model requires the IdP to trust the SP to preserve the privacy of the user's identity attributes that it is asserting, and the SP to trust that the IdP is the authoritative source of (all of) the user's identity attributes. Both of these trust requirements are unreasonable. No single IdP is the authoritative source of all a user's identity attributes, and users may want to present their identity attributes to SPs that IdPs do not fully trust. Secondly, the IdPs are the center of the identity eco-system, and issue short-lived identity assertions [1] or tokens [2] on-demand to trusted SPs. Consequently, they know which SPs the user is visiting and when, which allows them to track the user. In addition, besides violating the user's privacy, it also introduces a severe security vulnerability as the recent Facebook hack [3] highlighted. This allowed the attackers to access all the user's accounts at all the SP web sites that trusted Facebook as the user's IdP. Since IdPs, like Facebook, store a huge amount of information about lots of users, such attacks have a big impact. Finally, the assurance SPs can have on identity attributes is very low since attributes are mostly always self-asserted by the user when (s)he creates/updates his/her account at the IdP. This stifles the digitalisation of services and transactions. There are a few examples of IdPs that certify the integrity of users' attributes, such as universities, and France Connect. However, these identity federations are restricted to academia or French administration offices only (that brings us back to the first issue).

Our hypothesis is that 1) placing the user at the centre of the identity ecosystem rather than the IdP, and 2) splitting omnipotent IdPs into small and specialized Attribute Authorities (AA) or Issuers is both more secure and more privacy protecting, and consequently will become the predominant identity management architecture of the future. Our hypothesis is supported by the "plastic card model" (e.g., association membership cards, scuba diving accreditation cards, ID cards, etc), which allow the user to apply for plastic cards from multiple issuers, keep them in a wallet, and then present them to whomsoever they choose, when they choose. This model is already ubiquitous in the physical world, and we propose to use it in the electronic world. This architecture allows the user to:

1) choose which card issuers to enroll with,
2) choose which service providers to present his/her cards to,
3) not inform the card issuers prior to use of his/her cards,
4) choose which cards to use, from those accepted by the service provider,
5) update his/her personal details with the issuers and inform them to revoke one of his/her cards.

As a consequence, there is a need for a new FIM system that allows users to generate on-demand identities that contain only the necessary information by aggregating validated identity attributes from different attribute authorities. In

**Deleted: in this article**

**Deleted:**

**Deleted:** o

addition, attribute authorities should neither be able to control the disclosure of users' attributes, nor track them. Hence, our research follows the philosophy of the self-sovereign identity paradigm [4]. Although there is no consensus on the self-sovereign identity definition [5], W3C [6] explains that "in a self-sovereign system, users exist independently from services. In contrast, in a service-centric system, users are tightly bound to a particular service." As a consequence, in a self-sovereign identity system, the user must be central to the administration of its identity.

We present in this article a user-friendly identity system that places the user at the centre, using the new W3C Verifiable Credentials standard [7]. The architecture places the user at the center of the FIM eco-system rather than the IdP, and the trust model only requires the SP to trust the IdP. However, currently the W3C is only standardizing the data model for VCs and no protocols are being proposed.

We have built a VC eco-system based on an enhanced FIDO UAF protocol. Our identity system is highly secure and privacy protecting with the following properties. **Availability** is achieved because the user will be able to present her credentials to the SPs of her choosing at any time, and the SP should always be able to validate them. **Confidentiality** is respected because the credentials will only be visible to third parties (SPs) chosen by the user. **Integrity** since the SP will be able to validate the integrity of all credentials presented to it i.e. they are current, not modified, and all belong to the presenter. Finally, we followed the **Privacy-by-design** principle. In our system, the user should not have to present more credentials to an SP than are needed to access its resources (selective disclosure/data minimization/least privileges). She should consent to the release of her credentials (making our system compliant with the European GDPR), and she should not be trackable through the use of her credentials, except in cases of abuse. We implemented the whole system and proved its efficiency in an online banking registration scenario.

The rest of the article is structured as follows. Section II presents the related works. Section III describes the underlying technologies. In section IV, we describe our identity management system called Universal Authentication and Authorization Framework. We present the conceptual and implementation architectures. In section V, we illustrate its use in a bank scenario. Finally, section VI concludes and highlights future works.

## II. Related Works

The two main federated identity management protocols are SAMLv2 [1] and OpenID connect [2]. As used in the Shibboleth implementation, SAMLv2 is typically used for identity federations between organizations where users are employees of the IdP. This gives the IdP organisation complete control over the attributes disclosed to the SP. Also, as operated by the academic community, it requires strong trust relations between the IdPs and the SPs, who sign agreements in order to join the federation. OpenID connect fits more with the concept of user-centric federated identity management on the web because (theoretically) users can choose which attributes will be revealed to the SPs. In practice, however, users are given a 'take it or leave it' menu by the IdP and have to agree to the release of all of the attributes that the SP requests. Usually, the SP retrieves the user's attributes directly from the IdP via a backchannel, allowing the IdP to track users. Finally, there exist proprietary protocols that are interoperable with openID connect (e.g., Facebook connect, google+ sign-in, etc). However, in addition to the openID connect weaknesses, these protocols constrain the user to one central IdP per session (e.g., Facebook or Google). Solid (https://github.com/solid/) lead by Tim Berners-Lee is proposing to build a decentralized social application based on Linked Data principles where authentication is done by WebID-TLS. Although this approach complies with the self-sovereign principles, Solid cannot provide certified identity attributes.

Many new approaches propose to build a FIM system following the self-sovereign objective on top of blockchains [8,9], such as ShoCard, Sovrin, UPort, OneName, BitID, ID.me or IDchainZ. A first strategy consists in digitizing paper-based credentials and publishing a hash on a public distributed ledger. For instance, in ShoCard (https://shocard.com/), users scan their existing paper-based trusted credentials (e.g., their passport) using the device's camera and extract attributes by applying an optical character recognition technique. A signed hash of this data, called ShoCardID, is stored in Bitcoin transactions. In a second step, this data has to be certified by an identity provider and the hash of the certificate is published in the Bitcoin ledger. This certification task is similar to the registration process in X.509 PKI systems. All the attributes are stored in the users' mobile devices. A ShoCard server can store symmetrically encrypted certifications (called envelopes) in case the user loses its device. A relying party can validate an identity after the user has provided the symmetric key and the envelope reference from the ShoCard server. This solution has several drawbacks. First, ShoCard has privacy weaknesses. The hash of the users' identity attributes is publicly available, and the ShoCard server may be able to associate a particular ShoCardID with requests made by relying parties and then technically track users similarly to SAML or OIDC IdPs. Secondly, Bitcoin transactions take on average 10 minutes to be mined and waiting for six additional blocks is recommended for the settlement of a transaction [8]. This implies users will have to wait at least one hour before being able to use their certified identities. This constraint limits the usage of ShoCard to scenarios requiring only predefined identities (where attributes are known in advance) since it is not possible to create identities in real time. Finally, the central role of the ShoCard server raises the question of the sustainability of users' identities if the company ceases to exist.

Sovrin (https://sovrin.org) has developed another strategy that relies on a permissioned-ledger based blockchain to store identity records. An identity record is a transaction on the Sovrin Ledger that describes a Sovrin Entity that may include Public Keys, Service Endpoints, Claim Definitions, Public Claims, and Proofs. Identity Records are Public Data. Only trusted nodes, called stewards that satisfy the Sovrin Foundation Agreement can add identity records to the permissioned-ledger. Specific software programs, called Agents, act on behalf of the entity to interact with other Agents or with the Sovrin Ledger. There are two types of Agent: Edge Agents that run on a local device (e.g., on a user's mobile device), and Cloud Agents that run remotely on a server or cloud hosting service. Credentials are exchanged between Agents using a proprietary P2P-based protocol while the format of the credentials aligns with the W3C Verifiable Credentials standard. This solution fits more with our objectives and using a blockchain as a root of trust is

**Deleted:** s

**Deleted:** all

**Deleted:** s

**Deleted:** I

**Deleted:** s

**Deleted:** IdP

**Deleted:** claims

interesting. However, the Sovrin approach suffers some drawbacks. First, users could run their Cloud agents on their own servers, but more likely, they will delegate this task to specialized intermediaries agencies [8]. As a consequence, a lot of user information will be available to the agencies. Moreover, the Sovrin protocol is proprietary. A better approach is to adopt standards around which an ecosystem can be developed. In the same vein, Sovrin is not compatible with other FIM technologies. Confining the system can impact the adoption of the solution.

Last but not least, none of the current blockchain-based FIM solutions have considered usability and user understanding of privacy implications [8,10] whilst all these approaches advocate the need for user control. Finally, as pointed out by Yael Grauer [11], "multiple startups in decentralized sovereign identity solutions are competing to be the key players in this space. Confining users within a particular blockchain contradicts the main idea of self-sovereign identity".

## III. The Main Technologies

### A. W3C Verifiable credentials

A verifiable credential (VC) is defined in [7] as a set of one or more tamper resistant claims made by an *issuer*, where each claim asserts a set of properties about a *subject*. The architecture of the VC model is shown in Figure 1.

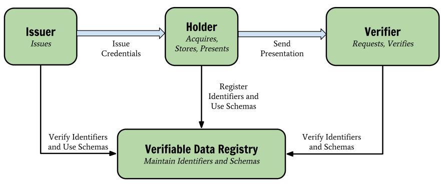

Issuer
Issues
Issue Credentials
Holder
Acquires, Stores, Presents
Send Presentation
Verifier
Requests, Verifies
Register Identifiers and Use Schemas
Verify Identifiers and Use Schemas
Verifiable Data Registry
Maintain Identifiers and Schemas
Verify Identifiers and Schemas

Fig. 1. W3C Verifiable Credentials Architecture [7]

The subject should create one or more globally unique identifiers mediated by a *verifiable data registry*. The *holder* asks the *issuer* to create a VC for one or more subjects by binding properties to identifiers. The holder is usually, but not always, the subject of the verifiable credentials they are holding, for example, a parent may hold a child's VCs. The issuer verifies the holder, its right to hold the subject's VC, the identifiers and the properties, and then issues the VC. The holder can store the issued VC in its repository for later use. Finally, holders combine one or more VCs together (i.e. attribute aggregation) to present *verifiable presentations* to verifiers. In this model, VC issuers don't know the identities of verifiers, which is a significant change from current FIM systems.

### B. FIDO Universal Authentication Framework

The FIDO Alliance has specified the Universal Authentication Framework (UAF) for password-less authentication [12] using asymmetric encryption on smart devices.

All UAF devices contain one or more FIDO Authenticators. An authenticator is a secure entity within the smart device that can create and store asymmetric key pairs and authenticate the user to access the keys via a specific method such as fingerprints, PIN, face recognition, etc. All FIDO authenticators must be certified and registered with the FIDO metadata service, which is managed by the FIDO alliance. They all contain an Attestation Private Key inside them that is used to certify the public keys they create. Attestation key pairs are generated by the device/authenticator manufacturer and are inserted into their authenticators. The FIDO Alliance recommends that the same attestation private key is inserted into around 100,000 authenticators to protect the user's privacy. In this way, there is no unique ID associated with the user (otherwise a unique attestation public key would become a globally unique correlating ID for the user). All FIDO servers store the attestation public key certificates to validate the integrity of the messages. They can retrieve the attestation public key certificates from the metadata service. Since all FIDO authenticators are uniquely identified by an Authenticator Attestation ID (AAID), the FIDO servers can limit the authenticator to be used by the user.

**Deleted:** choose

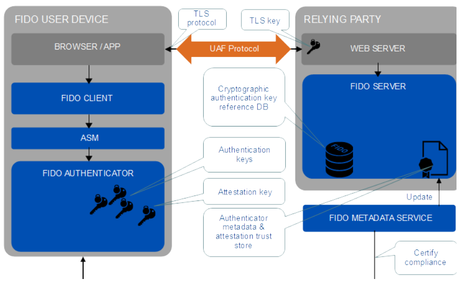


Fig. 2. FIDO UAF Architecture [9]

The FIDO user device creates a new asymmetric key pair for each web site the user authenticates to. UAF has adopted the Same Origin Policy (SOP) [13] to ensure signed data is only transferred between web pages of the same web site i.e. the pages have the same origin (same protocol, host and port). This helps in preventing cross-site scripting and similar attacks, as each private key is used for one site only.

When the user first contacts a web site (called Relying Party - RP) from her FIDO device, the site's FIDO server issues a registration request message to the FIDO client in the user's device, containing its authentication policy. This policy lists the FIDO authenticators that the RP accepts. The FIDO client asks the user to choose one authenticator that matches the policy. Then, the FIDO client calls the chosen authenticator via its Authenticator-Specific Module API (ASM). The user authenticates to the authenticator and it creates a new key pair dedicated to this site ($PrK^{Site}$, $PuK^{Site}$) where $PrK^{Site}$, $PuK^{Site}$ are the private and public keys respectively. The key pair is bound to the site's origin (i.e. name, port, and protocol) by the authenticator, in order to enforce the SOP. The authenticator returns the public key to the FIDO server (via the client) signed by its attestation private key $PrK^{Att}$ i.e. $sign_{PrKAtt}\{PuK^{Site}\}$ where $sign_{PrKi}\{data\}$ means *data* is signed by the private key i. The FIDO server can validate the signed registration response message using the trusted attestation public key and stores the user's public key in its database.

When the user logs in to the site again, the FIDO server sends an authentication request message, containing a challenge and its Authentication Policy, to the FIDO client. The client calls the authenticator that has registered with this

**Deleted:** code

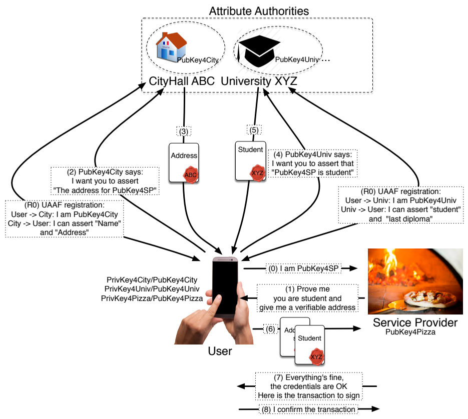
site to respond to the challenge. The authenticator checks the site's origin and only activates the private key $PrK^{Site}$ if it matches the stored origin. It then asks the user to authenticate via its supported method(s). Once the user has authenticated to it, the authenticator signs the challenge with the private key and returns the authentication response to the site via the client.

## IV. The Universal Authentication and Authorization Framework

On one hand, the W3C is only standardizing the data model for VCs and no protocols are being proposed. On the other hand, FIDO UAF only deals with authentication. In this section, we present our FIM framework called Universal Authentication and Authorization Framework that extends the FIDO UAF protocol to implement W3C Verifiable Credentials. We first introduce the conceptual model and then we describe the implementation.

### A. *The UAAF conceptual model*

Our FIM model places the user at the centre of the identity ecosystem rather than the IdP, and splits omnipotent IdPs into small and specialized Attribute Authorities (AA). Existing organizations already assert verifiable credentials in our daily life and thus they can play the role of an AA. For instance, Universities assert diplomas or student status, utility companies and the city hall know your name and address, the French scuba-diving federation FFESSM certifies diving degrees, etc. As a consequence, users can provide the SP with verifiable identities by aggregating identity attributes from these well-established authoritative entities. Based on this observation, we propose our UAAF model (Fig. 3).



Attribute Authorities
PubKey4City
PubKey4Univ
...
CityHall ABC
University XYZ
(3)
(5)
Address
ABC
Student
XYZ
(4) PubKey4Univ says: I want you to assert that "PubKey4SP is student"
(2) PubKey4City says: I want you to assert "The address for PubKey4SP"
(R0) UAAF registration: User -> City: I am PubKey4City City -> User: I can assert "Name" and "Address"
(R0) UAAF registration: User -> Univ: I am PubKey4Univ Univ -> User: I can assert "student" and "last diploma"
(0) I am PubKey4SP
PrivKey4City/PubKey4City
PrivKey4Univ/PubKey4Univ
PrivKey4Pizza/PubKey4Pizza
(1) Prove me you are student and give me a verifiable address
Service Provider
PubKey4Pizza
(6)
Student
XYZ
User
(7) Everything's fine, the credentials are OK Here is the transaction to sign
(8) I confirm the transaction

Fig. 3. Figure 1.The UAAF model

First, the user has to register with the AAs she has a relationship with (e.g. her city hall, her insurance company, her University, etc). We do not dictate the entire enrollment process because AAs already have their own enrollment procedure. The main goal of our UAAF enrollment is for the AA to link a FIDO public key with the user. In the most secure procedure, the user should physically present herself to the AA staff for them to validate her physical identity and her possession of the FIDO device. It is also possible to send a One Time Password (OTP) by regular mail to the user's home as is currently done for credit cards. Whatever the procedure, the user will then register her device using the FIDO UAF protocol. This consists in generating an asymmetric key pair and transmitting the public key to the AA (see section III.B). When this step is complete, the AA sends the user the list of attributes it can assert for her (see fig.3 step R0) through a new message that extends the UAF protocol.

Before making any transaction, the user should register its device on the SP website. When it receives a standard FIDO registration request, the FIDO authenticator of the user's device creates a new key pair for communicating with the SP (step 0 in fig.1) and sends it to the SP. When the user wants to make a transaction (e.g., buy a pizza), the SP replies to the user by presenting an authorization policy that indicates which attributes are needed for this transaction and the list of trusted AAs for each attribute (step 1). The SP should prepare its authorization policies beforehand for the various services it offers. Each protected URL can have a different policy. This allows fine-grained access control. Our authorization policy model covers both CNF and DNF policies to describe all possible policy combinations [14]. The UAAF module on the user's device looks in its database to see if the authorization policy can be matched by the attributes the user has already selected for asserting by her registered AAs. If no matching attributes are found, the transaction stops, and indicates to the user which attributes are missing. Otherwise, the user selects one AA from the SP's trusted AAs to issue a verifiable credential for each of the requested attributes. The UAAF client now performs a standard FIDO authentication exchange with the first AA's credential issuing web page, and then sends a new verifiable credential request message to the AA. The UAAF module asks the authenticator to sign both the AA's and SP's public keys with the attestation private key i.e. $Sign_{PrK^{Att}}\{PuK^{SP}, PuK^{AA}\}$, to prove that both keys belong to the same user (step 2). Otherwise it would be possible for one user to pass on the public key of a second user. We employed the FIDO ASM API extension features to make this new message compliant with FIDO UAF. In this way, the AA doesn't know anything about the SP the AA is visiting. It only knows the user has another public key. Finally, the AA returns the set of verifiable credentials to the user's UAAF module (step 3). The format of the credentials respects the W3C Verifiable Credential data model. This verifiable credential creation process is repeated until the user has verifiable credentials for all the attributes requested by the SP (e.g., steps 4 and 5). When she has retrieved all the verifiable credentials, the user returns them to the SP (step 6) in a verifiable presentation. This provides the attribute aggregation feature from multiple AAs. The SP can then validate the signature of the verifiable credentials and check if they match with the authorization policy. If so, the SP grants the user access to its protected resource. The SP can also ask the user to sign the transaction acceptance (step 7) for non-repudiation or user consent purposes. The user can sign this transaction using the private key created for this specific SP (step8).

### B. *Implementation*

We implemented the AA and SP servers by extending the UAF FIDO server provided by eBay [15] to be compliant with our authorization framework. Both AA and SP servers are built on the popular Spring framework and follow the Model/View/Controller software design pattern. We deployed our UAAF models with Spring Data, which provides a programming abstraction layer to access data in a unified way regardless of the actual data store. This allows us to distribute

**Deleted:** like it
**Deleted:** is
**Deleted:** pair of
**Deleted:** s
**Deleted:** all
**Deleted:** the
**Deleted:** that
**Deleted:** The
**Deleted:** is
**Deleted:** ing
**Deleted:** s
**Deleted:** and name
**Deleted:** for
**Deleted:** ing
**Deleted:** s
**Deleted:** attributes
**Deleted:** have
**Deleted:** for
**Deleted:** an
**Deleted:** is
**Deleted:** a
**Deleted:** people

two releases. The "integration" release allows administrators to integrate the servers (AA and SP) with their existing information system. Administrators can specify their own data model and link it to their databases. They also need to create a web site and complete the configuration files. The drawback of the "integration release" is it requires Spring skills making the deployment somewhat difficult if they are not present in the company. The "plug-and-play" release focuses on making the deployment fast and easy. It provides a data model and only requires administrators to create the web site and complete the configuration files. However, this version does not allow the administrator to integrate the servers with their existing databases. Finally, the FIDO UAAF client is available on Android (version 6.0 and higher) and iOS (version 10.3 and higher).

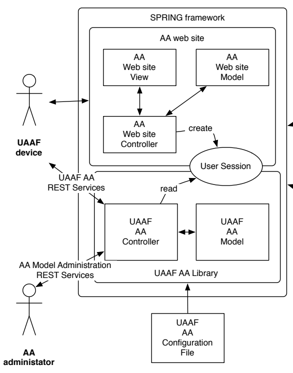


Fig. 4. The UAAF AA Architecture

Creating an AA web site entails deploying seven REST services, creating a configuration file, and populating a database that defines which users are entitled to which attributes. Three REST services implement the standard FIDO UAF protocol:

- *regRequest,* issues the UAF Registration Request message to the client (FIDO device) when asked for it;
- *authnRequest*, issues the UAF Authentication Request message to the client when asked for it;
- *uafResponse*, receives the client's response to the above messages; either UAF Registration Response containing the attested public key from the client, or UAF Authentication Response signed by the client's private key.

Four  Three? REST services implement our FIDO authorization service:

- *attrList* returns the set of attributes the AA is prepared to assert for the user (step R0);
- *userSelectedAttrList* receives the list of attributes the user consents to be used as credentials (step R0);
- *credentialsToCertify* and *credentials* receive the credential request (steps 2 and 4) and return the credentials to the client (steps 3 and 5).

Our UAAF AA library doesn't manage the first step of the initial registration, which is user identification and authentication. This gives the AA flexibility to choose the most suitable procedure, such as an OTP or Activation Code, etc. Once the user is authenticated, the AA web site controller must create a transient object to store the user session, which links the different TLS connections and informs the UAAF AA controller that the userID has been authenticated.

Administration of the database is possible through AA Administration REST services (add/modify/remove users, add/modify/remove attributes, etc). Finally, the AA administrator can customize the library in a FIDO metadata configuration file where (s)he can set: the list of allowed Authenticators' AAIDs, the trusted facet list (in FIDO, a trusted facet list states the different identities of a single logical application; it is used by the client to apply the SOP), the attestation keys, and the path to the private/public key to sign credentials.

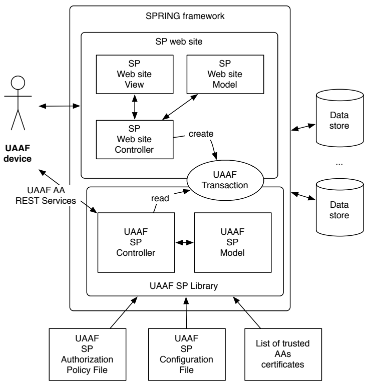


Fig. 5. The UAAF SP Architecture

The architecture of the SP is similar to that of the AA. Our FIDO UAAF SP library supplies the three FIDO REST services for authentication described above, as well as two more REST services for authorization:

- *UAAFPolicyRequest* allows the FIDO UAAF client to get the authorization policy related to the current resource (step 1);
- *signedUAAFResponse* receives the signed credentials (step 6).

The SP web site communicates with the UAAF SP library through a transient object of type *UAAF-Transaction* that the SP web site controller must create using our SP UAAF API. This object links the different TLS sessions and allows the SP web site to give all the information about the current transaction/resource access to the UAAF AA library.

Finally, the administrator of the SP web site should configure into the SP configuration file the metadata for FIDO authentication (allowed AAIDs, trusted facet list, attestation certificates), and FIDO authorization (the path to the trusted AAs' certificates and the path to the SP Authorization Policy File).

The SP's Authorization Policy File contains a set of policies. Each policy tells the user which alternative sets of attributes from which AAs are required to gain access to a particular resource/undertake a chosen transaction. This allows the site to support *the least privileges* principle. Each

**Deleted:** s

**Commented [MOU1]:** Why is this 4 REST services. Isnt it 3 since the last two is a rest request and response policy is defined by a name. The SP web site controller decides which policy to send to the client depending on the transaction (or resource) chosen by the user. When the UAAF SP controller gets the policy name, it sends the referenced policy to the user.

Finally, we implemented our UAAF client on both Android (version 6 and above) and iOS (10.1 and above). The software architecture on Android implements the FIDO ASM as a separate APK-packaged application, as suggested by the FIDO UAF documentation. The ASM protects the private keys in the Android KeyStore system. On iOS, the UAAF and the ASM are bundled together in the same app due to inter-process communication limitations and private keys are stored in the device's secure enclave. Except for these technical differences, the core of the UAAF client is identical on both platforms. We extended the ASM to certify that the authenticator manages two keys. This action is required to limit the scope of use of the credentials signed by the AA; otherwise, two users could collude and share an AA's issued credentials. The ASM API [16] is a JSON-formatted request/response protocol to communicate with the authenticator. The FIDO UAF standard defines six predefined request types (“GetInfo”, “Register”, “Authenticate”, “Deregister”, “GetRegistrations” and “OpenSettings”). However, this standard protocol is also extensible because the ASMRequest type contains a field dedicated to extensions. As a consequence, we extended the “Authenticate” ASMRequest message by using this extension field to inform the authenticator to sign two keys in order to prove it manages both keys (step 2 in figure 3).

## V. Applying our FIM system to online banking

We applied our user-centric FIM system to the bank context. We interviewed bank experts at informatique Banque Populaire to understand the current process and experience of users when they try to open a new bank account online. It consists of:

1) Manually filling in some fields to indicate their name, surname, address, etc.
2) Uploading pdf documents to prove the truthfulness of the information previously provided;
3) Waiting for 4 or 5 days to get their account opened because a human operator has to validate the uploaded documents.

This process has several drawbacks. First, it harms the users' experience. The whole process is painful, from manually filling in fields, to uploading pdf files using a mobile device, and then waiting 4 or 5 days for the response. Secondly, users can provide fake documents and can fool the manual validation. As a consequence, we developed a prototype to show how our system can help the bank to fight against identity fraud, improve the trust of their clients, speed up the entire process and facilitate the development of new businesses and services.

We started by analyzing the different documents needed to open a new bank account. The current process requires document-based credentials. However, these documents contain much information (i.e. many attributes), not all of it being relevant. Thus, we had to interview our partner to determine which of the attributes included in the documents are really needed. During this analysis, we could also determine the existing AAs that are already trusted by the bank (see fig. 6).

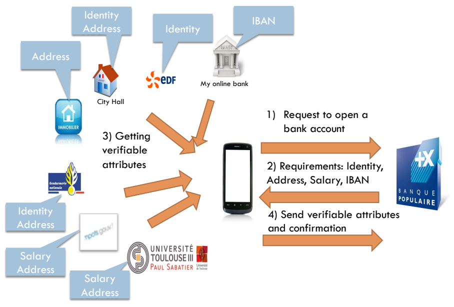
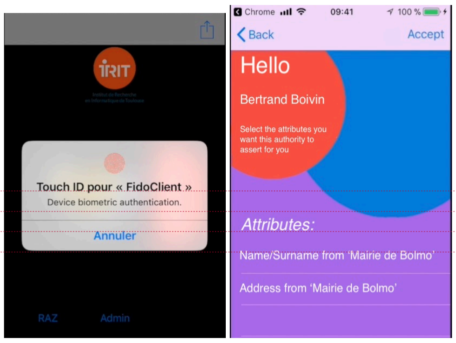

Identity
Address
Identity
IBAN
Address
City Hall
My online bank
1) Request to open a bank account
3) Getting verifiable attributes
2) Requirements: Identity, Address, Salary, IBAN
4) Send verifiable attributes and confirmation
BANQUE POPULAIRE
Identity
Address
Salary
Address
UNIVERSITÉ TOULOUSE III PAUL SABATIER
Salary
Address

Fig. 6. The ‘opening a new bank account’ use-case

In our new process, the user starts by registering her device on each AA she already knows. For instance, she can go to her city hall and register her device using an OTP provided by a city official. Fig. 7(a) is the screenshot when the user is asked to authenticate using the TouchID to create the FIDO key pair for the city hall. When the city hall receives the public key, it sends the list of assertable attributes back to the user (see fig. 7(b)).


IRIT
Touch ID pour « FidoClient »
Device biometric authentication.
Annuler
RAZ
Admin
Chrome 09:41 100 %
Back
Accept
Hello
Bertrand Boivin
Select the attributes you want this authority to assert for you
Attributes:
Name/Surname from 'Mairie de Bolmo'
Address from 'Mairie de Bolmo'

(a) (b)

Fig. 7. Screenshots of the enrollment process on the AA

Now let’s consider the user has registered her device at her city hall (here Mairie de Bolmo), an electricity provider (here EDF), her current bank (here my online bank), and her real estate agency (here ImmobCity). She connects to the new bank web site to create a new bank account. After asking some legal questions, the new bank website will start the UAAF process. The UAAF client of the user then creates a key pair for the new bank website and sends the public key to the web site. Then, the new bank web site sends its authorization policy to the user (fig. 8(a)) where it asks for three verifiable credentials: a proof of identity (name/surname), a proof of address and the IBAN number of the current user’s bank account. The real process actually requires more attributes but we simplified the use-case for clarity. For each requested attribute, the user can select the AA that will issue the related verifiable credential (fig. 8(b)). Once the AAs are selected, the user will generate signed verifiable credential requests (fig. 9(a)). For each request, the user will be asked to authenticate using the TouchID so that the UAAF client can use the private

**Deleted:** the

**Deleted:** employee

**Deleted:** could

**Deleted:** . Currently, the

**Deleted:** create

**Deleted:** in

**Deleted:** /

**Deleted:** on

**Deleted:** are painful tasks

**Deleted:** until

**Deleted:** /

**Commented [MOU2]:** You need to say which ISSUERS are trusted

**Commented [MOU3]:** Is this a good idea. I think you should include ALL the details

**Deleted:** exhibit

**Deleted:** s

**Deleted:** d

key for the respective AA to sign the messages. When all the verifiable credentials have been retrieved, they are transmitted to the new bank web site. In our use-case, the new bank website shows a transaction confirmation message (fig. 9(b)) after having verified the credentials. If accepted, the user signs the transaction confirmation using her new bank's private key.

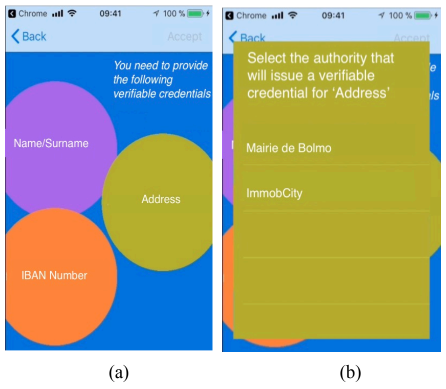

Back
Accept
You need to provide the following verifiable credentials
Name/Surname
Address
IBAN Number
(a)
Back
Accept
Select the authority that will issue a verifiable credential for 'Address'
Mairie de Bolmo
ImmobCity
(b)

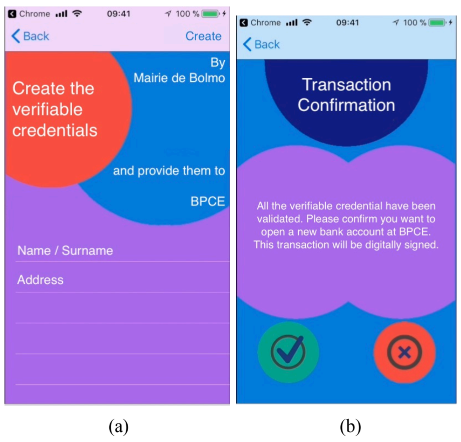
Fig. 8. Screenshots of the SP's authorization policy

## VI. Conclusion and Future Works

We presented in this article the Universal Authentication and Authorization Framework. This user-centric and decentralized identity management solution gives control of digital identities back to the users. In parallel, it reduces the power of the IdPs by splitting monolithic IdP entities into small AAs, and preventing them, by design, to track users. Our system is also more resilient. Indeed, users can obtain the same credentials from multiple AAs. This way, the impact of an AA closing its service is limited. Finally, our system is more secure for SPs that require assurance of credentials and for users by removing the need for passwords. The whole framework has been implemented (Spring servers for AAs and SPs, iOS and Android for the users' devices) and applied to opening a new bank account.

We acknowledge that our system radically transforms the current digital FIM ecosystem and the associated business model where identities are sold by IdPs. Nevertheless, our FIM ecosystem mirrors the trust model which is currently deployed by organizations in the physical document-based identity model. As a consequence, organizations can use their existing business relations to build their digital identity federations.

Our future work will focus on the acceptance of our new ecosystem. We have obtained feedback from industrial people that have ask for help in managing the transition from their currently deployed FIM technologies to our vision. As a consequence, we will work on integrating OIDC and SAML to our system as a transitional solution while preserving the privacy of the users and the trust that SPs have in existing attribute providers.



All the verifiable credential have been validated. Please confirm you want to open a new bank account at BPCE. This transaction will be digitally signed.

Fig. 9. Screenshots of the verifiable credentials creation and transaction confirmation

## Acknowledgment

We would like to thank M. Collas from informatique Banque Populaire for providing us with the scenario and allowing us to access the documentation. We also thank the members of the OCSSIMORE Association (https://www.ocssimore.net/) for their feedback.

## References

[1] OASIS. "Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V2.0", OASIS Standard, 15 March 2005

[2] N. Sakimura et al. "Final: OpenID Connect Core 1.0 incorporating errata set 1." 8 Nov 2014. Available: http://openid.net/specs/openid-connect-core-1_0.html

[3] facebook, Security Update, https://newsroom.fb.com/news/2018/09/security-update/, Sept 2018.

[4] Mühle, A., Grüner, A., Gayvoronskaya, T., & Meinel, C. (2018). A survey on essential components of a self-sovereign identity. Computer Science Review, 30, 80-86.

[5] https://www.coindesk.com/path-self-sovereign-identity/

[6] https://w3c.github.io/webpayments-ig/VCTF/charter/faq.html#self-sovereign

[7] W3C, "Verifiable Credentials Data Model 1.0 - Expressing verifiable information on the Web", February 2019

[8] Dunphy, P., & Petitcolas, F. A. (2018). A first look at identity management schemes on the blockchain. IEEE Security & Privacy, 16(4), 20-29.

[9] Orman, H. (2018). Blockchain: The emperor's new PKI?. IEEE Internet Computing, 22(2), 23-28.

[10] Elsden, C., Manohar, A., Briggs, J., Harding, M., Speed, C., & Vines, J. (2018, April). Making sense of blockchain applications: A typology for HCI. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (p. 458). ACM.

[11] Grauer Y (2018), https://breakermag.com/a-critical-look-at-sovereign-identity-startups/ (accessed 6/7/2019)

[12] FIDO Alliance. "FIDO UAF Architectural Overview." FIDO Alliance Proposed Standard. 8 December 2014

[13] A. Barth, "The Web Origin Concept." RFC 6454. Dec 2011.

[14] B. Moore, E. Ellesson, J. Strassner, and A. Westerinen, "Policy Core Information Model -- Version 1 Specification," RFC3060, Feb. 2001.

[15] [12] The eBay UAF FIDO implementation is available from https://github.com/eBay/UAF [Accessed: 17 Feb 2017]

[16] The ASM API is available from https://fidoalliance.org/specs/fido-uaf-v1.0-ps-20141208/fido-uaf-asm-api-v1.0-ps-20141208.html

**Deleted:** their

**Deleted:** y

**Deleted:** the

**Deleted:** by

**Deleted:** the IdPs

**Deleted:** s

**Deleted:** level

**Deleted:** n

**Deleted:** a

**Deleted:** got

**Deleted:** ing

**Deleted:** the

**Deleted:** that are currently deployed