Improved Identity Management with Verifiable Credentials and FIDO

David W Chadwick, Romain Laborde, Arnaud Oglaza, Remi Venant, Samer Wazan, and Manreet Nijjar

Abstract

We describe how FIDO and W3C VCs can overcome the problems of existing identity management systems. We describe our conceptual model and architecture, and the protocol we used by extending FIDO's UAF in order to provide both strong authentication and strong authorization. We built a pilot implementation for U.K. NHS patients to validate our implementation. Patients were able to use a mobile phone with a fingerprint reader to access restricted NHS sites in order to make and cancel appointments and order repeat prescription drugs. Our initial user trials with 10 U.K. NHS patients found the system to be easy to use, and fingerprints to be preferable to using usernames and passwords for authentication.

Introduction

Today's federated identity management (FIM) systems have a number of weaknesses. Architecturally they are flawed by placing the identity provider (IdP) at the center of the identity ecosystem. IdPs are privacy-invasive, tracking every user login. They are also honeypots as the recent Facebook hack [1] showed, by providing the attacker with login to every service provider (SP) where the user had an account. Furthermore, redirection by the SP to the IdP allows malicious SPs to phish users' login usernames and passwords. IdPs typically issue short-lived bearer identity assertions [2] or tokens [3], which can be stolen and used by an attacker. The FIM trust model is also flawed as it 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 (usually all of) the user's identity attributes, which no IdP typically is. Consequently, IdPs are not willing or able to release all the user attributes that SPs require for fine-grained authorization. "Insufficient attribute release by IdPs is considered by user communities as the major problem today in the eduGAIN space" [4]. This necessitates the pulling of user identity attributes from other attribute authorities, and to solve this "attribute aggregation" problem, some IdPs assign a persistent globally unique correlating identifier to each user [5, 6]. Finally, IdPs release all the user's attributes at login time, before the service is chosen, thereby providing maximum rather than least privileges.

The World Wide Web Consortium (W3C) Verifiable Credentials (VC) Data Model [3] overcomes the above weaknesses. First, it places the user at the center of the identity ecosystem (Fig. 1). Now the IdPs issue cryptographically protected VCs to the user, who stores them and releases them to SPs when needed. VCs are not bearer tokens and thus cannot be stolen. No redirection exists, so phishing and IdP tracking are negated. IdPs are not aware of which SP the user is contacting. Second, the trust model only requires the SP to trust the IdPs. Thus, the user can present multiple VCs from multiple IdPs to the SP as required, enabling attribute aggregation and fine-grained least privileges access. However, the W3C is only standardizing the VC data model, and no VC protocols are currently being proposed.

In contrast, the Fast Identity Online (FIDO) Alliance specified two authentication frameworks and protocols: the Universal Authentication Framework (UAF) for password-less authentication from smart devices [7], and the Universal Second Factor protocol (U2F) for two-factor authentication using a small hardware token to accompany a non-FIDO smart device having a FIDO-compliant web browser. Both operated on the same underlying principle of using asymmetric encryption for authentication, and both have now been combined into the W3C Web Authentication Recommendation (FIDO2) [2]. While FIDO2 provides strong authentication, it does not provide authorization, so we combined the FIDO UAF and W3C VC architectures and extended the UAF protocol to provide strong authentication and authorization. This generic architecture and protocol was then implemented and validated in a prototype application for the NHS, and tested at a U.K. hospital with 10 patients.

W3C Verifiable Credentials

A VC is defined in [3] as a set of one or more tamper-resistant claims made by an issuer, where each claim asserts a set of properties (viz: identity attributes) about a *subject*. The architectural components of the VC model are shown in Fig. 1.

The subject creates one or more globally unique identifiers, which are stored in a *verifiable data registry*. The *holder*, who in most cases is the subject,[1] asks the issuer (cf. IdP) to create a VC for a subject by binding her identity attributes to an identifier. The issuer verifies the holder, its right to hold the subject's VC, the identifier, and the attributes (perhaps via some out-of-band means), and then issues the VC. The holder stores the

[1] In special cases someone other than the subject may hold the VC. For example, a parent may hold the VC of a child. A relative may hold a prescription VC of a patient.

Digital Object Identifier:
10.1109/MCOMSTD.001.1900020

David W Chadwick (corresponding author) is with the University of Kent; Romain Laborde and Arnaud Oglaza are with the University Paul Sabatier; Remi Venant is with Le Mans University; Samer Wazan is with Zayed University; Manreet Nijjar is with Truu Ltd.

2471-2825/20/$25.00 © 2020 IEEE

VC in its digital wallet for subsequent use. Holders present *verifiable presentations* to verifiers by combining one or more VCs together (i.e., attribute aggregation). Note that holders do not reveal the verifiers to the issuers.

FIDO UAF/FIDO2

All FIDO UAF devices contain one or more FIDO authenticators — a secure entity within the device that can create and store asymmetric key pairs and can authenticate the user before allowing access to the keys. User authentication is via one or more methods supported by the device including fingerprints, face recognition, PIN, and so on. When a user first purchases a FIDO-ready smart device, they must register with it, which requires them to authenticate to the device's authenticator using the method(s) it supports. All FIDO UAF authenticators contain an attestation private key, inserted by the manufacturer, which is used to create trust in the public keys they create. The FIDO Alliance recommends that the same attestation private key is inserted into around 100,000 authenticators in order to protect the user's privacy (so there is no unique key associated with the user which would become a globally unique correlating ID). All FIDO servers store the corresponding attestation public key certificates, obtained from the metadata service (Fig. 2). FIDO2 has broadened the attestation trust model to allow for self-attestation, no attestation, and CA-based attestation.

FIDO2 creates a new asymmetric key pair for each web site to which the user authenticates. FIDO has adopted the Same Origin Policy (SOP) to ensure that the user's client will only transfer signed data between web pages of the same web site. When the user first contacts a web site via TLS, the FIDO server issues a registration request containing its authentication policy. This lists the FIDO authenticators it trusts. The FIDO client asks the user for consent, then calls the authenticator to create a new key pair for this site. The authenticator returns the public key to the FIDO server signed by its attestation private key. Finally, the FIDO server validates the response and stores the user's public key in its database.

When the user wishes to login again, her client makes a TLS connection to the web site, and the FIDO server sends an authentication request message containing a challenge and its authentication policy. The client calls its authenticator to respond to the challenge. Once the user has authenticated to the authenticator and consented, it signs the challenge with the private key for this site and returns the authentication response. The site now knows that it is the same user that originally registered with it, but not who this user is. This is the reason we have extended the FIDO UAF architecture to provide identification using W3C VCs.

The FIDO Universal Authentication and Authorization Framework

The preconditions for our FIDO Universal Authentication and Authorization Framework (UAAF) (Fig. 3) are that each credential issuer (IdP) is FIDO-enabled, has a key pair for signing credentials (PrKAA, PuKAA), and the public key PuKAA is distributed securely out-of-band to the SPs (web sites) who

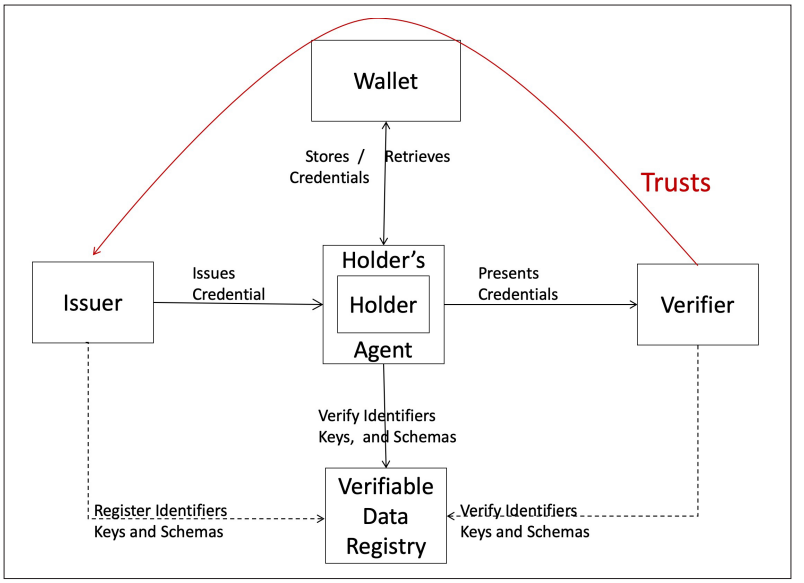

Figure 1. Verifiable credentials architecture.

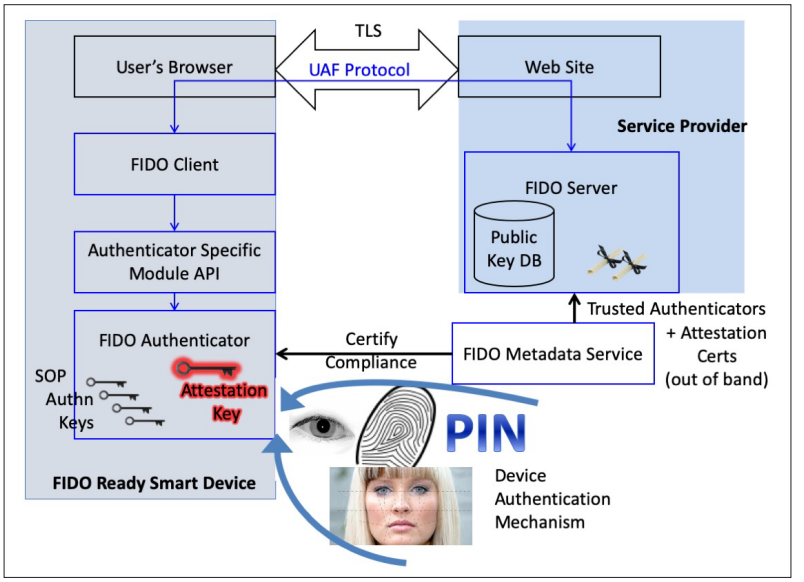

Figure 2. The FIDO UAF architecture.

are FIDO-enabled and who trust this IdP to assert its particular identity attribute(s). Each user has a FIDO-ready smart device and has registered with it.

User Registration

IdPs typically know the identities of their users and which identity attributes to assign to them. There are no privacy issues involved in an IdP issuing VCs to a user whose identity it already knows, and it has no knowledge where the user may subsequently present these VCs. To register for VCs, the user may physically present herself to the IdP while in possession of her FIDO device. Alternatively, she may already have a username and password for the IdP's web site, or a one-time password (OTP) could be sent by post to her home address. Either way, the user establishes a TLS connection to the IdP's web site, which causes it to issue a registration request, and the authenticator of the user's device to create a new key pair (PrK^{UAA}, PuK^{UAA}) for the IdP. The attested public key, $sign_{PrK^{Att}}\{PuK^{UAA}\}$, is passed to the

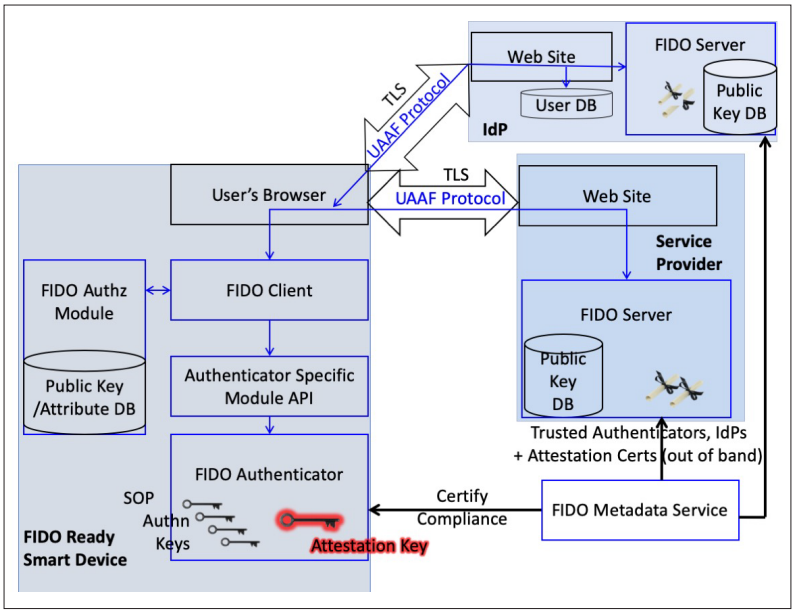

FIGURE 3. The FIDO UAAF architecture.

FIDO client, which passes it to the IdP's web site and simultaneously to our new FIDO authorization module, which records the public key to IdP mapping in its database. The IdP web site binds the user's public key to the user's identity and the attribute(s) that it is willing to assert for this user.

The IdP uses the TLS session to send a new FIDO UAAF protocol message to the authorization module of the FIDO device, listing the attribute(s) that it is willing to insert in VCs for the user. The user either agrees to these or selects a subset of them. Both the FIDO authorization module and IdP store the user's VC attribute selection.

$$IdP \rightarrow User: \ Attribute^1 ... Attribute^n \quad (1)$$

$$User \rightarrow IdP: \ Attribute^i ... Attribute^j \quad (2)$$

Authorization at a Web Site (SP)

The SP prepares its authorization policies for the various services it offers. Each protected URL can have a different policy, as this allows the principle of least privileges to be adopted. Each authorization policy says which attributes from which IdPs are needed to access this resource. This policy is written in either disjunctive normal form (DNF) or conjunctive normal form (CNF) as these can represent any attribute-based access control policy [8].

An extension to the FIDO Metadata Service/Verifiable Data Registry may store the public keys and assertable attributes of the FIDO-enabled IdPs. Different communities can define different metadata extensions. Note that a metadata extension may contain more IdPs than any particular SP needs. There is no requirement for an SP to trust any of them as each SP makes its own trust decisions when formulating its authorization policies. However, the registry provides a reliable way of obtaining the information about those IdPs that it does trust.

When a user first contacts an SP's protected resource, the SP issues the standard FIDO registration request, and her device's FIDO authenticator creates a new key pair for the SP, that is, (PrK^{USP}, PuK^{USP}). The authenticator signs the public key with the attestation private key, that is, $sign_{PrK^{Att}}\{PuK^{USP}\}$, and this is returned to the SP via the FIDO client. Both the SP and the FIDO authorization module record this public key. Note that the SP only knows that this public key is associated with some user, but knows nothing further about the identity of the user (unless she has presented some self-asserted attributes). The SP now sends its authorization policy to the FIDO authorization module:

$$SP \rightarrow User: \ AuthzPolicy \quad (3)$$

The FIDO authorization module looks in its database (a.k.a. digital wallet) to see if the authorization policy can be matched by the attributes the user has already selected for assertion by her IdPs. If not, it tells the user she cannot proceed until she first registers with the IdPs that the SP trusts to issue the required attributes. Assuming the user already has the required VCs, the FIDO authorization module asks the user to confirm that it is okay to send the matching attributes[2] to the web site.

The FIDO client now performs a standard FIDO authentication exchange with the first IdP's VC issuing URL, and sends a new FIDO claim request message to the IdP. The claim request is passed to the FIDO authenticator to be signed by the private key created for this origin, that is, PrK^{UAA}, thereby obeying the SOP. The claim request[3] comprises the following information:

$$User \rightarrow IdP: Sign_{PrK}UAA\left\{\begin{array}{c} Attribute^1,...Attribute^n, \\ nonce1, timestamp \end{array}\right\} \quad (4)$$

The IdP validates the signature, and checks that the *timestamp* is valid and that *nonce1* has not been used before. The IdP looks up the user's entry in its database, determines which attribute(s) this user has consented to release, and compares this to the attributes that are being requested. Assuming these are a subset of the consented set, the IdP returns a second nonce to the user along with nonce1, both encrypted with the user's public key, viz

$$IdP \rightarrow User: \{nonce1, nonce2\} enc_{PuK^{UAA}} \quad (5)$$

This ensures that only the user can obtain *nonce2*. The authorization module asks the authenticator to decrypt message (5), and to sign two public keys, the IdP's and the SP's, that is, $\{PuK^{USP}, PuK^{UAA}\}$, with the attestation private key, to prove that both keys belong to the same user; otherwise, it would be possible for one user to pass on the public key of a second user. The authorization module packages the attested keys with the decrypted *nonce2*, adds a *timestamp*, and asks the authenticator to sign this credential request message[3] for the IdP before sending it to the IdP, viz

$$User \rightarrow IdP: Sign_{PrK}UAA\left\{\begin{array}{c} Sign_{\mathrm{Pr}K^{Att}}\{PuK^{USP}, PuK^{UAA}\} \\ nonce2, timestamp \end{array}\right\} \quad (6)$$

The IdP validates the signatures on the credential request and public keys, and from *nonce2*

[2] If the user has more that one IdP capable of issuing the web site's required attribute(s) (e.g., assume a MasterCard attribute is requested and the user has two issuing banks), the user will be asked to choose and consent to just one of them.

[3] This message requires an update to the UAF ASM application programming interface (API) commands, as it is a new signed-object. The current FIDO specifications do not allow extension data passed to the authenticator to be signed. Consequently in our proof of concept implementation this message is not signed.

knows which attribute request message it is linked to. The IdP now issues a set of VCs for the user by associating each requested attribute with her public key for the SP. Issuing VCs at the granularity of an attribute ensures *selective disclosure* if the user stores them in her wallet and uses them again in different aggregations.

The IdP returns the set of VCs to the user's authorization module. Each VC comprises the public key of the user for the web site, PuK^{USP}, the validity time (start and end times), and one of the attributes, and is signed by the private key of the IdP, viz

$$IdP \rightarrow User: credential^1, \ldots credential^n \quad (7)$$

where $credential^i$ =

$$Sign_{PrK^{AA}}\{PuK^{USP}, Attribute^i, startTime, endTime\}$$

Upon receiving the set of VCs, the authorization module checks the signatures, and if the VCs are authentic, stores them in its wallet until their *endTimes*, after which they are discarded.

The authorization module now repeats steps (4) to (7) for each IdP from the SP's authorization policy. This provides attribute aggregation. Once the authorization module has received all the VCs that are needed by the SP, it creates an Authorization Response message (i.e., a Verifiable Presentation) and passes this to the authenticator for signing[2]. The Authorization Response message comprises

$$User \rightarrow SP: Sign_{PrK^{USP}}\{credential^1, \ldots, credential^n\} \quad (8)$$

The SP validates the signatures, and from this knows:

- Some user, identified by the public key PuK^{USP}, possesses the corresponding private key PrK^{USP}.
- This same user has a set of VCs issued by a set of trusted IdPs, because they all contain the same public key.
- All the VCs were valid when they were issued and are still within their validity period (revocation is discussed below).

The SP does not know anything further about the identity of the user, unless she has presented some self-asserted attributes as well. Finally, the web site checks that the presented set of VCs matches its authorization policy, and if it does, grants the user access to its protected resource.

Re-Authorization and Revocation

If the user returns to the same SP, the authorization service can retrieve unexpired VCs from its wallet and send these to the SP, instead of asking the IdP to create fresh ones. There are two alternative VC expiration models that can be adopted by the IdP: revocation or short-lived.

In the revocation model, the IdP issues long-lived VCs that can be used multiple times over many months, but the IdP can revoke the VCs. Different revocation schemes have been proposed, including OCSP, CRL, CRT [9], and so on. From a privacy perspective, we recommend third-party merged revocation lists, to prevent the IdP indirectly tracking the user. To minimize CRL retrieval, the SP should have a maximum validity period (MaxVP) and only retrieve the CRL once per MaxVP. Another option is to use OCSP Stapling [10], where users periodically retrieve signed validity status statements (VSSs) from IdPs and store them in their devices. Users present their VCs and associated VSSs to the SP. However, using OCSP stapling is very similar to the short-lived model, but is more complex, so is not recommended.

In the short-lived model, the IdP issues short-lived VCs and does not perform revocation. Users may reuse VCs during their short validity period, but will need to fetch new VCs more frequently. Which model to use depends on the nature of the VC. Some VCs, such as date of birth, are naturally long lived. Others, such as permission to enter a high security area, are naturally short lived.

If any VCs are not acceptable to the SP, either because they have been revoked or are stale, it notifies the user about the VCs it is unwilling to accept, using the following message:

$$SP \rightarrow User: credential^1, \rightarrow, credential^n \quad (9)$$

Issuer Delegation of Authority

Delegation of authority increases the scalability of privilege management by distributing the users between multiple IdPs rather than a single IdP (the Source of Authority, SoA). Both downward and upward delegation mechanisms exist. An example of the former is the issuing of credit cards by banks, and of the latter is the issuing of national student cards in the United Kingdom by the National Union of Students.

In the downward delegation mechanism the metadata registry stores the public key of the SoA as well as a pointer to the SoA's metadata service. The latter contains a signed list of IdPs and their public keys, to which the SoA has delegated VC issuing. In this way the SoA can dynamically update its IdP list without effecting the main metadata service. Users register with one IdP and have their VCs issued by it. The SP, which trusts the SoA, can validate any of the issued VCs by checking that the public key of the IdP that signed the VC has been signed by the SoA in its metadata. Two signature verifications now need to be performed by the SP instead of one. However, the user experience remains the same, as does the VC and revocation issuing mechanisms.

In the upward delegation mechanism there is no need for an SoA metadata service. The IdPs delegate VC issuing to their SoA, which issues VCs on behalf of all the IdPs. An IdP's primary task is to manage its users and their attribute entitlements. Consequently, the user must additionally register with the SoA. This process must contain an authenticator known only to the user and the IdP, such as an OTP. The SoA contacts the user's IdP by some proprietary means to validate the authenticator and confirm that the user is entitled to the requested attribute(s) before issuing the VC. With this mechanism, the SP's authorization procedure does not change, since it only trusts the SoA, and all credentials are issued (and revoked) by the SoA. However, the user's registration procedure is more cumbersome and/or time consuming due to the double registration, and revocation involves communication between the IdP and the SoA.

Delegation of authority increases the scalability of privilege management by distributing the users between multiple IdPs rather than a single IdP (the Source of Authority, SoA). Both downward and upward delegation mechanisms exist. An example of the former is the issuing of credit cards by banks, and of the latter is the issuing of national student cards in the United Kingdom by the National Union of Students.

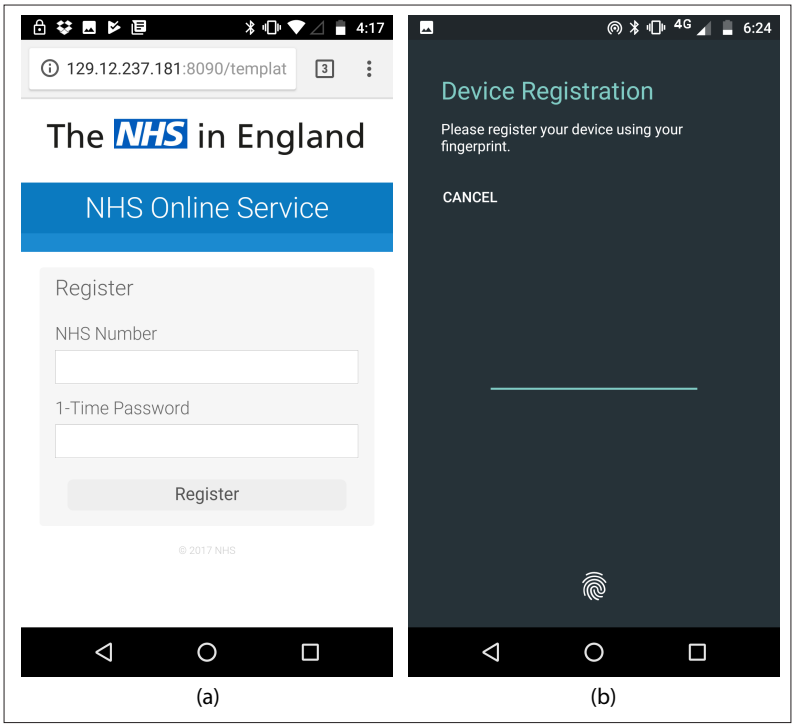

FIGURE 4. a) FIDO device registration at the NHS; b) the fingerprint reader.

Abusive Users

Users who abuse their privileges by exploiting an SP can be immediately blocked by the SP blacklisting the user's public key, PuK^{USP}. However, this does not stop the user from registering again with the SP and generating a new key pair. The privacy design of FIDO is such that the SP cannot tell when the same user is authenticating using a different public key. The solution is for the SP to contact the trusted IdPs who have asserted the VCs of the abusive user, and to inform them about the abuse and the public key that was used. Since the IdPs have records of the VCs they have issued, along with their public keys, they can identify the user. The IdPs can take appropriate action to revoke the privileges of the user, warn him, and so on without revealing the user's identity to the SP. If the abuse is more serious, which requires criminal prosecution, an IdP could reveal the user's identity to the SP if required by law.

The Implementation

We implemented the IdP and SP servers using the Spring framework by extending the UAF FIDO server provided by eBay [11]. We only issue short-lived VCs, so no revocation service is needed. Both our SP and IdP support the three standard FIDO UAF REST services (*regRequest*, *authnRequest*, and *uafResponse*). Our IdP supports four additional REST services: *attrList* returns message (1), *userSelectedAttrList* receives message (2), *credentialsToCertify* receives message (4) and returns message (5), and *credentials* receives message (6) and returns message (7). Our SP supports two additional REST services: *UAAFPolicyRequest* returns message (3), and *signedUAAFResponse* receives message (8), and optionally returns message (9).

We leave the management of the first step of the initial user registration (identification and authentication) to the IdP. This gives the IdP flexibility to choose the most suitable procedure, such as an OTP or physical presence.

Our UAAF client is implemented on both Android (version 6 and above) and iOS (10.1 and above). The software architecture on Android implements the FIDO Authenticator Specific Module (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 security enclave. Except for these technical differences, the core of the UAAF client is identical on both platforms. We extended the ASM to certify to the issuer that the authenticator manages two keys — message (6) — to stop two users colluding and sharing an IdP's issued VCs. The UAAF client stores the VCs in an SQLite database, allowing it to search for VCs that match the SP's authorization policy.

Use Case and User Trials

Missed doctor and hospital appointments cost the English health service nearly £1 billion a year [12]. Changing an appointment by telephone is time consuming, so patients frequently give up. We gave patients a smartphone and our National Health Service (NHS) app to allow them to instantly make and cancel hospital appointments and reorder repeat prescription drugs, another time-consuming process.

Patients were given a letter containing a fictitious NHS number, a 24-digit OTP, and the NHS URL to contact. On accessing this URL with the phone's browser, user registration takes place. A FIDO key pair is generated for the NHS IdP, and the user swipes her finger to confirm this (Fig. 4). The NHS number and OTP are transferred to the IdP, allowing it to authenticate the user. The IdP sends message (1) to the phone, and the user is shown the attribute(s) that are available (one in this case; Fig. 5a). The user chooses it, message (2) is returned to the IdP as a result, and the ability to obtain an *NHS-Patient* VC is recorded by the app.

The NHS will eventually offer many different online services, each requiring different VCs for fine-grained access control, so the patient must also get specific VCs for the specific NHS services she is using. In our use case, a hospital consultant (Dr. Nijjar) has an online service for making and cancelling appointments and ordering repeat prescriptions. When the patient visits the consultant, she is given a 4-character OTP, and asked to register with the consultant's IdP at the given URL in order to obtain the *Dr-Nijjar's-Patient* VC. The previous procedure is repeated with the new IdP. The patient now has the ability to obtain two VCs, one from the NHS and one from the consultant.

When the patient subsequently accesses the NHS hospital web site with her browser, she clicks the login button, which causes a new FIDO key pair to be generated. This will be used to authenticate the user in all subsequent exchanges. The hospital's authorization policy is sent to the phone (message (3)), which asks for the *NHS-Patient* VC. The app matches the policy with the available VCs and asks the user to consent to the release of her *NHS-Patient* VC (Fig. 6a). The app fetches this VC from the NHS IdP (messages (4), (5), (6), and (7)), stores it on the

phone, then signs it and sends it to hospital (message (8)). After validating the VC, the hospital web site displays the online services that are available to all NHS patients. Clicking on any of these will cause the respective authorization policy to be sent to the phone. The user selects the Consultant service, and the policy asks for the *Dr-Nijjar's-Patient* VC (Fig. 6b). The app confirms that the patient has this VC available, and requests it from the consultant's IdP. On receipt it is stored on the phone, signed, and sent to the web site, which validates it. A valid VC causes the site to display the three services that are available to the consultant's patients: make a hospital appointment, cancel a hospital appointment, and order repeat prescriptions.

We evaluated the usability of the app with patients waiting to see their consultant at University Hospital Southampton. We simply asked patients on arrival if they would like to trial a new mobile phone NHS app while waiting. We observed their behavior and made notes. When they finished, we asked them to complete a questionnaire, while we re-initialized the phone for the next patient. During the trial, the patient's fingerprints were protected from direct access and extraction by the native Android security mechanisms, and the smartphone was re-initialized immediately after the trial to remove all data related to the patient.

The questionnaire contained 10 questions that used a five-point Likert scale to ask about the usability, security, and privacy of the app. Four subsequent questions asked what the users thought about the app, and three questions captured their demographic information. Ten patients of three different ethnicities tested the app (9 female, 1 male) with ages ranging from under 20 to over 80. Only one patient had insufficient time to complete the questionnaire.

Prior to starting, each user was given a document explaining the background and setting the scene. For each task we had a set of written instructions to give to patients who got stuck. We wanted to see how intuitive the whole procedure was without any help being provided.

Observations and Discussion

Most users (9/10) had problems registering their fingerprint with the phone, and hence obtained further instructions for this. This was not unsurprising, as none of the users already owned the Motorola phone being tested. However, once given the instructions, all users found it easy or very easy to do. At the end of the trial, 9/10 users said they liked the use of fingerprints instead of username/passwords very much, and said they would always use the app if the NHS made it available. Seven free-form positive comments were provided and no negative ones. 5/10 thought fingerprints made the app more secure and 2/10 less secure. One elderly patient had problems typing in the 24-character OTP (Fig. 4a). She kept touching characters on the virtual keyboard next to the correct one, making the OTP incorrect. The observer eventually had to enter the OTP for her. In a live system the NHS will have to consider how best to balance usability and security, as we had erred on the side of security.

Most users (9/10) had problems getting a patient VC from the NHS because the screen was poorly designed (Fig. 5a), so they obtained the instructions. After this, all users found it easy or very easy to do. We subsequently redesigned this screen (Fig. 5b).

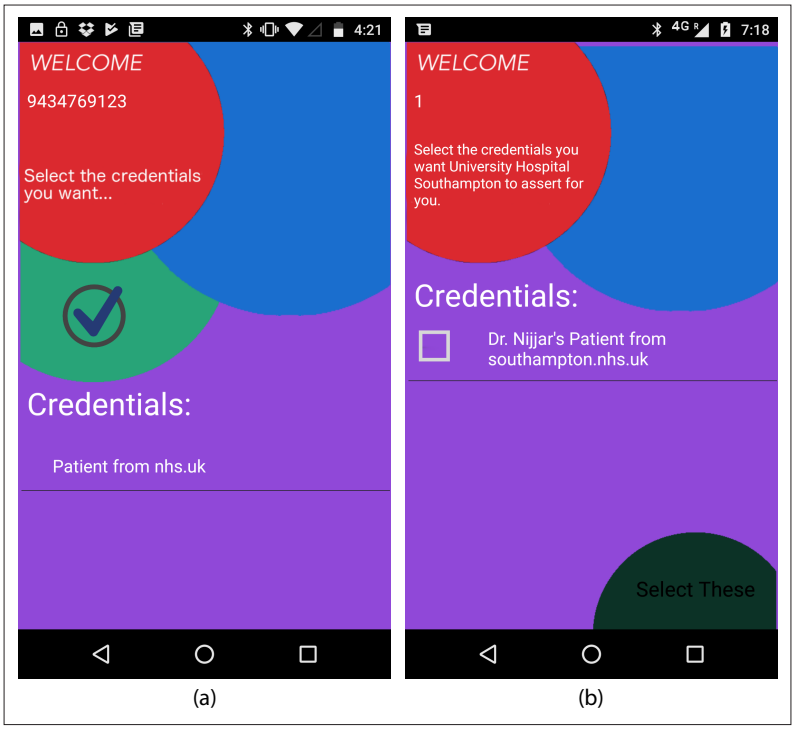

Figure 5. Attribute choice: a) original design; b) design after the trials.

Only one of the 10 had problems the second time around (i.e., getting a VC from the consultant) because a) this was a repeat procedure and b) the (pretend) consultant was present to help them if they got stuck. The patient who had difficulty was not helped by the (pretend) consultant so (s)he requested the instructions.

About half the patients (6/10) had problems booking an appointment and hence requested further instructions. This was because the hospital welcome page (which we had copied) had a *Sign In* button that was small and easily overlooked. Below this were a set of quick links, one being "Cancel your appointment," so naturally the patients looked for "Book your appointment," which did not exist. Once the sign-in problem was overcome, booking an appointment was very easy. We subsequently redesigned this screen by making *Sign In* more obvious and removing "Cancel your appointment."

The Repeat Prescription form listed the patient's drugs and had an *Order* button at the bottom of the page below the visible screen, so a few patients did not understand that they had to scroll down the screen. One patient asked for additional instructions. After the trials, we redesigned the screen by placing the Order button above the list of drugs.

Overall, we were extremely pleased with these (albeit limited) patient trials, because they employed a good cross-section of the population in age, and involved different ethnic origins and genders (although predominantly female). The trials found a few sub-optimal design features in the app that we subsequently corrected, but the patients unanimously liked the app, found it easy

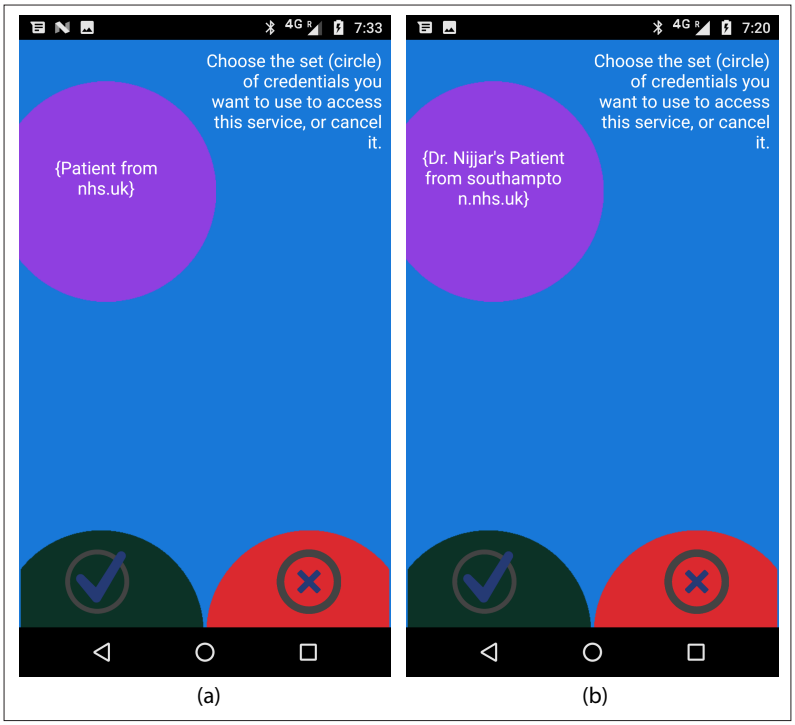

FIGURE 6. Display of the SP's matched authorization policy for: a) the hospital; b) the consultant service.

to use, preferred fingerprints to usernames and passwords, and were more than satisfied with its security and privacy preserving properties.

Discussion, Limitations, Further Work

Our user trials are limited and insufficient to generalize from, so more are needed.

The number of issued VCs can be large, as each attribute is issued for each SP. However, this is likely to be smaller than the number of assertions a typical SAML IdP will issue, as it has to issue a fresh assertion each time the user initiates a session with the SP. Our VCs can be reused multiple times.

The authorization module will need to store a large number of VCs, but since a single attribute VC in JSON-LD is smaller than 1 kB, the total storage for 2500 VCs will be less than 2.5 MB, which is easily manageable on today's smartphones.

We are currently migrating our implementation to FIDO2 [4].

Further work is needed to define the attributes that user communities will use for identification and authorization purposes. It is essential that IdPs and SPs have a common understanding of these attributes, so an ontology will be helpful, with class hierarchies and membership lists. Internationally standardized VCs, for example, mobile driving licenses and credit cards, will aid the universal recognition of VCs.

References

[1] *The Guardian*, "Huge Facebook Breach Leaves Thousands of Other Apps Vulnerable," 2 Oct 2018; https://www.theguardian.com/technology/2018/oct/02/facebook-hack-compromised-accounts-tokens, accessed 9 Sept. 2019.

[2] W3C Rec., "Web Authentication: An API for Accessing Public Key Credentials Level 1," 4 Mar. 2019; https://www.w3.org/TR/webauthn/, accessed 9 Sept. 2019.

[3] W3C Rec., "Verifiable Credentials Data Model 1.0 — Expressing Verifiable Information on the Web," 19 Nov. 2019; https://www.w3.org/TR/vc-data-model/, accessed 21 Dec. 2019.

[4] EU AARC Project Deliverable DNA2.4 "Training Material Targeted at Identity Providers," 27 July 2016; https://aarc-project.eu/wp-content/uploads/2016/07/AARC-DNA2.4-v3.pdf, accessed 9 Sept. 2019.

[5] EC AARC Project AARC Blueprint Architecture, 18 Apr. 2017; https://aarc-project.eu/wp-content/uploads/2017/04/AARC-BPA-2017.pdf, accessed 9 Sept. 2019.

[6] S. Cantor, "NativeSPAttributeResolver," 7 Apr. 2014; https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPAttributeResolver, accessed 9 Sept. 2019.

[7] FIDO Alliance, "FIDO UAF Architectural Overview," Proposed Standard, 8 Dec. 2014; https://fidoalliance.org/specs/fido-uaf-v1.0-ps-20141208/fido-uaf-overview-v1.0-ps-20141208.html, accessed 9 Sept. 2019.

[8] B. Moore *et al.*, "Policy Core Information Model — Version 1 Specification," RFC 3060, Feb. 2001.

[9] P. Wohlmacher, "Digital Certificates: A Survey of Revocation Methods," *Proc. 2000 ACM Wksps. Multimedia*, pp. 111–14. DOI=http://dx.doi.org/10.1145/357744.357892.

[10] Y. Pettersen, "The Transport Layer Security (TLS) — Multiple Certificate Status Request Extension," RFC 6961, June 2013.

[11] The eBay UAF FIDO Implementation; https://github.com/eBay/UAF. accessed 9 Sept. 2019.

[12] *BBC*, "NHS to Reveal Cost of Missed Appointments to Patients" 3 July 2015; https://www.bbc.co.uk/ news/uk-33375976, accessed 9 Sept. 2019.

Biographies

David W. Chadwick is a professor of information systems security at the University of Kent, Canterbury, United Kingdom, and a co-author of the W3C Verifiable Credentials Data Model recommendation. He specializes in authorization, identity management, trust management, and privacy. He is a BSI representative to ISO and ITU-T standards meetings on X.509, mobile driving licenses, and identity management.

Romain Laborde is an associate professor at the University of Toulouse (Paul Sabatier- IUT 'A'), France. He is also a member of the Institut de Recherche en Informatique de Toulouse. He received his Ph.D. in computer science from University Paul Sabatier in 2005. Then he was a research associate in the Information Systems Security Group in the Computer Science Department, University of Kent. His research focuses on security management applied to network security configuration, identity and access management, and privacy.

Arnaud Oglaza is a postdoctoral researcher for the French National Center for Scientific Research at the Institut de Recherche en Informatique de Toulouse. He received his Ph.D. in computer science from the University of Toulouse in 2014. His research focuses on security management, identity and access management, privacy, and machine learning.

Rémi Venant, PhD is an associate professor at Le Mans University, France, a member of the Laboratoire d'Informatique de l'Universit du Mans, and a member of the Institut de Recherche en Informatique de Toulouse. His research mainly focuses on technology-enhanced learning within the fields of learning analytics, artificial intelligence for education, and learning environment design. He is also involved in security management research activities that tackle identity and access management and privacy issues.

Ahmad Samer Wazan, Ph.D., is an associate professor at Zayed University, United Arab Emirates. His research topics include trust management, PKIs, access control, and, more recently, security requirement engineering. His research group originally proposed to extend the X.509 trust model by adding a new entity called, the Trust Broker. This proposal was accepted by ISO/ITU_T and included in the 2016 edition of X.509.

Manreet Nijjar is a consultant physician in infectious diseases and acute medicine at Barts Health NHS trust and an NHS England Clinical Entrepreneur Fellow. He co-founded Truu, a company that specialises in digital trust, credentialing and creating a digital passport for healthcare workers and a passwordless single sign-on solution for organizations. Leveraging emerging technologies such as distributed ledger technology, decentralized identifiers, and verifiable credentials. he believes these technologies will facilitate trust as healthcare undergoes a rapid digital transformation built on the principles of open standards, collaboration, and privacy by design.