Establishing Trust in SSI Verifiers

David W. Chadwick[1], Michael Kubach[2], Ioram Sette[3], Isaac Henderson Johnson Jeyakumar[14]

Abstract: We present a conceptual model that enables a user/holder with a wallet holding W3C Verifiable Credentials (VCs) to determine if the verifier is trusted to conform to GDPR so that it might be given the user's personal identifying information contained in their VCs. We describe the implementation of this model using the TRAIN trust infrastructure and how wallets might interoperate with verifiers using different trust infrastructures. This leverages the OIDC GAIN proof of concept network currently being built using the draft OIDC Federation specification. We briefly describe the experiments that we have undertaken to date and the research that is still outstanding.

Keywords: Self-sovereign identity, digital identity, decentralized identity, identity management, trust registry, trusted issuers, trust lists, digital wallet, GAIN, TRAIN, OIDC Federation.

1 Introduction

The three primary entities in self sovereign identity (SSI), based on the W3C Verifiable Credentials (VCs) Data Model [W321], are the issuer, holder and verifier. The VC trust model requires both the holder and the verifier to trust the issuer, at both the technical and administrative levels. The technical level is implemented using cryptography, typically digital signatures, to ensure that the verifiable credential has not been tampered with since it was issued. In a previous paper [Jj22] we describe how we have implemented trust in the issuer at the administrative level using the TRAIN infrastructure. TRAIN uses ETSI TS 119 612 standard trust lists [Et16] to hold details about the VC Issuers and credential schemas, and these lists are pointed to from DNS entries that are controlled by the administrators of the various trust schemes[5] (or trust domains). The use of the DNS (DNSSEC [ls11] is recommended) provides decentralised administration, an established and trusted infrastructure, and scalability to global proportions. It also allows one trust domain administrator to cross certify another trust domain by including a pointer to the latter's DNS entry in its own DNS entry. The use of TRAIN to ensure that issuers are trustworthy has been validated in several interworking trials [Ju21, Ya22, Ng22].

[1] TrueTrust Ltd., Chartham, CT4 7TG, UK, d.w.chadwick@truetrust.co.uk
[2] Fraunhofer IAO, Nobelstr. 12, 70569 Stuttgart, Germany, firstname.lastname@iao.fraunhofer.de
[3] CESAR, Rua Bione, 220, Recife, PE, Brazil, ioram.sette@cesar.org.br
[4] University of Stuttgart, Institute of Human Factors and Technology Management IAT, Allmandring 35, Stuttgart, 70569, Isaac-Henderson.Johnson-Jeyakumar@iat.uni-stuttgart.de
[5] TRAIN uses the term 'trust scheme', while OpenID Federation uses 'federation' when referring to what we call 'trust domain' in this paper: a group of entities that share a common trust model, processes, and procedures.

cbe doi: 10.18420/OID2023_01

In this paper we turn our attention to trust in verifiers. The SSI model ensures that issuers do not know who the verifiers are, since the VCs are given to the holder, and the holder passes them to verifiers. So trust in verifiers becomes the responsibility of the holder. Since holders are typically human beings, this is placing a great deal of responsibility on people, which most of them are not skilled enough to undertake. Requiring users to know which verifiers to trust is very similar to asking users to know which websites to trust, even when they have not visited them before. Browsers and web PKI now facilitate this process by using trusted third parties, namely X.509 Certification Authorities (CAs), to issue X.509 domain validated public key certificates (PKCs) to web sites. Web browsers indicate if a secure TLS session has been established to the domain validated web site by displaying a lock icon next to the web site's URL. Something similar will be needed for SSI if VC eco-systems are to enable human users to determine if a verifier is trustworthy or not. The German ID-Wallet can serve as a negative example, illustrating possible consequences of not integrating a mechanism for the establishment of trust in verifiers. The project had been initiated by the German government, was supported by a number of big German companies and the wallet was briefly available in the App Store. Soon after the release it was pulled off the App Store after it had been demonstrated that due to the lack of verifier identification a man-in-the-middle-attack could be possible [Mr21]. The project was cancelled and this severely damaged the reputation of Self-sovereign and decentralised identity solutions in the public discussion in Germany.

The rest of this paper is structured as follows. Section 2 describes previous research in this area. Section 3 describes our conceptual model for trust in verifiers, whilst section 4 describes our implementation. Section 5 concludes with a description of the evaluation tests we have performed, the current limitations, and where further research is still needed.

2 Prior Research

[Section cut due to page limit. Full paper available at: https://s.fhg.de/TrustFullPaper]

3 Conceptual Model

In the VC model, the holder contacts various issuers to obtain their VCs and stores them in their digital wallet. This wallet may be held locally on the user's device e.g. laptop or smartphone, or remotely e.g. in a cloud. The holder then transacts with various verifiers and sends them the VCs they request. This raises two primary trust questions: (1) Are the holders connected to the verifiers that they think they are? (2) Is the verifier a responsible entity i.e. can it be trusted with the personal identifying information (PII) that it asks for?

The first question is a technical trust question. For web-based VC transactions, it can be answered in the same way that it is answered for normal web-based transactions i.e. the wallet utilises a TLS session between it and the verifier's web server, to ensure it is

connected to the same entity as identified in their X.509 PKC. For cloud-based wallets the TLS session can be mediated by the user's browser on the device that he/she is using. For device-based wallets, the browser on the device can also be used to mediate the TLS session with the verifier's web server. For face-to-face transactions e.g. a holder wishing to enter a nightclub, the user must use the local environment to determine if the verifier is who they think it is, before displaying their VCs on their device. Face to face verification of the identity of the verifier can be more reliable than remote web-based verification, because TLS sessions are not without their problems e.g. as described here [Aw17]. Nevertheless, identifying a remote SSI verifier is no less risky for the human holder than it is in performing identification of a web server in a normal browser-based interaction.

The second question is about administrative trust. The simplest solution is to place the responsibility for this trust in the domain administrator and to require it to only add verifiers to its trusted list of verifiers when the latter are deemed to be trustworthy. The wallet must then contact the administrator's trust list before allowing the user to send any VCs to the verifier. If the verifier is present in the trust list, then the wallet can allow the user to continue with the transaction. If the verifier is not listed in the trust list, two wallet behaviours are possible: (1) a "strict" wallet can abort the user's transaction and not allow the user to continue, whereas (2) a "permissive" wallet can warn the user that the verifier is not in the administrator's trust list, and then either ask the user if they wish to proceed or, recognising that most users always select Yes to such questions, simply warn the user that the verifier is not trusted, and then allow the user to proceed. In order for this solution to work, wallets need to be configured with the administrator(s) of the trust list(s) for the trust domain(s) that the user is a member of. In the case of TRAIN, this equates to the DNS name controlled by the domain administrator, so one would think it is quite straightforward to implement this in wallets.

But this raises a practical issue, akin to the browser issue of knowing which CAs should be trusted. We cannot expect the user or wallet to be familiar with all the SSI trust domains in the world, or worse still, which trust domain administrators should be trusted. So should we expect wallets to come pre-configured with the list of trust domain administrators? Clearly not, as today there are very few SSI trust domains. Furthermore, we might expect the number of trust domains to grow substantially and quickly as SSI becomes established. This will mean that wallet software will continually need to be updated with the list of trusted administrators. The implication of this solution is that the wallet software provider becomes the ultimate arbiter of which trust domain administrators are deemed to be trustworthy. One can see that this problem is identical to the one that web browser suppliers first encountered in the 1990s when the web PKI started to evolve and be rolled out across the global Internet. Web browser suppliers initially each configured their own (different) list of trusted CAs into their browsers and allowed users to add and remove root CAs from this list. Then some of the browser suppliers switched to using the CA trust lists provided by the operating system providers. The CAB-Forum was initiated in 2004 in order to provide a set of rules that CAs should follow in order to become trusted CAs that could be added to such trust lists. Clearly SSI is just at the start of this journey and is at a comparable stage that X.509 web PKI was at during the 1990s.

Notwithstanding the issues with wallets determining which trust domain administrators should be added to their configuration information, this still places a lot of burden on each trust domain administrator. In order to determine the trustworthiness of a verifier the following sorts of questions will need to be answered by the trust domain administrator. 1. Will the verifier request and keep the holder's VCs securely? 2. Will the verifier dispose of the VCs once they are no longer required? 3. Will the verifier not pass the VCs to third parties without the user's consent? 4. Are the VCs the ones necessary for the current transaction, or is additional unnecessary PII being requested? Obviously, these questions fall within the remit of the GDPR, namely the principles of data minimisation, purpose limitation, integrity and confidentiality (security), storage limitation, and lawfulness, fairness and transparency. How can we address these in our conceptual model? (Note that GDPR is mentioned here as an example of an important data protection regulation – outside the EU other regulations might be relevant.)

Purpose limitation can be addressed by the verifier only requesting the VCs that are required for the current transaction that the user has requested. Data minimisation can be addressed through the concept of selective disclosure of the requested VCs. Integrity and confidentiality can be addressed by transferring VCs that are digitally signed by their issuers, and that are encrypted by the wallet specifically for the verifier. Storage limitation can be addressed by the wallet placing a time limit on the verifiable presentation and requiring the verifier to delete the VCs once this time has expired. Of course, the wallet cannot technically enforce the subsequent deletion of VCs, nor can it enforce the subsequent illegal disclosure of the VCs to third parties that the user has not consented to. The enforcement of these will need to rely on regulations and legal means, as will all the GDPR principles of lawfulness, fairness, and transparency.

4 Implementation Details

4.1 Trusted Administrators

In our initial implementation our wallet allows the user to configure the identities (i.e. DNS names) of trusted TRAIN domain administrators into the wallet, since this gives us much more immediate control over administrative trust in an experimental environment rather than having to continually update the wallet software with the wallet provider's list. Our wallet has integrated the open source TRAIN API, called the Automatic Trust Verifier (ATV) into its interactions with verifiers. In order to support both trust in issuers and trust in verifiers, we enhanced the TRAIN API to include the service name:

```
{ "ServiceURL": "<string>",
    "Trust_Scheme_Pointer": ["<string>" ],
    "ServiceName": "<string>" }
```

where "Trust_Scheme_Pointer" is the DNS name of a trusted domain administrator, "ServiceURL" is the URL of a remote verifier (or issuer), and "ServiceName" is the type

of service it offers, i.e."Verifiable Credential Verifier" (or "Verifiable Credential Issuer"). The API returns a response detailing whether the verifier is trusted or not by this domain administrator. If not, the wallet warns the user that the verifier cannot be trusted, and then invites the user either to continue or to abort the transaction (i.e. the permissive approach). As can be seen, integration of TRAIN into a wallet is a relatively simple task.

4.2 Holder-Verifier-Protocol

The protocol that we used in our implementation was the draft OpenID for Verifiable Presentations (OpenID4VPs) [Op22] protocol that is being defined by the OpenID Foundation. This protocol is the ubiquitous OAuth2 authorization request protocol [Dh12] with the added ability for the verifier to request a vp token (a new type of OAuth2 token) and for the wallet to return this containing a set of VPs in the authorization response. Thus in the simplest case there only needs to be two round trips between the wallet/device/ browser and the verifier/RP/client. The first https message from the browser to the RP details the transaction that is requested, and this solicits an OAuth2 client authorization request to the wallet. The wallet's authorization response containing the vp token is sent to the verifier's URL (i.e. the RP's redirect URL) which is sufficient for the RP to provide its transaction response to the browser. The OAuth2 client request allows the verifier to request any combination of VCs that it needs, along with the selective disclosure of each VC's various properties. This is achieved by utilising the DIF Presentation Exchange (PE) protocol [Di22]. DIF PE thus allows the verifier to conform to GDPR's principles of data and purpose minimisation. The OpenID4VPs protocol allows the response to be encrypted, thereby facilitating integrity and confidentiality. The VC Data Model allows the VP to have a validity period, thereby addressing the storage limitation.

Unfortunately, the initial draft specification of OpenID4VPs does not take into account the concept of administrative trust in verifiers, so we have proposed additions to the draft protocol to cater for this. The most significant change is the addition of a new parameter to the OpenID4VPs client request protocol, and to the client's metadata file, namely, the trust_methods_supported parameter, via:

```
{ "trust_methods_supported": [{
        "type": "<some URl>",
        "<URI specific object>": {} }] }
```

where: "*type*" allows the RP to specify the type of trust method, e.g. TRAIN, OpenID Federation etc. (see later) - represented as a URI to ensure global uniqueness. "*URI specific object*" allows the RP to indicate the parameters that are required by this type of trust method. In the case of TRAIN, this parameter will be a set of DNS names, where each DNS name refers to a trusted domain administrator. So for the TRAIN trust method the above structure becomes:

```
{ "trust_methods_supported": [{
                "type": "https://train.trust-scheme.de/info",
                "trust_schemes": ["https://eu.ngi.train.trustscheme.de"]}] }
```

How the wallet/holder obtains this parameter is a matter of choice by the RP. The RP could include the parameter in the client request message, in its metadata file, or both.

The remaining changes we made to the OpenID4VPs protocol involve the way in which existing OpenID4VPs parameters are used. Our implemented procedure is as follows:

i) The RP must assert its client_id in the OpenID4VPs request. This client_id must be the verifier's URL registered with the trust domain administrator.

Note. It does not matter if the RP gives a false client_id e.g. pretends to be a trusted verifier, as our trust validation procedure addresses this.

ii) The wallet determines the trust methods supported by either obtaining the parameter from the authorization request message, but if it is not present, by reading in the RP's metadata from the well known page https://<client_id>/.well-known/openid-credential-verifier. If the metadata URL was present in the RP's request (as allowed by the OpenID4VPs protocol) then it MUST be equal to the well-known URL of the client_id, otherwise the RP's request cannot be trusted. It should either be rejected with a 400 Bad Request response with the error 'invalid_request' as defined in OAuth 2.0 https://www.rfc-editor.org/rfc/rfc6749#section-5.2 (strict wallet), or the user informed that the RP is untrustworthy (permissive wallet).

Note. This is to stop an untrustworthy RP from pointing to the good metadata of a trustworthy verifier.

iii) If the RP has not provided any trust_methods_supported, or the wallet does not recognise any of the provided trust methods, it either rejects the authorization request (strict wallet) or tells the user "warning this web site <client_id> cannot be trusted. Do you want to proceed?" - allowing the user to either quit or continue (permissive wallet). If the user is willing to continue, then the wallet behaves as if the RP is trusted.

Note. An untrustworthy RP that the user says can be trusted is allowed to send its own metadata in the request, and not have it at the well-known endpoint, because the user is willing to trust the RP. But if the RP is found to be trustworthy by the trust infrastructure then its metadata must be at the well-known endpoint.

iv) If the wallet recognises a trust method of type X (in our case "https://train.trust-scheme.de/info") that they have in common, the wallet calls its trust service API to ask if the verifier with URL <client_id> is trusted.

Note. If the trust service is temporarily unavailable then the wallet will have to behave as if the verifier is not trusted. This is akin to a CRL not being available in web PKI.

v) If the result is negative (meaning the client_id is not trusted) then the wallet either rejects the authorization request (strict wallet) or tells the user "Warning this web site <client_id> cannot be trusted. Do you want to proceed?" (permissive wallet) - allowing the user to either quit or continue.

vi) If the user decides to quit (permissive wallet), or the wallet decides (strict wallet) then the wallet returns 400 Bad Request to the redirect_uri of the authorization request with the error 'invalid_request' as defined in OAuth 2.0 https://www.rfc-editor.org/rfc/rfc6749#section-5.2

vii) Assuming the RP's client_id is trusted, the wallet will subsequently ask the user for consent to send the VCs to <client_id>.

Note. The client_id might be different to the URL of the web page that the user is browsing, so the user must be told the URL to where their PII is being sent.

viii) The wallet retrieves the metadata of the RP from https://<client_id>/.well-known/openid-credential-verifier if it has not already done so.

ix) The retrieved metadata contains the redirect_uris parameter for the vp_token (containing the VPs) to be returned to. If only one URL value is present then the wallet should use this to send the vp_token to. If multiple values are present, then the wallet should check that the redirect_uri from the RP's OpenID4VP's request is present in this set, otherwise return the error 400 Bad Request.

Note. The wallet now knows this return URL is a trusted endpoint to send the user's PII to, because the RP's client_id URL is trusted, so the metadata at its well-known endpoint is trusted, so the return_url is trusted.

4.3 Selecting the VCs to be Returned

As stated above, the OpenID4VPs protocol requires a DIF PEv2 request to be incorporated into its protocol request message. The OpenID4VPs protocol allows the DIF PE request to be either embedded in the request (request by value) or to be referred to via a URL (request by reference). The latter allows verifiers to publish their PE requests in a public repository - a policy server. The advantages of this are several. First it allows multiple verifiers to share the same PE policy. Secondly, it allows a verifier to be audited for conformance to data protection legislation. Consequently, we require the PE policy to be transferred by reference in order to enforce legislation. Consider that a trust administrator validates a RP/verifier and checks that it only requests the VCs that it requires for a particular service, and furthermore only requests selected properties from these VCs i.e. selective disclosure. However, after the verifier is audited by the trust administrator to ensure that it conforms to the legislation's data and purpose limitations, then if the RP is able to dynamically send any PE request to the wallet, it could sidestep its audited method without the trust administrator being aware of this (unless it constantly monitors the requests that the verifier sends to wallets). Our procedure ensures that the verifier abides by its audited PE policies by storing their URLs in the ETSI trust list, and the policy server not allowing URLs to be re-used for different policies.

Consequently, for trusted verifiers, the wallet will only pick up the PE policy from the reference URL sent in the OpenID4VPs authorization request message that is confirmed by the TRAIN API. If the PE policy is embedded in the OIDC4VP request message, then the verifier will be switched from trusted to untrusted, as untrustworthy verifiers are allowed to send any PE policy by value (to permissive wallets). We have used an existing parameter of the ETSI trust list (the TSPServiceDefinitionURI parameter) to return the URL of the PE policy to the wallet. The wallet can now compare this URL with that provided in the presentation_definition_url of the authorization request message. If identical, the wallet knows that the verifier is only requesting the VCs and their properties that the trusted administrator has verified are necessary for the requested transaction, and

that comply with the legislation. An example trusted verifier service specification using the ETSI standard is included in the full paper: *https://s.fhg.de/TrustFullPaper*.

Points to note in the above are the following. The *service type identifier* and *service name* are fixed fields indicating the TSP is a VC verifier, *Service digital identity* contains the public key of the verifier, and may be used by the wallet to encrypt the returned VP. *Service status* is fixed and defined by ETSI. *Service supply point* is the same URL as the *client_id* that the verifier sends in the OpenID4VPs authorization request. *TSP service definition URI* is a pointer to the PE policy stored in a public policy server. It is expected that RPs will have a one to one mapping between service supply points and TSP Service Definitions, i.e., each service will have its own specific PE policy. Consequently, these two sets of URLs should be treated as ordered sets and should each have the same number of elements. As part of the eSSIF Lab PolicyMan project [Pm20] we developed the policy server as open source code and it is available here [GV23].

4.4 Multiple TRAIN Trust Domains

In the NGI Atlantic project "Next Generation SSI Standards" [Ng22] we performed interworking tests between the USA and Europe using profiles of the OpenID4VCs protocol suite, and the TRAIN trust infrastructure. Because our tests were being performed in the USA and Europe we decided that two TRAIN trust domains provided a more realistic scenario than one international trust domain incorporating both the USA and Europe. Furthermore this would demonstrate the scalability of TRAIN and the ability of multiple trust domains to interwork. Whilst multiple domains is a feature and part of the original TRAIN design, interworking between domains had never been realised in practise before our NGI project. The DNS names of our two trust domains were: `us.ngi.train.trust-scheme.de` and `eu.ngi.train.trust-scheme.de` .

The associated US trust list contained details of the VC Issuers and the Verifiers/RPs of trusted US organisations, including Spruce the US NGI project partner. The EU trust list contained details of the trusted EU organisations' Issuers and RPs/Verifiers, including those of Crossword, one of the European NGI project partners. If we followed the simplest TRAIN implementation then neither Crossword nor Spruce would be able to trust each other as they are in separate trust domains i.e. if a Crossword Verifier or Wallet received a request from a Spruce system, then asking the TRAIN API if Spruce was a member of the eu.ngi.train.trust-scheme.de would yield a negative result (and vice versa for a Spruce component asking if a Crossword component was a member of us.ngi.train.trust-scheme.de). There are several potential solutions to this issue: (1) the recipient expands its supported trusted domains, (2) the trust domain administrators cross certify each other, and (3) the trust domain administrators create a hierarchical trust infrastructure.

Expanding supported TRAIN Trust Domains: In this mode of working, the recipient (Verifier/Wallet) decides which trust domains it trusts, and configures this list of trust domains into its software. So both the wallet and verifier will configure both

us.ngi.train.trust-scheme.de and eu.ngi.train.trust-scheme.de into their trust lists. The issuers won't change. If a Spruce wallet gets a request from either a Spruce verifier or a Crossword verifier then the wallet will be able to call the TRAIN API to check if the verifier is trusted ("Is Spruce or Crossword a member of the US trust scheme?", "Is Spruce or Crossword a member of the EU trust scheme?"). Similarly, if a Verifier receives a VC from either Spruce or Crossword repeated calls to the TRAIN API will be able to find out that the Issuer is trusted. Configuring Verifiers with multiple trusted administrative domains is a viable option, but is less so for Wallet apps downloaded from App Stores as they will either need to be continually updated or users will need to add trust domain DNS names to the configuration data of their wallets. (Note. This is the pragmatic approach we implemented in order to facilitate testing). It should be noted that wallet apps that have not been updated with new trust domains will fail to validate Verifiers from these domains. The major disadvantage of this approach is inefficiency and limited scalability. As the number of trust domains grows so will the list of trusted domains. Hence, an increasing number of calls will need to be made to the TRAIN API and this will become unsustainable. The next method solves this problem.

Trust Domain Administrators Cross Certify Each Other: In this mode of working, a trust domain administrator decides that another trust domain is equivalent to its trust domain, so it adds a PTR record in its DNS entry that points to the other trust domain, e.g. the us.ngi.train.trust-scheme.de administrator decides that the European trust scheme is equivalent to (or better than) its scheme, and so adds a pointer to the eu.ngi.train.trust-scheme.de trust domain to its DNS record. US Verifiers will continue to ask the TRAIN API if the Crossword Issuer is a member of the US trust domain, and will now get a positive result. Conversely, if the US administrator decides that Europe is no longer trusted, and removes the PTR record from its DNS entry, then US Verifiers will instantly cease to accept Crossword issued VCs.

Wallets should ignore all trust schemes that RPs say they are a member of, and instead should always ask the TRAIN API if the remote Verifier is a member of their (one and only) trusted domain. Wallets in the US trust domain should obtain a positive answer for both Spruce and Crossword verifiers, because the TRAIN API will find the PTR to both Europe and the US in the US DNS entry and will check the European trust list in addition to the US trust list. However, wallets in the European domain will not trust US Verifiers until the European administrator adds a pointer to the US trust domain in its DNS entry.

Hierarchical Trust Infrastructure: In this mode, one trust domain acts as the main trust-root and all entities configure its DNS name into their systems. The trust administrator of this domain adds PTR records to all trusted subordinate domains, so that their members automatically become trusted by every other entity. The difference between this mode of working and the previous ones is that it now requires all entities to place transitive trust in all domain administrators (including their own), and direct trust in the root domain administrator. In the previous two methods entities had direct trust in their own domain administrators. Hence from a trust perspective, the hierarchical mode is inherently less trustworthy than the other two modes.

4.5 Support for Different Trust Methods

The array of supported trust methods described in 4.2 above, allows the RP/verifier to be a member of more than one trust method. We are already aware of several different trust methods in existence today. The most obvious one is TRAIN, which the authors have been developing. But there is also OpenID Federation [Of22], which details the protocols for the operation of an OpenID Federation in which OpenID providers and RPs can publish their trust related meta-information at well-known locations. Yes.com also has its own proprietary trust method [Ye19]. The OpenID Foundation's GAIN project [Ga22] is building a proof of concept system to link these different trust federations together, in which TRAIN, Yes.com and others will all participate. The common linking protocol for administrative trust will be OpenID Federation. In order for a verifier to indicate that it supports the GAIN trust method and is a member of the GAIN federation, it would set its trust methods supported parameter to the following:

```
{ "trust_methods_supported": [{
   "type":"https://openid.net/specs/openid-connect-federation-1_0.html",
       "trust_anchors": ["<https URL of its GAIN trust anchor>"]}] }
```

where: *type* indicates that the method is OpenID Federation, *trust_anchors* lists the trust anchor(s) of the GAIN federation(s) (i.e. root(s) of trust) that this verifier is a member of.

But what if the RP and wallet support different trust methods? For example, say the wallet supports TRAIN and has the TRAIN API integrated into it, whilst the RP supports Yes.com's trust method and has the Yes.com specific trust parameters configured in. GAIN provides a solution that requires both trust methods (TRAIN and Yes.com) to implement an adaptor to connect to GAIN. The adaptor must use the OIDC Federation protocol (identified by the type: https://openid.net/specs/openid-connect-federation-1_0.html) and publish its publicly accessible trust_anchor URL. Any entity calling the GAIN method with this trust_anchor URL will learn that the Verifier is a trusted member of the domain identified by this trust_anchor URL. Now the RP will say that its trust methods supported are Yes.com and GAIN/OIDC Federation.

The wallet also needs to have the adaptor code built in to find out about GAIN members. When the wallet gets a match on the GAIN trust method, it calls its local GAIN API (TRAIN adaptor code) to ask if client_id is a member of GAIN and will get a positive or negative answer. The wallet then proceeds as before. When the wallet has been configured to support GAIN, it will recognise the trust method type parameter https://openid.net/specs/openid-connect-federation-1_0.html. It then compares the trust anchor URL(s) sent by the RP with the configured trust anchor URL(s) (either entered by the wallet user or built in by the wallet software provider). If a URL matches, the wallet knows that it and the RP have a common trust anchor. But if the URLs do not match then the wallet knows that it cannot determine the trustworthiness of the RP by calling GAIN, so it moves to the next supported trust method to see if it has a match. Assuming there is a match on a GAIN trust anchor URL, and that the local GAIN API has the following JSON input parameters:

```
{   "entity": "<string>",
  "trust_anchor": ["<string>" ],
  "ServiceName": "<string>" }
```

where: *entity* is the URL or the RP i.e. client_id, *trust_anchor* is the common trust anchor shared by the wallet and the RP, *ServiceName* is the service that the wallet is looking for, in this case "Verifiable Credentials Verifier". Then, the wallet's GAIN API (TRAIN adaptor code) will perform the following steps: 1. extract the trust anchors parameter from the GAIN "type" (which is "https://openid.net/specs/openid-connect-federation-1_0.html"), and obtain a set of URLs. 2. Remove https:// from the first trust anchor URL. 3. The resulting DNS name as trust scheme pointer is to be passed to the TRAIN API. 4. Call the TRAIN API as before passing it:

```
{   "ServiceURL": "<client_id>",
  "Trust_Scheme_Pointer": [
        "<trust scheme DNS name obtained from GAIN trust anchor URL>"],
  "ServiceName": "Verifiable Credentials Verifier" }
```

The TRAIN API should now send the same response as before, which the GAIN adaptor API can return to the wallet for it to process as before: 5. If it is a negative result, move to the next trust anchor URL, 6. if it is still negative, move to the next trust method, 7. if it is still negative, either cease the communications (strict wallet) or tell the user that the verifier cannot be trusted (permissive wallet).

5 Evaluation and Conclusion

The evaluation of the current implementation is ongoing. As part of the [Ng22] project we added verifiers to our TRAIN trust lists and updated our Crossword wallet to call the TRAIN API whenever a verifier claimed that it supported TRAIN. We allowed wallet users to add DNS names of trusted administrators to the configuration data and gave the user a warning message when the verifier was not trusted, i.e., we implemented and tested the permissive approach. We required the wallet to pick up the PEv2 request from the policy server. The following features are yet to be implemented and tested in future project work. (1) A strict wallet that does not allow users to send their VCs to untrustworthy verifiers. We assume that users will dislike this functionality, as in their eyes, usability trumps security and privacy. This hypothesis should be evaluated in a user study. In addition, particular issuers may not want their VCs to be presented to untrustworthy verifiers. Thus, we need to investigate whether individual VCs could be marked as strict or permissive, and wallets should act accordingly with these different types of VC. (2) An immutable policy store that does not allow verifiers to update their PE policies. Our existing policy store allows PE policies to be updated. A future version will require policy URLs to be persistent and their stored policies to be immutable in order to provide strict support for GDPR. (3) Support for different trust methods. Since our project partners only supported TRAIN it was not possible to perform interworking tests with verifiers or wallets that supported different trust methods, or to implement support for GAIN. Finally, trust in the wallet software is a bigger issue that has not yet been addressed. This is a topic that we expect to investigate in a subsequent research and development project.

Bibliography

[Aw17] Wazan, A.S. ; Laborde, R. ; Chadwick, David ; Barrere, F. ; Benzekri, A.: TLS Connection Validation by Web Browsers: Why do Web Browsers still not agree? In: 2017 IEEE 41st COMPSAC. Turin, S. 665–674, 2017.

[Dh12] Hardt, D.: The OAuth 2.0 Authorization Framework, IETF RFC 6749. 2012.. http://tools.ietf.org/html/rfc6749, accessed 23/01/18.

[Di22] DIF: DIF Presentation Exchange," 2022. https://identity.foundation/presentation-exchange/, accessed 23/01/18.

[Et16] ETSI: Electronic Signatures and Infrastructures (ESI), General Policy Requirements for Trust Service Providers. ETSI EN 319 401, 2016. etsi.org/deliver/etsi_en/319400_319499/319401/02.01.01_60/en_319401v020101p.pdf.

[Ga22] Global Assured Identity Network (GAIN): Proof of Concept Community Group. https://openid.net/2022/03/02/introducing-the-global-assured-identity-network-gain-proof-of-concept-community-group/, accessed 23/01/18.

[GV23] Gitlab Verifiable Credentials.info: Policy Registry. 2023. https://gitlab.com/verifiablecredentials.info/policy-registry, accessed 2023/01/18.

[Is11] Internet Society: DNSSec RFCs, Internet Society, https://www.internetsociety.org/resources/deploy360/2011/dnssec-rfcs-3/, accessed 23/01/18.

[Jj22] Jeyakumar, I. H J.; Chadwick, D. W.; Kubach, M.: A novel approach to establish trust in verifiable credential issuers in Self-sovereign identity ecosystems using TRAIN, In: Roßnagel, H., Schunck, C. (eds.), Open Identity Summit 2022, pp. 27-38.

[Ju21] Jurado, V. M.; Vila, X.; Kubach, M.; Johnson, I.; Solana, A.; Marangoni, M.: Applying assurance levels when issuing and verifying credentials using Trust Frameworks, In: Roßnagel, H., Schunck, C. H. (eds.), Open Identity Summit 2021, pp. 167–178.

[MR21] Market Research: ID Wallet: The German government had long known about IT security vulnerabilities, 21/10/30. https://marketresearchtelecast.com/id-wallet-the-german-government-had-long-known-about-it-security-vulnerabilities/190919/, accessed 23/01/18.

[Ng22] Next Generation SSI Standards: https://ngiatlantic.info/, accessed 23/01/18.

[Of22] OpenID: OpenID Connect Federation 1.0 - draft 26. 2022. https://openid.net/specs/openid-connect-federation-1_0.html, accessed 23/01/18.

[Op22] OpenID: OpenID Connect for Verifiable Presentations, 2022. https://openid.net/specs/openid-4-verifiable-presentations-1_0.html, accessed 23/01/18.

[Pm20] ESSIF-Lab Policy Management Project: https://gitlab.grnet.gr/essif-lab/business/policyman/policyman_project_summary, accessed 23/01/18.

[W321] W3C: Verifiable Credentials Data Model v1.1, W3C Recommendation 3/3/2022. https://www.w3.org/TR/vc-data-model/, accessed 23/01/18.

[Ya22] Yang, L.: LFPH Completes the Proof-of-Concept of Its GCCN Trust Registry Network. https://www.lfph.io/2022/04/19/lfph-completes-the-proof-of-concept-of-its-gccn-trust-registry-network/, accessed 23/01/18.

[Ye19] Yes: Ecosystem Architecture, https://www.yes.com/docs/ecosystem/1.0/index.html#_base_concepts, accessed 23/01/18.