H. Roßnagel et al. (Eds.): Open Identity Summit 2019,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2019 49

DNS-based Trust Scheme Publication and Discovery

LIGHTest's Trust Scheme Publication Authority

Georg Wagner[1], Sven Wagner[2], Stefan More[1] and Martin Hoffmann[3]

Abstract: Trust infrastructures are at the heart of a digital world. Within those trust infrastructures, trust schemes play an important role and often represent legal or organizational entities. Right now, trust schemes are published in the form of lists. Those lists enumerate all the trust services and their level of assurance. Trusted discovery only works if the URI of the trust list is known to the verifying party. In this paper, we introduce a Trust Scheme Publication Authority for arbitrary trust schemes. Our approach uses the Domain Name System (DNS) and its security extensions (DNSSEC) to publish discovery data securely.

Keywords: Trust Schemes; Publication; Discovery; eIDAS; LIGHT^{est}

1 Introduction

Trust infrastructures organize trust services, which provide digital services like identification and authentication of people, transactions, and more. Trust services enable a secure and trusted environment for all stakeholders.

Trust schemes are an essential part of many trust infrastructures. They often represent legal or organizational entities which regulate transactions, not only in the digital world. Therefore, a Trust Scheme is operated by a Trust Scheme Authority, and it comprises the organizational, regulatory/legal, and technical measures to assert trust-relevant attributes about enrolled entities in a given domain of trust. One core task of a trust scheme is to inform about the trust services inside its domain.

Right now, this is done by merely publishing an XML list, called trust list (or trust service status list). Such a trust list enumerates all trust services and, if present, their level or assurance. Also, it contains a lot of metadata of the trust services, like the company which operates them and their postal address, validity period and the used X.509 certificates. For the publication process, the existing and widely accepted standard for trust lists is ETSI TS 119 612 [Eu16].

[1] Technische Universität Graz, Institut für Angewandte Informationsverarbeitung und Kommunikationstechnologie, Inffeldgasse 16a, 8010 Graz, firstname.lastname@iaik.tugraz.at
[2] University of Stuttgart IAT, Institute of Human Factors and Technology Management, Allmandring 35, 70569 Stuttgart, Germany, firstname.name@iat.uni-stuttgart.de
[3] NLnet Labs, Science Park 400, 1098 XH Amsterdam, Netherlands firstname@nlnetlabs.nl

For example, the European Commission publishes the trust lists for the European trust scheme eIDAS on a daily basis on a publicly known server. This example shows that the discovery of trust schemes is a difficult task if we don't know all the addresses of all existing trust schemes. To simplify the discovery of trust lists, we propose a solution which uses the Domain Name System (DNS) for publication and discovery.

1.1 Current State of Trust Scheme Publication

The European Commission (EC) issues a trust list for the eIDAS trust scheme. This list contains pointers to the trust lists of the EU member states. The specification of this list of lists is given in the ETSI TS 119 612 standard [Eu16]. In the current publishing process of a trust list, a rigorous framework on how to publish a trust scheme is given to operators. Other countries use a different system to publish trust schemes or trust lists. For example, in the USA there exists neither a federal trust list nor a trust list for each state. Exceptions are states like the State of California [Of18], which publishes a list of trusted certification authorities for digital signatures for communications with public entities. However, this trust list is not available in a machine-readable format.

1.2 LIGHTest

LIGHT^{est} (*Lightweight Infrastructure for Global Heterogeneous Trust management in support of an open Ecosystem of Stakeholders and Trust schemes*) is a European research project (https://www.lightest.eu/). It thrives towards establishing a global trust infrastructure, also called LIGHT^{est}. A brief introduction is given in [BL16]. The project builds a system to automatically publish and discover trust schemes, as introduced in [Wa17]. Also, it enables the automated verification of transactions based on generic trust policies. In addition, the project supports established organizational structures by means of a flexible delegation mechanism. To unify the formats of delegations, LIGHT^{est} proposes a harmonized delegation data format, as shown by [WOM17]. To "go global", the project introduces the concept of automated translations between heterogeneous trust schemes.

The trust scheme publication process and its discovery is the topic of this paper and is explained in more detail in the following sections.

2 Introducing the Trust Scheme Publication Authority (TSPA)

The Trust Scheme Publication Authority (TSPA) is a central component of the LIGHT^{est} infrastructure. It enables the discovery and verification of trust scheme memberships for automatic trust verification, as shown by [Wa17]. The TSPA defines how Trust Schemes are published making use of the existing infrastructure and established

global trust anchor of the Domain Name System (DNS). For this purpose, the TSPA consists of two components: an off-the-shelf DNS nameserver with DNSSEC extension, and a Trust Scheme Provider server. The DNS nameserver is used for discovering the claim of Trust Scheme Association and the corresponding Trust List. The Trust Scheme Provider, implemented as an HTTPS component, provides this signed Trust List, which contains this required association.

Figure 1 provides an overview of the concept for trust scheme publishing in the TSPA. It shows the two components: DNS nameserver with the DNS records (left side), and Trust Scheme Provider (right side). The records on the DNS nameserver include data containers for Issuer and Trust Schemes. The containers for an Issuer are identified by an Issuer Name and include the Name of the associated Trust Scheme (*SchemeName*). Data Containers for a Trust Scheme are identified by a *SchemeName* and, if Level of Assurances (LoAs) are present, as *LevelName.SchemeName*. A Trust Scheme data container includes the Trust Scheme Provider Domain Name (*SchemeProviderName*). Besides, the data containers restrict the set of trusted certificates using the SMIMEA DNS resource record, as specified in [HS17]. Using SMIMEA enables to define the set of certificates which are accepted for signing the trust list. With this limitation of certificates, the signature of the Trust List can be verified.

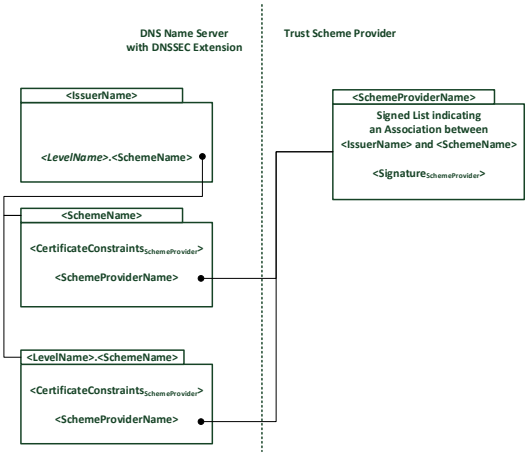

Fig. 1: Conceptional framework for the publication and discovery of Trust Schemes

To update nameservers, we introduce two components: TSPA and ZoneManager. The TSPA component itself acts as the endpoint for operators, which can be clients publishing trust schemes. It receives all relevant data via an HTTPS API to create the trust scheme. It can process links to existing trust schemes as well as full trust scheme data. In the first case, the TSPA component creates the DNS entries together with the ZoneManager. In the second case, the TSPA component stores the trust scheme data locally and creates the DNS entries together with the ZoneManager. The second component, the ZoneManager, acts as the

endpoint on the nameserver and modifies the zone data directly. It also ensures any zone data is properly signed using an existing DNSSEC setup. Thus, the ZoneManager must be installed on the nameserver. The ZoneManager's interface is only called from the TSPA component, and must never be called from the operator directly. This separation together with the interfaces is shown in Figure 2. Both components implement a RESTful API that is used by clients to publish the trust scheme information. This design also allows us to separate the server running the TSPA from the DNS server.

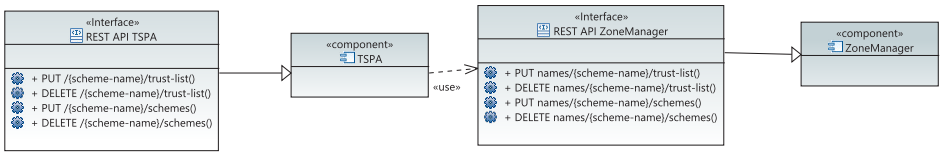

Fig. 2: Communication between components

3 Publishing Data using the TSPA

In order to be found in the DNS, each trust service and trust scheme taking part in LIGHT^{est} picks a domain name as its identifier. It can announce this name in its associated certificates, as described in Section 4. This domain name is used as the basis for querying trust-related information for the service or scheme. In order to allow publication of information for different topics, the actual domain name to be queried is formed by prepending a prefix to the name. For trust scheme publication, this prefix is the `_scheme._trust` zone.

The ZoneManager takes care of all DNS-related activities. It offers a generic interface that allows a TSPA to provide it with all relevant information necessary to maintain the DNS records that clients use to discover the information published by the TSPA.

There are three kinds of DNS resource records the ZoneManager publishes. First, a trust service can announce a claim to be a member of one or more trust schemes via *PTR* records that contain the domain name of the trust scheme in question. Second, a trust scheme publishes the location of its trust list in the form of a HTTPS URI in *URI* records. Finally, the trust scheme can publish restrictions for the certificate used to sign its trust list using the DANE protocol through *SMIMEA* records.

Hence, the TSPA provides the domain names of a trust scheme, the location of its trust list, and information regarding its signing certificates to the ZoneManager. The ZoneManager then generates the required DNS resource records from this information and adds them to the correct DNS zone for publication. It also signs that zone for DNSSEC validation by creating signatures for the newly created records. In addition, it creates any additional records and signatures necessary to provide a DNS zone that will be validated correctly by resolvers. The ZoneManager finally pushes the signed zone to a regular authoritative DNS nameserver for publication in the DNS.

3.1 API

Both the TSPA and the ZoneManager use a RESTful API to allow management of trust schemes and DNS zones. Using the interface the API provides, a user can create, modify, and delete trust schemes. The server components use the established HTTP request methods PUT, DELETE, and GET to create, update, and retrieve data.

Method	Endpoint	Parameter(s)
PUT	`/{scheme-name}/schemes`	schemes
DELETE	`/{scheme-name}/schemes`	-
PUT	`/{scheme-name}/trust-list`	trust list data or link
DELETE	`/{scheme-name}/trust-list`	-
GET	`/{scheme-name}/trust-list`	-

Tab. 1: TSPA Endpoints relative to the base URI (variable names in curly brackets)

Table 1 shows the endpoints for the TSPA server component. The first endpoint, the `PUT /{scheme-name}/schemes`, receives a list of scheme names. This list is then forwarded to the ZoneManager's endpoint to publish schemes. The second endpoint, the `DELETE /{scheme-name}/schemes` endpoint, deletes the given scheme. Therefore, it passes the list to the ZoneManager where the DNS entries are deleted. Also, it deletes any local files for those schemes. The third endpoint, `PUT /{scheme-name}/trust-list`, receives either the trust list data that should be published or a link to an existing trust list. If the received data is a trust list, then the data is stored locally on the TSPA servers hard disk and the link to this storage is forwarded to the ZoneManager to create the correct DNS entries. If the received data is a link, only the link is forwarded to the ZoneManager.

Access to the ZoneManager requires authentication. This authentication happens via bearer tokens. Users have to request those tokens before they can use the ZoneManager.

The ZoneManager provides the HTTP methods given in Table 2. All the parameters are encoded in JSON format and are considered as mandatory to be provided. Nevertheless, the parameters to access the ZoneManager are transmitted by the TSPA, and an ordinary user must not have access to the ZoneManager as a standalone component.

Method	Endpoint	Parameter(s)
PUT	`/names/{scheme-name}/trust-list`	URL, certificate
DELETE	`/names/{scheme-name}/trust-list`	-
PUT	`/names/{scheme-name}/schemes`	schemes
DELETE	`/names/{scheme-name}/schemes`	-

Tab. 2: ZoneManager Endpoints used by the TSPA

The first endpoint from Table 2, `PUT /names/{scheme-name}/trust-list`, requires two parameters. The first parameter is the `URL`, a simple string containing the URL of the

trust service status list of the given scheme. The second parameter is the `certificate`, a DaneCertificate object that describes the used certificate for the signed trust service status list. This last field is optional. If it is missing, no DANE [HS12] records are published.

The DaneCertificate contains four fields. The first field, the *usage* field, contains a string for the usage of the DANE record and can be one of the following *pkix-ta*, *pkix-ee*, *dane-ta*, *dane-ee*. If the field is left empty, the ZoneManager assumes *dane-ee*. The second field, the *selector* field, contains a string for the selector of the DANE record and can either be *cert* or *spki*. If the field is left empty the ZoneManager assumes *spki*. The third field, the *matching* field, contains a string for the matching field type of the DANE record and can either be *full*, *sha256*, or *sha512*. If the field is left emtpy, the ZoneManager assumes *sha256* as default. The fourth and last field, the *data* field, contains the data of the record, according to the other fields.

The third endpoint from Table 2, `PUT /names/{scheme-name}/schemes`, contains one parameter. This parameter is called *schemes* and contains a list of strings. Each string is a domain name identifying one trust scheme the service claims membership of.

The two `DELETE` endpoints do not require any parameters. The endpoints will remove the entries from the nameserver and thus delete the trust scheme.

4 Retrieving Data from the TSPA

Clients (in LIGHT^{est} they are called ATVs: Automatic Trust Verifiers) query the TSPA to learn about the trust status of a trust service. They do so if they want to verify a signed transaction they received. In specific, they want to verify whether the signer of the transaction is part of a trusted trust scheme. This is done by first discovering the API endpoint of the TSPA responsible for the respective scheme.

The signer's certificate is signed by a certificate authority (CA). This CA claims membership in a specific trust scheme. Such a claim is represented by a pointer in the CA's DNS zone. To retrieve this pointer, the client first needs to extract the CA's hostname from the transaction.

There are two ways of doing this: If the hostname is stored directly in the signer's certificate, the Issuer Alternative Name is used (defined in RFC5280 [Co08]). Otherwise, the hostname is contained in the certificate which issued the signer's certificate, using the Subject Alternative Name field, as defined in RFC5280 [Co08].

After extracting the CA's hostname, the client queries the CA's DNS for the *PTR* record of the *_scheme._trust* zone. This record contains a pointer to the DNS zone of the trust scheme of the CA, and thus also of the signer. The client then queries the trust scheme's DNS zone for the *URI* record. This record contains the HTTPS URI of the TSPA API, which can be used to retrieve the desired trust status information. In practice, the trust status information

retrieves if the CA is listed as a trust service in the trust list of the corresponding trust scheme.

All DNS records involved are authenticated using DNSSEC. Also, the trust status information is signed using a certificate which is pinned in the DNS using DANE. The certificate's fingerprint is stored in the SMIMEA record of the trust scheme's DNS zone.

5 Results

As an example on how the TSPA works, we chose Alice who is operating a TSPA at `tspa.example.com`. Her nameserver is located at `ns1.example.com`. She wants to publish a new trust scheme, called *TrustSchemeAlice*. Alice wants to create a trust scheme, which should contain the German *The-Trust GmbH* CA, which thereby becomes a TrustSchemeAlice qualified trust service provider. Alice has to create the correct trust list, which contains the provider, and send it to the correct TSPA endpoint, which is the trust-list endpoint as shown in Listing 1.

```
PUT https://tspa.example.com/api/v1/TrustSchemeAlice/trust-list
  TRUST LIST CONTENT
```

List. 1: Publication of the trust list (TRUST LIST CONTENT is used as a spaceholder)

With this simple call, Alice has created the correct discovery information for the trust-list XML file and has successfully created the DNS-entry for the discovery of the trust scheme. The TSPA endpoint has internally called ZoneManager and fed parts of Alice's data to ZoneManager which then created the DNS and DANE information. The TSPA has at the same time saved the trust-list information for TrustSchemeAlice on local storage.

```
...
_scheme._trust.tspa.example.com.    IN  URI
        1 1 https://tspa.example.com/api/v1/TrustSchemeAlice/trust-list
...
```

List. 2: DNS entry created by Alice for the publication of the trust scheme (for BIND9)

Now Alice is confident that everybody can discover TrustSchemeAlice.On the other side, we have Bob working for The-Trust GmbH. The-Trust GmbH now needs to claim that they are part of TrustSchemeAlice. They also operate a TSPA instance and can use the interface

to publish this claim. To do so, Bob needs to send the list of schemes his company is part of to the schemes endpoint, as shown in Listing 3.

```
PUT https://tspa.the-trust.eu/api/v1/TrustSchemeAlice/schemes
{ "example-ca.de.de",
  "tspa.example.com"
}
```

List. 3: Publication of the scheme claims

Bob has now successfully added his company to the TrustSchemeAlice, which can be successfully discovered.

```
_scheme._trust.the-trust.eu.  PTR _scheme._trust.example-ca.de.de
_scheme._trust.the-trust.eu.  PTR _scheme._trust.tspa.example.com
```

List. 4: DNS entries of the scheme claims (for BIND9)

Last but not least, Charlie has to verify a document. The document was signed by David using a certificate signed by The-Trust. To do so, Charlie starts the ATV on his machine. The ATV proceeds by analyzing the transaction and detects that it has a signature from David and that David's signature is from The-Trust. Now the ATV queries the The-Trust to find out in which trust schemes they are. This query is shown in Listing 5.

```
;; QUESTION SECTION:
;_scheme._trust.the-trust.eu.  IN  PTR

;; ANSWER SECTION:
_scheme._trust.the-trust.eu.   IN  PTR  _scheme._trust.example-ca.de.de.
_scheme._trust.the-trust.eu.   IN  PTR  _scheme._trust.tspa.example.com.
```

List. 5: First stage of the discovery of the trust scheme

Based on the query's result the ATV knows in which trust scheme The-Trust claims to be. Next, the ATV follows the pointers to the trust schemes' TSPA. Using the TSPA's trust-list, the ATV finally finds that The-Trust is part of TrustSchemeAlice, hosted at example.com. The result from this query is shown in Listing 6.

```
;; QUESTION SECTION:
;_scheme._trust.tspa.example.com.   IN  URI

;; ANSWER SECTION:
_scheme._trust.tspa.example.com.   IN  URI  1 1
        https://tspa.example.com/api/v1/TrustSchemeAlice/trust-list
```

List. 6: Second stage of the discovery for the trust scheme

The trust-list is a document signed by the TSPA. The key used to sign the document is likewise published in the DNS and authenticated using DNSSEC. Bob's ATV can, therefore, check whether the signature is valid by conducting another DNS query.

This example shows how Alice can operate her TSPA and add a new trust scheme. Also, it shows how Bob can add his CA into this trust scheme. Since the record published by Bob for The-Trust is only a claim used for discovery, Alice is still the authority for her TrustSchemeAlice. A malicious user could still publish this claim on their own, but TrustSchemeAlice's TSPA would not verify it. This example also illustrates how easy Charlie can find out in which trust scheme David is; by sending simple queries to the CA to find the TSPA and the correct trust scheme.

6 Conclusion

Trust scheme publication in the European Union is currently solved via the publication of a list where the trust scheme of a particular member state can be found. This approach requires a verifier to know where the trust scheme is saved at, and it would be more desirable if a CA can publish a membership claim, that can be verified during the verification of a transaction. In this paper, we have shown the technical solution to solve this problem. We described the external API of the involved components, and how they can be used to publish trust scheme information. We also have shown how we can use DNS to make trust scheme membership claims discoverable by a verifier in an automated way.

Acknowledgments

The LIGHT^{est} project is partially funded by the European Commission as an Innovation Act as part of the Horizon2020 program under grant agreement number 700321.

Bibliography

[BL16] Bruegger, B. P.; Lipp, P.: LIGHT^{est} - A Lightweight Infrastructure for Global Heterogeneous Trust Management. In: Open Identity Summit 2016, 13.-14. October 2016, Rome, Italy. Pp. 15–26, 2016.

[Co08] Cooper, D.; Santesson, S.; Farrell, S.; Boeyen, S.; Housley, R.; Polk, W.: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, RFC 5280, RFC Editor, May 2008.

[Eu16] European Telecommunications Standards Institute: Electronic Signatures and Infrastructures (ESI); Trust Lists, EN 119 612 V2.2.1, ETSI, Apr. 2016.

[HS12] Hoffman, P.; Schlyter, J.: The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA, RFC 6698, RFC Editor, Aug. 2012.

[HS17] Hoffman, P.; Schlyter, J.: Using Secure DNS to Associate Certificates with Domain Names for S/MIME, RFC 8162, RFC Editor, May 2017.

[Of18] Office of the California Secretary of State: Approved List of Digital Signature Certification Authorities, `https://www.sos.ca.gov/administration/regulations/current-regulations/technology/digital-signatures/approved-certification-authorities/`, Accessed: 10 2018.

[Wa17] Wagner, S.; Kurowski, S.; Laufs, U.; Roßnagel, H.: A Mechanism for Discovery and Verification of Trust Scheme Memberships: The Lightest Reference Architecture. In (Fritsch, L.; Roßnagel, H.; Hühnlein, D., eds.): Open Identity Summit 2017. Gesellschaft für Informatik, Bonn, 2017.

[WOM17] Wagner, G.; Omolola, O.; More, S.: Harmonizing Delegation Data Formats. In (Fritsch, L.; Roßnagel, H.; Hühnlein, D., eds.): Open Identity Summit 2017. Gesellschaft für Informatik, Bonn, 2017.