Trust Scheme Interoperability: Connecting Heterogeneous Trust Schemes

Stefan More
stefan.more@iaik.tugraz.at
Graz University of Technology
and Secure Information Technology Center Austria (A-SIT)
Graz, Austria

ABSTRACT

The growing interconnectedness of computer systems has led to the need for a flexible approach to trust management. Many countries operate trust schemes to enable the automated assessment of the trustworthiness of information. But this assessment remains a challenge if the information was issued in a foreign trust scheme. An issue is the lack of a root of trust shared between the trust schemes. Other challenges are the heterogeneity of trust models used by entities operating in different legal and cultural environments.

In this paper, we present a novel approach to facilitate the interoperability between different trust schemes. In our approach, trust scheme operators take legal agreements that exist between two countries and publish them as a machine-readable *trust recognition*. Additionally, a scheme operator codifies the rules for trust recognition of the other scheme in the form of a *trust translation*. Using this information, a trust verifier maps trust data from the other scheme into its own scheme. This allows a verifier to automatically process transactions from other trust schemes in a trustworthy way.

CCS CONCEPTS

• **Security and privacy** → **Trust frameworks**; **Authentication**; *Key management*; *Access control*; • **Applied computing** → *Electronic commerce.*

KEYWORDS

Trust, Trust Schemes, Interoperability, Global Trust Management

ACM Reference Format:
Stefan More. 2023. Trust Scheme Interoperability: Connecting Heterogeneous Trust Schemes. In *The 18th International Conference on Availability, Reliability and Security (ARES 2023), August 29–September 01, 2023, Benevento, Italy*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3600160.3605046

1 INTRODUCTION

Trustworthy identification is a crucial concept in electronic transactions. For example, to establish liability in a business transaction, a service provider (SP) needs to determine the identity of their business partner. Trust schemes are used to support SPs in assessing the trustworthiness of the identifier of some entity. Further, trust schemes are also used to establish trust in electronic transactions issued by such entities.

To do so, the SP uses an automated trust verifier. The trust verification process starts when the service provider receives a signed attestation.

Challenge 1: To know if the attestation has any legal value, the verifier needs to check if it was issued by a qualified issuer (i.e., if the issuer is accredited by this trust scheme). This is a challenge if the issuer is qualified in a different trust scheme than the one the verifier trusts. Since there is no common root of trust between the schemes, the trust scheme authorizing the issuer is not known to the verifier. Thus, the verifier has no information about the trustworthiness of that issuer. Consequentially, the attestation has no value for the verifier.

In some cases there is a legal relationship between two trust schemes – e.g., a treaty between two countries, or some other form of recognition. In that case, the verifier needs to learn about this relationship. This allows the verifier to assess the trustworthiness of the foreign issuer, but it is currently a manual process that involves non-machine-readable information.

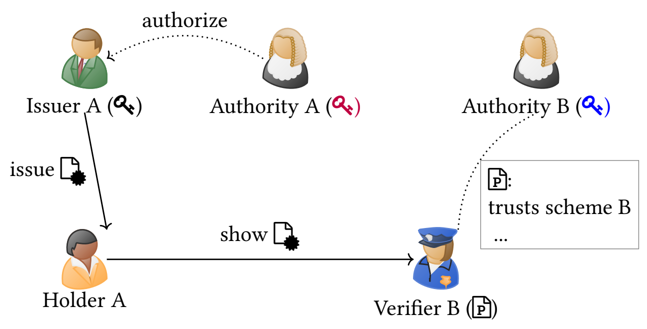

Figure 1: Global trust management actors and their relationship. Verifier B does not trust Issuer A authorized in the untrusted trust scheme A.

Challenge 2: Additionally, the verifier needs to assess the confidence it can have in the information provided in the attestation. For a certificate that binds a cryptographic key to a (legal) identity, this means how sure the verifier can be about the identity of the holder that signed the incoming document. Depending on the trust scheme, the trustworthiness of the identity is determined by factors such as how the holder needed to prove their identity (at enrollment), e.g., remote or in-person.

ARES 2023, August 29–September 01, 2023, Benevento, Italy
© 2023 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0772-8/23/08.
https://doi.org/10.1145/3600160.3605046

Since different legislations and use cases require a different quality of identity, the verifier needs to ensure that the provided attestation meets their requirements. To do so, the verifier acquires the level of trust of the attestation, and compares it with their local trust policy. The structure and semantics of this level trust depend on the individual trust scheme. For their local trust scheme, the verifier is either aware of the trust levels of attestations issued by qualified issuers. Or there is no need to know, since the trust scheme already ensures compliance to the local laws which the verifier needs to follow. But, this is not the case if the attestation was issued in a different trust scheme.

1.1 Contributions

In this paper we solve these challenges with a *automated trust recognition* system. We enable verifiers to automatically assess the trustworthiness of attestation issuers from other trust schemes. Additionally, we support the *translation* of trust data between trust schemes of different types. This allows verifiers to automatically establish trust in transactions coming from other trust schemes.

(C1) Trust Recognition: We enable trust scheme operators who recognize other schemes as equivalent to publish this recognition in form of a *trust recognition*. The trust recognition can later be retrieved by a verifier. Using this recognition, the verifier can automatically authenticate the issuer of information, even if this information was issued in a different trust scheme.

(C2) Trust Translation: For situations where trust schemes are not equivalent, we provide a system to translate information about the quality of identities between schemes. We do so by attaching *trust translation data* to the published recognition. This translation data can be used by a verifier to automatically map trust data into a trust scheme type and semantics it understands. This enables a verifier to work with trust information from trust schemes with a different understanding of trustworthiness.

(C3) Reference Implementation: We provide an implementation to demonstrate the feasibility of our approach. Our implementation builds on the architecture of the Horizon 2020 project LIGHTest. We extend LIGHTest's existing trust scheme publication and verification system with support for trust recognitions and translations. Further, we enrich the trust policy system TPL with functionalities to define translation rules. We also show how trust translations can be encoded in machine readable form.

1.1.1 Outline. The rest of this paper is structured as follows: Section 2 and Section 3 describe concepts relevant for our paper, with a focus on trust schemes. In Section 4 we introduce our trust recognition and translation concept, and in Section 5 we describe our reference implementation. We conclude the paper in Section 6 with a discussion of several aspects of our approach, and list ideas for future work.

2 BACKGROUND

2.1 Preliminaries

2.1.1 DNS and DNSSEC. The Domain Name System (DNS) is a critical component of the internet that is commonly used to translate human-readable domain names into IP addresses [17]. It operates on key-value pairs of strings, so it can also be used to translate to other types of data, such as X.509 certificates [13]. DNS uses a hierarchical structure to manage names, delegating the management of specific domain names to regional registries. For example, the German registry DENIC manages *.de* domains, while EURid manages *.eu* domains. However, the DNS system is vulnerable to various types of attacks, such as DNS spoofing and cache poisoning, that can redirect users to malicious websites. DNS Security Extensions (DNSSEC) aims to secure the DNS system by providing data integrity and authentication [2]. DNSSEC uses cryptographic signatures to sign the DNS records. Similar to the hierarchical management structure of DNS, DNSSEC uses a top-down trust model with name subordination [23].

2.1.2 LIGHTest. The EU LIGHTest project[1] developed a lightweight, global trust infrastructure, which enables automatic validation of trust based on the policy of a verifier [3, 24, 27]. LIGHTest uses the Internet's DNS with its existing global infrastructure, organization, governance and security standards. LIGHTest uses DNS to publish information protected by DNSSEC. It enables trust scheme operators to publish information (trust data) about their trust schemes. Further, LIGHTest provides an automated trust verification system (ATV) which is capable of automated discovering and retrieval of the published trust information [26]. To enable verifiers to define which scheme they trust, LIGHTest uses the trust policy system TPL [18, 19], which was recently extended with support for SSI and privacy-features [1, 21] This paper builds on ideas of the LIGHTest project.

2.1.3 PKI and eIDAS. The Public-key Infrastructure (PKI) based on X.509 certificates (PKIX) – called *web PKI* – is the predominant authentication model on the internet [6, 15]. In this hierarchical model, the list of Certificate Authorities (CAs) – the trust store – forms the root for trust chains binding an entity's identifier (e.g., domain name) to its cryptographic keys. CAs are only considered trustworthy if a client trusts them directly, which is usually achieved by bundling a set of root certificates with the operating system or web browser. While the current web PKI serves its purpose, it is limited to establishing the binding between a domain name and a cryptographic key.

The EU's eIDAS regulation and technical specification [8] establishes a trust infrastructure between the EU's member states along with legal liabilities but is currently limited to Europe. eIDAS uses its own PKI, where each EU member state defines a Trust Status List (TSL), i.e., a list of all trusted certificate authorities [25]. The locations of those lists are in turn published by the European commission in the form of a List of Trust Lists (LOTL), forming eIDAS' root of trust.[2]

2.2 Related Work

Article 14 of the eIDAS regulation provides the legal basis for a framework for the recognition of trust services operated by trust service providers from a country outside of the EU (the "3rd country"). This is relevant if the EU and the 3rd country have reached a legal agreement about that recognition. Article 14 covers only trust services (e.g., signatures and seals), but not electronic identities. A

[1]https://www.lightest.eu, accessed on 2023-02-13
[2]https://ec.europa.eu/tools/lotl/eu-lotl.xml

pre-requirement for mutual recognition under Article 14 is that the 3rd country publishes a trusted list , ideally following the same standard as the EU's lists (ETSI TS 119 612). If recognized, a pointer to the 3rd country list would be included in the EU's LOTL, next to pointers to EU member states' lists. ETSI TR 103 684 a technical report that studies existing trust schemes around the world and their possible mutual recognition in the EU [7].

Wagner et al. propose a unified data model by consolidating the data models of nine existing trust schemes [27]. It can be used to describe trust schemes in a unified way, and to compare different schemes to simplify mutual recognition.

The Futuretrust project [16] proposed the concept of a global Trust Status List (gTSL) [9] to extend the EU's TSLs to institutions from outside the EU. This gTSL is hosted on a Distributed Ledger; access is provided using a smart contract and write access is limited to governmental authorities.

3 ON TRUST SCHEMES

Trust schemes help entities to automatically make trust decisions about electronic transactions they receive. They consist of technical standards, legal regulations, and infrastructure.

3.1 Actors

On a technical level, trust schemes are used to authenticate **attestations**. An attestation is a signed document containing some claims about the holder. For example, the document is called certificate if it attests the binding between the user's name and their public key (e.g., X.509 certificate). Or, it is called assertion, if it contains identity attributes about the holder, directly used to authenticate at the service (e.g., SAML assertions, W3C Verifiable Credential).

A simplified trust scheme consists of the following actors, also shown in Figure 1:

- **Holder:** the user in possession of some **attestation**, which they present to the verifier or use to sign some data.
- **Verifier**: a component that is typically operated by each service provider. It assesses the authenticity of incoming attestation by checking if the attestation was issued by a trustworthy issuer. A issuer is considered trustworthy, if it is authorized by a authority trusted by the verifier. Additionally, the verifier checks if the attestation values match other business logic specific rules. To do so, it evaluates the values of the attestation against some local policy.
- **Issuer:** the entity that attests attributes of other entities (i.e., the holder) after verifying them. To do so, the issuer signs the holder's claims. This results in a attestation, which the issuer sends to the holder. In the eIDAS qualified signature use case, a issuer is called "trust service".
- **Authority:** the entity that authorizes some issuer to issue attestations, commonly the operator of a trust scheme. Depending on the trust model, authorities are either directly authorized, or themselves receive their authorization from some other authority. The authority on top of this authorization chain is called *root of trust* and must be trusted by all entities in a trust scheme.

3.2 Types of trust schemes

A trust scheme is, among other things, characterized by the requirements on how to acquire a qualified attestation. Wagner et al. describe three types of trust schemes [27]:

Boolean trust schemes only separate between certificates that fulfill all requirements, and those that do not. In that case, the sole existence of a (qualified) signature on the attestation is enough to certify that the attestation fulfills *all* requirements of the trust scheme.

Ordinal trust schemes differentiate between different types of attestations depending on the quality of trust that can be achieved. This differentiation is represented in the form of a numeric level, such as the level of assurance (LoA) a verifier can have about the received information. By stating that a attestation has a certain level, a issuer certifies that the specific requirements for that level are fulfilled. A verifier can then specify the required level in their policy.

Tuple based trust schemes are an even more flexible approach to differentiate the level of trust. This is achieved by directly publishing the full list of requirements of the trust scheme, and specify which are fulfilled by a certain attestation in the form of key-value pairs. In the simplest form, each value is a boolean, but can also be a ordinal level, or even a string.

For ordinal and tuple based schemes, the level or tuples can be part of the attestation document, e.g., the certificate or assertion. In some trust schemes, the information is instead part of the issuer's authorization. For example, in a scheme using trust lists (e.g., eIDAS), it can be part of the issuer's trust list entry. Thus, to assess the trustworthiness of some information, the verifier first needs to retrieve the authorization of the issuer.

4 TRUST RECOGNITION & TRANSLATION

In this section we introduce the concept of our approach.

When a verifier receives an attestation as part of some electronic transaction, it first authenticates the issuer of this attestation. To do so, it uses its local trust store to check if this issuer was authorized by the local trust scheme authority. If this check succeeds, the verifier can then directly check whether the attestation satisfies the service's trust policy. But: this authentication check fails if the attestation was signed by an issuer that is not authorized by one of the authorities in this trust store.

In that case, the verifier checks if there is a **trust recognition** of the issuer's trust scheme by the verifier's trust scheme. If this recognition exists, the verifier retrieves it, and checks if the issuer is qualified in the other trust scheme. Further, if the schemes are not equivalent, the verifier retrieves the corresponding **trust translation** and translates the trust data into its own scheme.

4.1 Roles of trust schemes

We name the roles of the involved trust schemes from the perspective of the translation process.

- **Source scheme** is the trust scheme of the holder. It is responsible for defining the authorities that in turn authorize issuers qualified in that scheme. One of these issuers then issues the attestation we are going to verify. The verifier does not trust the source scheme directly.

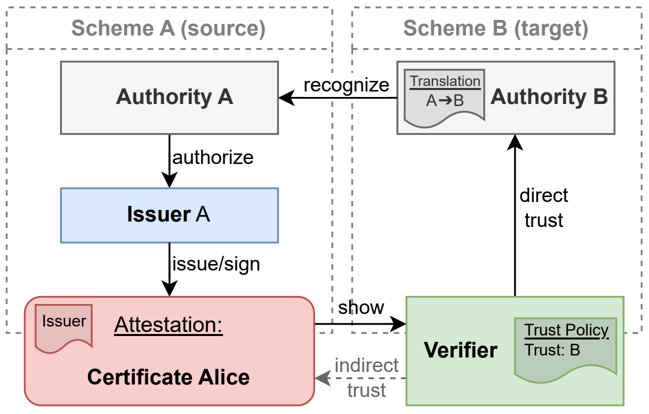

Figure 2: Example trust path between two trust schemes. Trust scheme B recognizes trust scheme A and also publishes a translation from A to B. The verifier in scheme B can use this trust path to authenticate a attestation issued in scheme A, and execute the translation to understand trust data from scheme A.

- **Target scheme** is the trust scheme of the verifier. The verifier trusts the target scheme, and uses its infrastructure to retrieve the list of qualified issuers. Additionally, the target scheme is responsible for publishing the trust recognition and translation data.

For example, if country B – operating the target scheme – recognizes the trust scheme of some other country A, it publishes a trust translation *from* that trust scheme A to its own trust scheme B. From the verifier's perspective, the "target scheme" is the home scheme, and the "source scheme" is the foreign scheme.

4.2 Trust Recognition

If a trust scheme operator recognizes some other trust scheme, it publishes this statement in form of a trust recognition. A recognition is published by *the target* of the recognition, since it represents the authority (root of trust) for its own scheme. It is thus already trusted by all verifiers operating in its scheme. In contrast, the authority of the source scheme is not trusted by a verifier, thus the verifier does not trust a recognition published by that other scheme.

For example, if scheme B recognizes scheme A, this is a recognition (and later translation) from scheme A to scheme B ($A \rightarrow B$). This recognition is hence published by scheme B, as illustrated in Figure 2.

The recognition of a trust scheme by another trust scheme leads to a trust model with cross certifications [10, 23], as both scheme authorities are the respective root of trust in their scheme. This removes the need for a root of trust shared by all trust schemes.

If the two schemes are equivalent, the sole existence of the recognition is already enough for a verifier in the target scheme to work with an attestation issued in the source scheme. It can use the now-trusted authorities of the other scheme to authenticate the attestation's issuer. If the schemes are not equivalent, the verifier first needs to translate the trust data of the attestation.

4.3 Trust Data

The trust data of an attestation specifies the requirements for trust scheme entry. Those are the requirements that the holder needed to fulfill to receive the attestation. Trust data is either directly attached to that attestation, or can be retrieved by the verifier from the trust scheme, e.g., an authority. The structure of the trust data depends on the type of the trust scheme (see Section 3). Trust data is encoded in different ways depending on the type of scheme and the relevant trust scheme requirements. Also, the semantics of trust data depends on the scheme. But, a verifier needs the trust data in an encoding and with semantics it understands. Therefore trust data is the subject of the trust translation process.

4.4 Translation Data

The trust translation is a function to translate trust data between two schemes. The goal is to translate trust data of the source scheme's type and semantics to the target scheme's type and semantics.

The operator of the target scheme is trusted by all verifiers inside that scheme. Since it manages legal recognition between countries, it also defines this trust data mapping, and encodes it in form of trust translation data. The trust scheme then publishes this translation data alongside the corresponding recognition. As also shown in Figure 3, the input to the `translate` function are trust data (issued by the source scheme), and the corresponding translation data. The output of the function is trust data (understood by the target scheme).

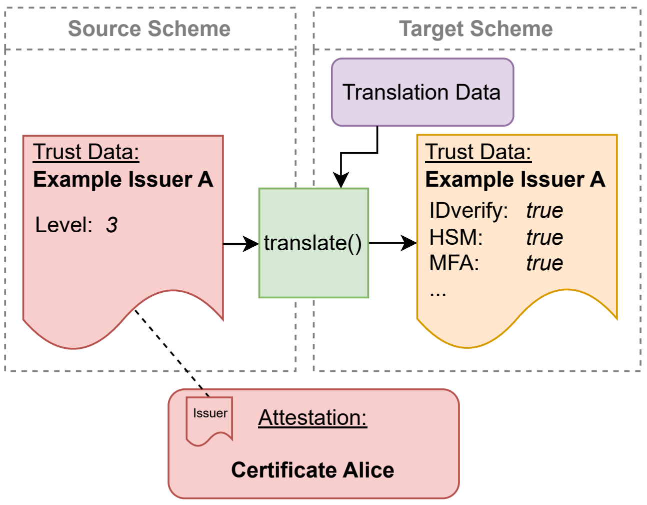

Figure 3: Example trust data translation process. The trust data describing Alice's attestation is translated from a ordinal scheme into a tuple based scheme.

For publication, the translation function is formulated as a table. Table 1 illustrates a trust translation data table for a simple translation. The first column lists all possible trust scheme requirement combinations as trust data for the incoming attestation. In the case of the example, the source scheme is a ordinal scheme, so this list consists of all possible levels. The second column maps this levels to a set of tuples of the target scheme.

In the Table 1 example, the highest level 3 requires that an issuer verifies the holder's legal identity (*IDverify*), and that the holder uses a HSM to store the attestation key material (*HSM*), and has performed multi-factor authentication (*MFA*). An example translation executing this translation data is shown in Figure 3. This table is then encoded in machine-readable form (e.g., as XML or JSON) and published by the trust scheme.

Table 1: Example trust translation data for a translation from a ordinal scheme into a simple tuple based scheme.

Source Scheme	Target Scheme
Level: 1	IDverify: true
	HSM: false
	MFA: false
Level: 2	IDverify: true
	HSM: false
	MFA: true
Level: 3	IDverify: true
	HSM: true
	MFA: true
...	...

There is only a mapping to some combinations of tuples, since other combinations of requirements are not possible in the source scheme. For example, in the source scheme there are no attestations issued without identity verification. The table for the other direction of the translation (formulated and published by source scheme) might look different. And, multiple combinations of tuples could be mapped to the same level.

4.4.1 Special case: equivalent schemes. If the semantic meaning of all levels/tuples is the same in both schemes, the schemes can be considered *equivalent*. In that case the trust recognition can be published without trust translation data. But: translation of trust data between two schemes of the same type is not automatically an identity map, as different levels could have different semantics in each scheme.

4.5 Translation Process

Our trust translation approach can be split into three phases: (1) Initially, two trust scheme operators negotiate a legal agreement and compare their trust schemes. They also agree on a translation between their schemes and formulate this translation of trust data. (2) The trust scheme operators encode the trust recognition and translation in machine readable form and publish it. (3) Later, a verifier receives a electronic transaction. To be able to verify it, the verifier retrieves the trust recognition and translation data, and executes the translation.

Phase 1) Legal Agreement: The preparation of a trust recognition starts with a negotiation phase between the two trust scheme operators (e.g., two governments). The details of this phase depend on legal circumstances, but might include a feasibility study of the planned recognition. Further, a self assessment of the individual trust schemes could be conducted. Next, the scheme operators need to ensure technical compatibility between their schemes. While agreeing on a technical standard to encode attestations is preferable, we discuss alternatives to this limitation in Section 6.1.

Mapping Trust Scheme Data: To formulate a trust translation, the scheme operators then compare the characteristics of their schemes. For example, they both map the characteristics of their schemes to a unified data model. A possible data model for mapping trust schemes was established by Wagner et al. [27]. Or, they use a process like the eIDAS Article 14 Mutual Recognition check-list, which maps the 3rd party requirements to the eIDAS requirements [4]. The result of this process is a mapping between the two schemes. If there is a direct mapping between all tuples of the two schemes (i.e., the mapping is the identity map), they can be considered equivalent. In that case, only the recognition itself – and no translation data – is needed. If the schemes are not equivalent, a mapping between different trust data is created. The scheme operators then individually formulate their mapping as trust translation data.

Mutual or Unilateral Recognition: While these steps describe a mutual recognition of two schemes, it is also possible that only one scheme recognizes the other. But, a trust recognition is only possible if the operator of the target scheme can define a trust data mapping. In a similar manner, it is possible that the schemes mutually recognize each other, but use different translation rules. Thus, the outcome of the first phase might differ between the two schemes, but the following phases are performed in the same way.

Phase 2) Publication: After formulating their trust recognition, the trust scheme operators publish it. They also encode and publish the trust translation data. Each trust scheme individually performs this phase. The encoded translation data is a machine readable document in a format that all involved trust schemes understand. The operator then signs the document to ensure its integrity and authenticity. It uses the same public key it uses for signing the trust scheme's list of authorities. Next, the trust scheme operator publishes the recognition and translation data at a location known by all entities operating in the trust scheme.

Phase 3) Verification & Trust Translation:

The verification process starts when a service provider receives an attestation as part of a electronic transaction.

Trust Store and Local Trust: To ensure the authenticity of the attestation, the verifier first extracts the information about its issuer. It then loads its local trust store. The trust store was previously retrieved from the trust scheme authority and contains the certificates of all issuers that are qualified in the trust scheme. The trust store is typically signed by the trust scheme authority, so the verifier can assess its integrity. For example, in eIDAS the trust store is generated by retrieving the trust lists of all EU member states and – after authenticating the list – extracting the certificates of qualified issuers from it. The verifier then checks if the issuer of the incoming attestation is part of the trust store. If it can find the issuer certificate, it proceeds with the verification of the signatures of the trust chain and executes the service provider's trust policy. Otherwise, the verifier proceeds to the trust recognition process.

Discover Trust Recognition: First, the verifier extracts the trust scheme membership information from the issuer information. The trust scheme membership information is the identifier of a trust scheme of which the issuer claims to be a member. The verifier then uses this information to query its own trust scheme authority for a trust recognition to the specified scheme. If such a recognition exists, the verifier also retrieves the corresponding trust translation data from the authority. It then authenticates this data using the public key of the trust scheme. Using this recognition, the verifier establishes trust into the source scheme's authority.

Execute Trust Translation: Next, the verifier retrieves the trust data of the attestation, either directly from the attestation (see Section 4.3), or using the (now trusted) source scheme. This trust data uses the source scheme's format, and also follows it semantics. Thus, the verifier cannot directly use it in its trust policy, and needs to translate it. This translation is performed by applying the translate function on the trust data with the previously retrieved trust translation data. The translation process is a simple table lookup: the verifier iterates over the list of trust-recognitions and compare each source scheme specification (i.e., the left side of the translation data table) with the trust data at hand. If it finds a match, it returns the corresponding target scheme specification (i.e., the right side of the table).

The result of this translation process is trust data in the format of the target scheme. Since the verifier now understands the data, it can load its trust policy and executes it on the trust data, and thereby assess the trustworthiness of the attestation.

5 IMPLEMENTATION

We implement our trust recognition approach to study it further. This implementation also demonstrates the feasibility of our trust translation approach.[3]

Our implementation is based on the trust infrastructure established by the LIGHTest project (see Section 2.1.2). We extend LIGHTest's trust scheme publication system to enable the publication of recognition of other schemes and trust translations. Further, we extend the automated trust verification tool (ATV) to automatically discover, retrieve, and authenticate trust recognitions. The legal agreement phase (Phase 1) is out of the scope of our implementation.

5.1 Publication (Phase 2)

In phase 1, a trust scheme reached an agreement with some other trust scheme, and encodes this agreement in form of a mapping of trust data. This mapping is then encoded as a machine-readable trust translation data table. In our implementation, we serialize this table as a JSON file. This file contains a list of `trust-recognition` JSON objects, where each line in the table is represented by one object in the list. An example trust translation data encoded as JSON is shown in Listing 1.

This JSON file is then signed, and published by the target scheme using the LIGHTest infrastructure. As a result, the translation data is reachable at a domain name constructed by combining the DNS identifiers of both scheme (i.e., *source-scheme._translation.target-scheme*, e.g., *pof.org._translation.eidas.eu*).

At this location, the (target) scheme publishes a PTR record pointing to the DNS identifier of the source scheme. Additionally, it publishes a URI record with a HTTP link to the trust translation data document.[4] The URI record is then signed using the DNSSEC key of the scheme's zone. In LIGHTest, the signing key used by the (target) scheme to sign the translation data itself is already published in a TLSA record at, e.g., *_scheme._trust.eidas.eu* [14].

[3]github.com/H2020LIGHTest/PumpkinSeedOilDemo

5.2 Verification (Phase 3)

In our implementation, the service provider uses the LIGHTest verification tool ATV to authenticate attestations. Additionally, the SP uses the trust policy system TPL to codify its rules about trust conditions, such as which trust scheme it considers trustworthy. The trusted scheme is specified by its DNS identifier. In the trust policy the SP also activates trust translations. An example TPL trust policy with enabled trust translation is shown in Listing 2 in the Appendix.

During authentication of an attestation, the ATV starts the TPL interpreter with the SP's trust policy. It then extracts the issuer's trust scheme membership claim (represented by the scheme's DNS name). If this claim matches the scheme identifier defined in the policy, the verification can proceed with the local scheme. Otherwise, a trust recognition is needed.

In that case, the ATV queries the recognition domain name constructed from the DNS identifiers of its trusted scheme (target scheme) and the issuer's claim (source scheme), i.e., a PTR record at *source-scheme._translation.target-scheme.* If a recognition from that scheme exists, the DNS returns the the trust recognition. If the schemes are not equivalent, a URI record at the same location contains a link to the translation data (the signed JSON published in phase 2). The ATV also verifies the DNSSEC signatures, and the signature on the JSON. Iff the signature verification succeeded, the foreign scheme is trusted. The ATV then queries the now trusted authority of the foreign scheme for the issuer's certificate. If the authority returns the correct certificate, the issuer's membership claim is confirmed. Next, the ATV uses the retrieved issuer certificate to verify the signature on the attestation.

If the schemes are not equivalent (if translation data exists on the recognition), the ATV retrieves the issuer's trust data from the scheme's authority. It then executes the translation on the trust data. The result is trust data which the TPL interpreter can use.

The ATV concludes the process by returning the translated trust data to the policy interpreter. The interpreter uses the data to check if fulfills the policy, and if other business-logic specific rules are fulfilled. It does so without the need for knowledge about the other scheme.

5.3 Example Trust Translation

Listing 1 shows an example of trust translation data for a translation from an ordinal scheme (POF federation) into a simple tuple based scheme (demo-eIDAS). This translation would be published by the operator of the demo-eIDAS scheme (the demo-EC) at their TSPA/TTA, e.g., ato *pof-federation.example.com._translation.demo-eidas.ec.europa.eu.*

[4]If the schemes are equivalent, no URI record is needed. In that case, NSEC3 records are used to prove the nonexistence of the URI record. This prevents an adversary from hiding a translation, which would "promote" a source scheme to be equivalent even if it is not.

```
{
"trust-recognitions": [
 {
  "source": {
   "name": "pof-federation.example.com",
   "provider": "POF",
   "level": "3"
  },

  "target": {
   "name": "demo-eidas.ec.europa.eu",
   "provider": "EC",
   "params": [
    {
     "value": "ServiceTypeIdentifier",
     "name": "ETSI/CA/QC"
    }
   ]
  },

  "name": "pof-to-eidas-esig",
  "creation-date": "2023-02-13",
  "activation-date": "2023-02-23",
  "status": "active",
  "leaving-date": "2023-07-13"
 },
  ...
]
}
```

Listing 1: Example (unsigned) trust translation data.

6 DISCUSSION

6.1 Attestation Format

In its basic form, our approach assumes that both trust schemes use the same standard for encoding attestations. This is required because the verifier needs to be able to parse and understand the attestation and the certificate of the signer. Since X.509 is an established standard that is commonly used by trust schemes, this is a realistic assumption [15, 27]. If another standard is used, but both schemes are using the same format (e.g., W3C verifiable credentials), our approach can be applied as well [22].

6.1.1 Future Work: Translation of the Attestation. If the one scheme uses a different attestation format than the other, there is a lack of understanding of the attestation, which is a issue for trust verification. In that case, the verifier needs to translate the attestation as well. To do so, it requires translation information about how to transform the attestation from one format into the other [20]. Additionally, the verifier requires information about how to verify the signature on the attestation in its original format, since transforming it obviously breaks the signature. This attestation translation information could be provided by the same party that publishes the trust translation information.

6.2 Transitive Trust Recognitions?

Trust transitivity is a property of trust where an entity believes in the trust assumptions of another entity about a third entity (if A trusts B and B trusts C, then A trusts C). In the case of trust recognitions this means that a verifier would not only trust a recognized scheme, but also all other schemes recognized by that scheme. While transitive recognitions – and even translations – are technically possible with our approach, we note that trust recognition transitivity is more complicated in terms of recognition agreements. Just following along a transitive recognition can lead to unintentional trust in an issuer, i.e., trust that is not covered by the recognition agreement [5].

6.3 Governance

Any country that already operates their own trust scheme and is interested in implementing such a trust recognition approach needs to establish a legal framework for trust scheme recognitions. In addition to the mapping of trust data between schemes, rules for the operation of trust infrastructure are needed. These rules also need to define the liability of involved parties for damage caused intentionally or negligently.[5]

6.3.1 Future Work: Legal Framework. Our reference implementation builds on the LIGHTest system, which uses the DNS for trust data publication. While the DNS has a governance structure, it only defines rules and processes about the management and publication of standard DNS records. Using the DNS as trust scheme root of trust has different legal implications. It thus requires that the government of the trust scheme establishes a legal framework that defines liability of the country registry (ccTLD) to establish trust in the ccTLD.

For the time until interested governments establish such a legal framework, the LIGHTest project proposes an alternative framework directly between the involved parties [11, 12].

We note that it is not always necessary that a government implements our approach for to provide value. For example, even if the government of Fantasyland is currently not interested in recognizing the eIDAS trust schemes, the business trade union of Fantasyland could nevertheless use their own trust scheme to publish a trust recognition to eIDAS. Since this recognition is not published by the government, it does not have legal value. But, it still reduces the transaction overhead between the two schemes by supplying local business with trustworthy information.

Conclusions

In this paper, we presented an approach for trust management in a heterogeneous environment.

We enable trust scheme operators who recognize other schemes as equivalent to publish this recognition in the form of a *trust recognition*. The trust recognition can later be retrieved by a verifier. Using this recognition, the verifier can automatically authenticate

[5]cf. eIDAS articles 11 and 13 [8]

the issuer of information, even if this information was issued in a different trust scheme.

For situations where trust schemes are not equivalent, we provide a system to translate information about the quality of identities between schemes. We do so by attaching *trust translation data* to the published recognition. This translation data can be used by a verifier to automatically map trust data into a trust scheme type and semantics it understands. This enables a verifier to seamlessly work with trust information from trust schemes with a different understanding of trustworthiness.

We also presented a reference implementation of our approach and discussed severally of its aspects.

ACKNOWLEDGMENTS

This work was supported by the European Union's Horizon 2020 research and innovation programme under grant agreement № 101020416 (ERATOSTHENES). We build on results of the LIGHTest project – we thank all members of the LIGHTest consortium. Further, we are grateful to Jakob Heher for inspirational discussions and proofreading the manuscript.

REFERENCES

[1] Lukas Alber, Stefan More, Sebastian Mödersheim, and Anders Schlichtkrull. 2021. Adapting the TPL Trust Policy Language for a Self-Sovereign Identity World. In *Open Identity Summit (LNI, Vol. P-312)*. Gesellschaft für Informatik e.V., 107–118.

[2] Roy Arends, Rob Austein, Matt Larson, Dan Massey, and Scott Rose. 2005. DNS Security Introduction and Requirements. *RFC* 4033 (2005), 1–21.

[3] Bud P. Bruegger and Peter Lipp. 2016. LIGHT^{est} - A Lightweight Infrastructure for Global Heterogeneous Trust Management. In *Open Identity Summit (LNI, Vol. P-264)*. GI, 15–26.

[4] CEF eSignature. 2021. *MRA Cook-book.* Technical Report. European Commission: DIGIT.

[5] Bruce Christianson and William S. Harbison. 1996. Why Isn't Trust Transitive?. In *Security Protocols Workshop (LNCS, Vol. 1189)*. Springer, 171–176.

[6] Zakir Durumeric, James Kasten, Michael Bailey, and J. Alex Halderman. 2013. Analysis of the HTTPS certificate ecosystem. In *Internet Measurement Conference*. ACM, 291–304.

[7] ETSI. 2020. *TR 103 684 V1.1.1: Electronic Signatures and Infrastructures (ESI); Global Acceptance of EU Trust Services.* Report. European Telecommunications Standards Institute.

[8] EU. 2014. Regulation (EU) No 910/2014 of the European Parliament and of the Council of 23 July 2014 on electronic identification and trust services for electronic transactions in the internal market. Official Journal of the European Union. Retrieved 16 October 2020 from https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32014R0910

[9] FutureTrust Consortium. 2017. *Design documentation Global Trust Service Status List.* Technical Report.

[10] Morrie Gasser, Andy Goldstein, Charlie Kaufman, and Butler Lampson. 1989. The Digital Distributed System Security Architecture. In *12th National Computer Security Conference*. NIST/NCSC, 305–319. https://www.microsoft.com/en-us/research/publication/the-digital-distributed-system-security-architecture/

[11] Hans Graux and Edwin Jacobs. 2019. LIGHTest D3.7 Cross-Border Legal Compliance and Validity of Trust Scheme Publication. https://www.lightest.eu/static/deliverables/D3.7.pdf. online, accessed on 17 February 2023.

[12] Hans Graux and Edwin Jacobs. 2019. LIGHTest D6.8 Cross-Border Legal Compliance and Validity of Trust Policy and Trust Decisions. https://www.lightest.eu/static/deliverables/D6.8.pdf. online, accessed on 17 February 2023.

[13] Paul E. Hoffman and Jakob Schlyter. 2012. The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA. *RFC* 6698 (2012), 1–37.

[14] Martin Hoffmann. 2020. Publishing Information for Entities Identified by Domain Names. https://datatracker.ietf.org/doc/draft-hoffmann-dnsop-names-as-identifiers. *Expired Internet-Draft* (2020).

[15] Russell Housley, Warwick Ford, W. Timothy Polk, and David Solo. 1999. Internet X.509 Public Key Infrastructure Certificate and CRL Profile. *RFC* 2459 (1999), 1–129.

[16] Detlef Hühnlein, Tilman Frosch, Jörg Schwenk, Carl-Markus Piswanger, Marc Sel, Tina Hühnlein, Tobias Wich, Daniel Nemmert, René Lottes, Juraj Somorovsky, Vladislav Mladenov, Cristina Condovici, Herbert Leitold, Sophie Stalla-Bourdillon, Niko Tsakalakis, Jan Eichholz, Frank-Michael Kamm, Andreas Kühne, Damian Wabisch, Roger Dean, Jon Shamah, Mikheil Kapanadze, Nuno Ponte, Jose Martins, Renato Portela, Cagatay Karabat, Snezana Stojicic, Slobodan Nedeljkovic, Vincent Bouckaert, Alexandre Defays, Bruce Anderson, Michael Jonas, Christina Hermanns, Thomas Schubert, Dirk Wegener, and Alexander Sazonov. 2016. FutureTrust - Future Trust Services for Trustworthy Global Transactions. In *Open Identity Summit (LNI, Vol. P-264)*. GI, 27–41.

[17] Paul V. Mockapetris. 1987. Domain names - concepts and facilities. *RFC* 1034 (1987), 1–55.

[18] Sebastian Mödersheim, Anders Schlichtkrull, Georg Wagner, Stefan More, and Lukas Alber. 2019. TPL: A Trust Policy Language. In *IFIPTM (IFIP Advances in Information and Communication Technology, Vol. 563)*. Springer, 209–223.

[19] Stefan More and Lukas Alber. 2022. YOU SHALL NOT COMPUTE on my Data: Access Policies for Privacy-Preserving Data Marketplaces and an Implementation for a Distributed Market using MPC. In *ARES*. ACM, 137:1–137:8.

[20] Stefan More, Peter Grassberger, Felix Hörandner, Andreas Abraham, and Lukas Daniel Klausner. 2021. Trust Me If You Can: Trusted Transformation Between (JSON) Schemas to Support Global Authentication of Education Credentials. In *SEC (IFIP Advances in Information and Communication Technology, Vol. 625)*. Springer, 19–35.

[21] Stefan More, Sebastian Ramacher, Lukas Alber, and Marco Herzl. 2022. Extending Expressive Access Policies with Privacy Features. In *TrustCom*. IEEE, 574–581.

[22] Grant Noble, Brent Zundel, Dave Longley, Daniel Burnett, and Manu Sporny. 2019. *Verifiable Credentials Data Model 1.0.* W3C Recommendation. W3C.

[23] Radia J. Perlman. 1999. An overview of PKI trust models. *IEEE Netw.* 13, 6 (1999), 38–43.

[24] Heiko Roßnagel. 2017. A Mechanism for Discovery and Verification of Trust Scheme Memberships: The Lightest Reference Architecture. In *Open Identity Summit (LNI, Vol. P-277)*. Gesellschaft für Informatik, Bonn, 81–92.

[25] Klaus Stranacher, Thomas Lenz, and Konrad Lanz. 2013. Trust-Service Status List Based Signature Verification - Opportunities, Implementation and Survey. In *EGOVIS/EDEM (LNCS, Vol. 8061)*. Springer, 29–42.

[26] Georg Wagner, Sven Wagner, Stefan More, and Martin Hoffmann. 2019. DNS-based Trust Scheme Publication and Discovery. In *Open Identity Summit (LNI, Vol. P-293)*. GI, 49–58.

[27] Sven Wagner, Sebastian Kurowski, and Heiko Roßnagel. 2019. Unified Data Model for Tuple-Based Trust Scheme Publication. In *Open Identity Summit (LNI, Vol. P-293)*. GI, 131–142.

A EXAMPLE TRUST POLICY WITH TRANSLATION

Listing 2 (on the next page) shows the corresponding example trust policy in the Prolog-inspired TPL syntax.

```
accept(Transaction) :-
  extract(Transaction, signer_cert, Certificate),
  check(Certificate, format, x509cert),
  extract(Certificate, pubKey, PK),
  verify_signature(Transaction, PK),
  check_qualified(Certificate).


check_qualified(Certificate) :-
  extract(Certificate, issuer, IssuerCert),

  trust(IssuerCert, demoEidas, TrustData),

  verify_service_type(TrustData),

  extract(TrustData, pubKey, PkIss),
  verify_signature(Certificate, PkIss).


trust(IssuerCert, TrustedScheme, TrustData) :-
  print(trust_without_Translation),

  extract(IssuerCert, trustScheme, Claim),
  trustlist(Claim, IssuerCert, TrustData),

  trustscheme(Claim, TrustedScheme).


trust(IssuerCert, TrustedScheme, TrustedData) :-
  print(trust_with_Translation),

  extract(IssuerCert, trustScheme, Claim),
  trustlist(Claim, IssuerCert, TrustData),

  encode(Claim, TrustedScheme, TTAdomain),
  lookup(TTAdomain, TranslationData),

  translate(TranslationData, TrustData, TrustedData).


verify_service_type(TrustData) :-
  extract(TrustData, serviceType, ca_qc),
  print(verified_ServiceType_as_Qualified_Cert).

verify_service_type(TrustData) :-
  extract(TrustData, serviceType, ca_qseal),
  print(verified_ServiceType_as_Qualified_Seal).
```

Listing 2: Example TPL trust policy including an explicit trust translation to target scheme *demoEidas*.