D4.2

Conceptual Framework for Trust Scheme Translation (2)

Document Identification	
Date	16.08.2018
Status	Final
Version	1.0

Related WP	WP2, WP 3, WP5	**Related Deliverable(s)**	D3.1, D2.14, D4.1, D4.3, D4.4
Lead Authors	ATOS	**Dissemination Level**	PU
Lead Participants	ATOS	**Contributors**	FHG, G+D, DTU, USTUTT, UBI
Reviewers	OIX, CORREOS		

This document is issued within the frame and for the purpose of the LIGHT*est* project. LIGHT*est* has received funding from the European Union's Horizon 2020 research and innovation programme under G.A. No 700321.

1 Executive Summary

The aim of this deliverable is to provide the final design of the Conceptual Framework for Trust Scheme Translation within the LIGHT^{est} architecture.

This final stage of the conceptual framework results from the integration of the different trust schemes, and their representation, and the formats for expressing trust translation lists, in order to support reasoning about translation in a compatible way with the publication framework of WP3 and achieve seamless integration.

To have a general view of the concept of a Trust Translation Authority, as the main actor for the translations in LIGHT^{est}, Chapter 6 presents the design of the model of the TTA, consisting of a DNS server with DNSSEC extension and at least one Trust Translation List Provider, which is operated by the Trusted Scheme Operator and provides a list of the Recognized Trust Scheme Levels for which a bilateral agreement provides a translation to a given Trusted Scheme.
It includes the interplay of Trust Scheme Level Translation and Trust Scheme Publication, as part of the integration of the different components in the framework. In section 6.3, the details of the Trust Policy Language (TPL) are provided, explaining how this tool can be used as a way of representing the translations between Trust Scheme Levels not only in the simplest cases but in more complex cases like tuple-based schemes.

In section 6.4, the different components of a Trust Translation Authority are explained from a technical and functional point of view, describing the modules that are needed for the construction of a complete framework and how they are performing their different tasks involved in the translation creation. A Trust Translation List Provider as the main component that provides the data required for representing a translation between a Trusted Scheme and a Recognized Scheme.

After presenting all the components, it is explained in Chapter 7 how these components perform the two main actions of the TTA: the publication and the discovery. The publication as the method to make the translation declaration available using DNS-based mechanism and the discovery process as the way the Trust Translation List is being retrieved from LIGHT^{est}.

Finally, in Chapter 8, an example of a scenario of a trust translation is presented. This scenario has been chosen in the mobile environment, in order to open the range of possibilities where the LIGHT^{est} infrastructure can be used as a use case.

2 Document Information

2.1 Contributors

Name	Partner
Miguel A. Mateo	Atos
Javier Presa	Atos
Miryam Villegas	Atos
Frank-Michael Kamm	G+D
Sebastian Kurowski	FHG
Sebastian Mödersheim	DTU

2.2 History

Version	Date	Author	Changes
0.1	04.06.2018	Javier Presa	Initial draft, Table of Contents
	07.06.2018	Miryam Villegas	New sections and comments added.
	08.06.2018	Javier Presa	Review comments and update content
0.2	13.06.2018	Miryam Villegas	New chapter with a complete LIGHTest scenario.
0.3	14.06.2018	Miguel Angel Mateo Javier Presa	Review comments and update content Homogenize ToC with previous WP4 deliverables
0.4	19.06.2018	Frank-Michael	Section 8.1
0.5	03.08.2018	Javier Presa Miryam Villegas Miguel Angel Mateo	Completion of Section 6.3 and65.4 Chapter 7
0.6	13.08.2018	Miryam Villegas Javier Presa	Reviewer comments included
1.0	16.08.2018	Miryam Villegas	Final version

3 Table of Contents

3.1 Table of Figures

3.2 Table of Acronyms

AAID	Authenticator Attestation Identifier (FIDO protocol)
API	Application Programming Interface
ATV	Automated Trust Verifier
DANE	DNS-based Authentication of Named Entities
DNS	Domain Name System
DNSSEC	Domain Name System SECurity extensions
eIDAS	Electronic IDentification And Signature (Regulation (EU) No 910/2014 on electronic identification (eID) and trust services for e-transactions in the internal market)
FIDO	Fast IDentity Online protocol
FIDO UAF	FIDO Universal Authentication Framework
HTTP(S)	Hypertext Transfer Protocol (Secure)
ID	Identity, identification
JSON	JavaScript Object Notation. It is a lightweight data-interchange format
LoA	Level of Assurance
PIV	Personal Identity Verification
PKI	Public Key Infrastructure
REST	Representational State Transfer (service)
SMIMEA	DNS resource record (RR) is used to associate an end entity certificate or public key with the associated email address, thus forming a "SMIMEA certificate association".
SSL	Secure Sockets Layer
TPL	Trust Policy Language
TSL	Transport Layer Security
TSPA	Trust Scheme Publication Authority
TTA	Trust scheme Translation Authority
TTL	Trust Translation List
URI	Uniform Resource Identifier
WP	Work Package

4 Scope of the Deliverable

This deliverable describes the final design of the Conceptual Framework for Trust Scheme Translation within the LIGHT^{est} architecture. As this D.4.2 is the second deliverable which introduces the design of the framework in its final stage, it is based on the previous deliverables of WP4. Therefore, this deliverable includes summarized parts, as an introduction, from different sections previously published in D.4.1 ([1]), D.4.3 ([2]) and D.4.4 ([3]) for better understanding and aiming to have a single self-explained document. In addition, due to the confidential nature of deliverable 4.1, and for better understanding of this final document, some sections are directly extracted from the initial iteration of the conceptual design and included in this document.

The Conceptual Framework includes an overall view of the Trust Translation Authority, its main components needed to provide the Trust Translation Lists and the explanation of the different activities to be performed during the mentioned provision: Publication (D.4.3 [2]) and Discovery (D.4.4 [3]).

The trust translation is studied for different types of trust scheme levels, including boolean, ordinal and trust levels that use arbitrary sets of attributes namely tuple-based trust schemes, including the information items necessary to represent these different kinds of trust translation lists, and define suitable translation algorithms.

The creation of concrete mappings from one trust scheme to another is instead outside the scope of this Trust Translation Authority framework, as they have to be provided through bilateral agreements between authoritative organizations.

In order to reach the goal of a final design, the study of several key trust schemes carried out during the initial stage of the conceptual design in D4.1 ([1]) has allowed the design of the presented data model to be used in the TTA components.

In addition, following the life cycle of the project, after the study, analysis and definition of the publication and discovery activities in its respectively documents, it has been possible to define the module specification of the TTA components, including all the interfaces with the rest of the components of LIGHT^{est,}

5 Terminology

This section collects the definitions of the most important terminology used throughout this deliverable. The intention of this section is to provide the same general view of the concepts to all the readers.

- **Automated Trust Verifier**

ATV is a component used by anyone to verify electronic transactions. This component queries the Trust Translation Authority to get Trust Scheme Levels equivalent to the object of verification.

- **Recognized scheme**

Given an input trust scheme in which we trust already, the recognized scheme is the result of the LIGHT^{est} translation process having that input trust scheme as the trusted scheme.

Synonyms are: target scheme, or output scheme.

- **Trust levels**

(Or Trust scheme level) They are the different categories given to a trust scheme depending on their levels of assurance in order to classify the grade of trust provided by such a trust scheme.

- **Trust Translation Authority**

This entity provides the translation between trust levels, by enabling trust scheme providers to indicate which trust levels are recognized as trustworthy by their trust scheme.

6 Trust Translation Authority Model

6.1 Conceptual View of the Trust Translation Authority (TTA)

As presented in the Reference Architecture ([4]) Trust Translation Authority is one of the three main pillars of the LIGHT^{est} architecture (see Figure 1 Trust Translation Authority). It operates a standard DNS Name Server with DNSSEC extension. A server is able to publish multiple Trust Translation Lists under different sub-domains of the Authority's domain name. Trust Lists express which authorities from other Trust Domains are trusted. In order to reduce the response package sizes of the DNS Name Server, the TTA consists of 2 components: The DNS Server and the Trust Translation List Provider.

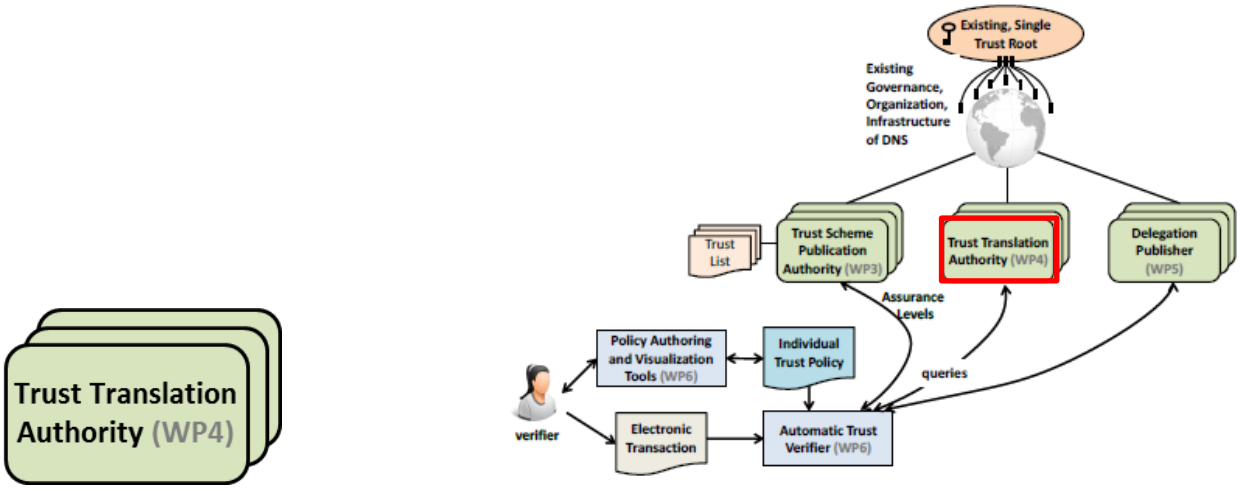

Figure 1 Trust Translation Authority in LIGHT^{est} architecture

In the Figure 2 it is shown how the TTA always consists of a DNS Server and at least one or more Trust Translation List Providers. The DNS Server is used for discovering the Trust Translation List of the Trust Scheme Levels that an issuer complies with. Any translation is hereby carried out between a scheme which is trusted by the ATV (Trusted Scheme) and the scheme that is recognized by the operator of the Trusted Scheme (Recognized Scheme). The Trust Translation List Provider is always operated by the Trusted Scheme Operator and provides a list of the Recognized Trust Schemes.

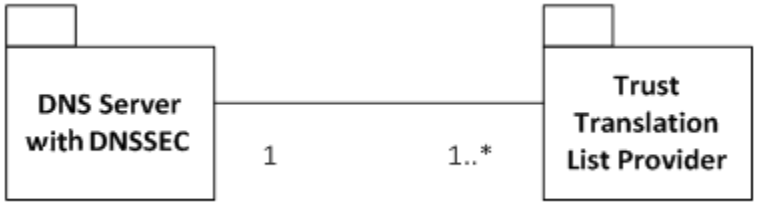

Figure 2 Main components of the Trust Translation Authority

Trust Schemes in the LIGHT^{est} framework are represented in a unified way, by using the representation scheme provided in the universal framework for trust scheme publications by the Trust Scheme Publication Authority, but a dependency of the TTA from this TSPA exists. Both authorities could be communicated by the Automated Trust Verifier independently, as the trust scheme is being made available by the TSPA and the translation by the TTA.

In Figure 3 Conceptual view on the data representation of TTA translation an overview is provided on the conceptual view of data representation in the TTA. The conceptual view distinguishes between data required for discovery of the Trust Translation List (left), and data required for representing a translation between a Trusted Scheme and a list of Recognized Schemes (right).

Such a Trust Translation List contains a relation between the Trusted Scheme and one or many Recognized Schemes, as a Trusted Scheme could be equivalent to not only one Recognized Scheme.

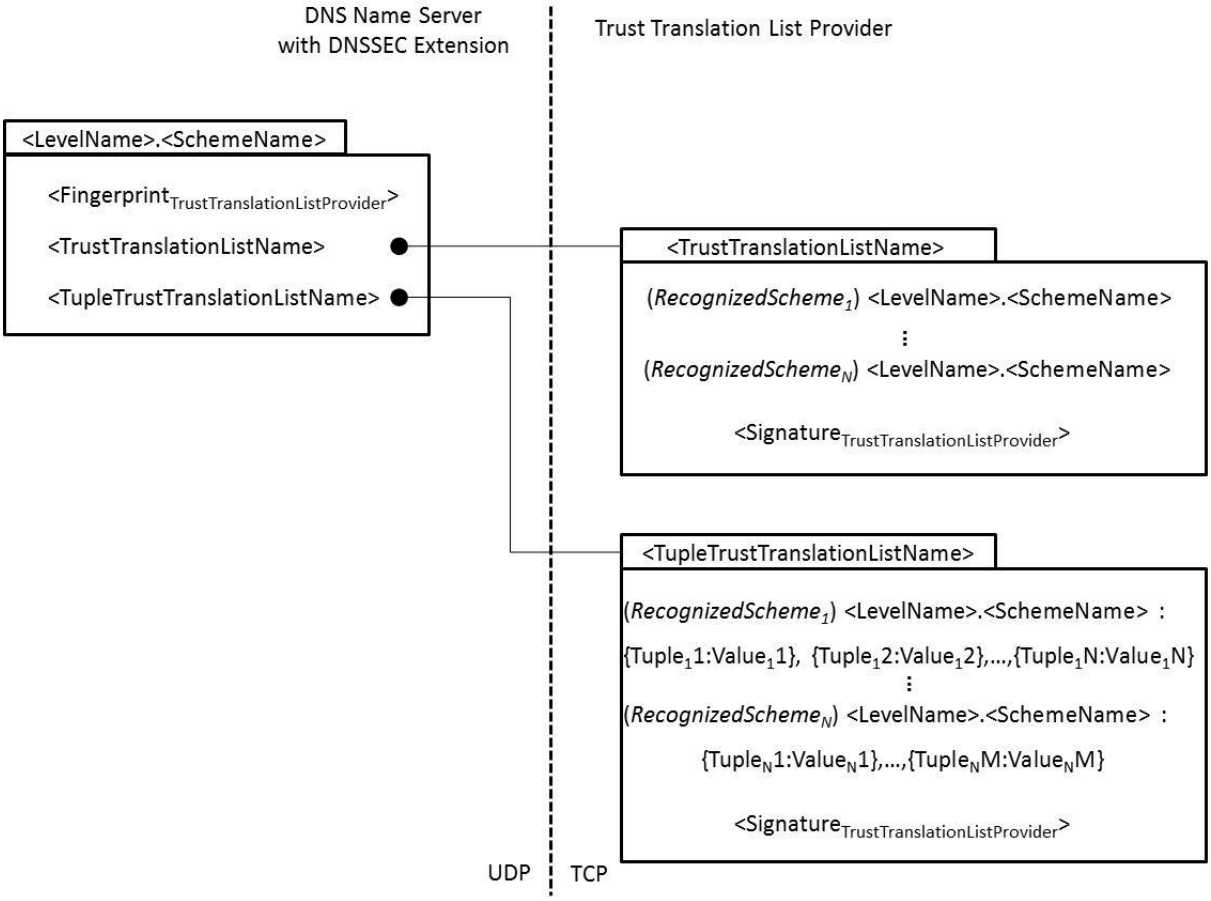

Figure 3 Conceptual view on the data representation of TTA translation

The TTA is designed to provide translations between trusted and recognized schemes, these schemes schemes being Boolean, Ordinal or Tuple-based. Any data package on the DNS is identified by the associated Boolean (<SchemeName>) Trust Scheme Name or Ordinal (<Levelname>.<SchemeName>) Trust Scheme Name. Both Boolean and Ordinal Trust Schemes are managed as they provide translations for tuple-based schemes but in a simpler way of representing it.

In the case of the translation between Tuple-based Schemes this is achieved by providing a Tuple-Based Trust Translation List, which is available under the Tuple-Based Trust Translation List Name <TupleTrustTranslationListName> at the Trust Translation List Provider.

The explanation above means any trust levels of a boolean or ordinal trust scheme can be expressed like tuple-based trust levels, leading the TTA to manage the trust schemes in the same way no matter if they are boolean, ordinal or tuple-based. It is important to understand that every published trust level could be kept in a tuple-based way.

6.2 Trust Scheme Level Translation and Trust Scheme Publication

The Trust Translation Authority (TTA) provides translations between trust scheme levels, as shown in the previous section. Trust Schemes in the LIGHT^{est} infrastructure are represented in a unified way, by using the representation scheme provided in the universal framework for trust scheme publications (see Deliverable D3.2 [5]). Therefore, a strong interplay between the TTA and the Trust Scheme Publication Authority (TSPA) exists (see Figure 4).

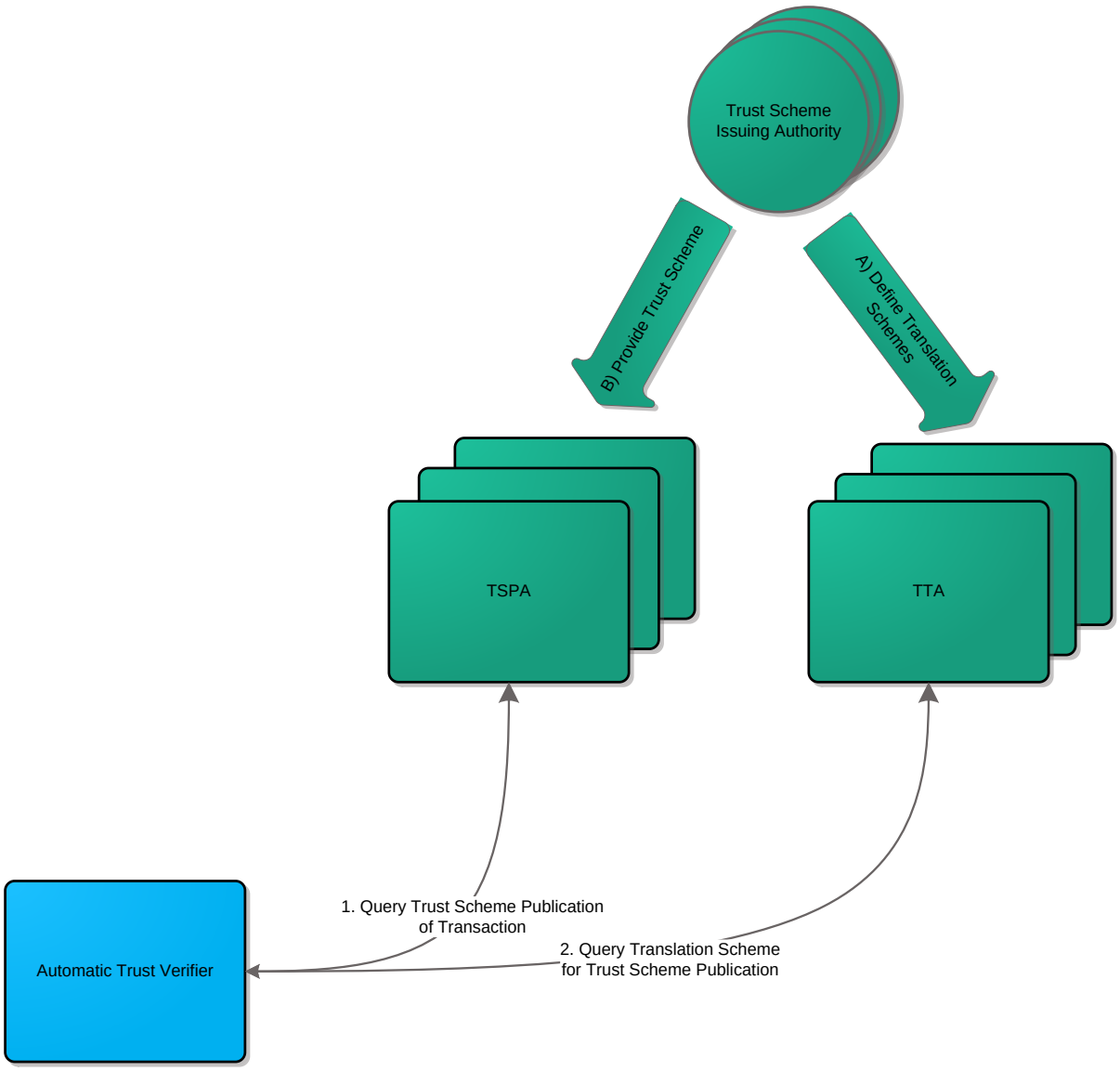

Figure 4 Relationship between the TSPA, TTA, ATV, and the trust scheme issuing authorities

A Trust Scheme is being made available, and processed by representing the trust scheme with the universal framework for trust scheme publications. A trust scheme issuing authority, such as ISO, governments, or companies, therefore provides representations of the trust scheme, which it wants to be queried with LIGHT^{est}. This representation of the trust scheme is being published by the TSPA (See Step B in Figure 4).

As trust itself is a diverse and temporal topic, which can not only result from different trust aspects, such as technical components, managerial procedures, or legal compliance, but may also be transient in nature, since state-of-the-art, e.g. in technical components changes over time, an objectified translation is hard, if not impossible to achieve. Even if an objectified translation between two trust schemes could be achieved, its validity is restricted by time, due to the transient nature of trust.

However, the TTA is able to provide parties defining trust schemes with the capabilities of defining additional trusted schemes, or additionally trusted tuple values. A Trust Scheme Issuing Authority can hereby negotiate with other Trust Scheme Issuing Authorities on whether their schemes trust each other, and in what way. The outcomes of these negotiations are then provided to the TTA (Step A), which represents the translation scheme for automated processing in LIGHT^{est}.

6.3 Trust Policy Language in Translation

While translation of trust schemes is trivial to express for the simplest cases – for Boolean/Ordinal schemes it is a matter of stating which trust schemes are equivalent – the more involved cases are more demanding.

A suitable description language can be used to specify such arbitrary translations, and the Trust Policy Language (TPL), which is used to define trust policies in LIGHT^{est} by the ATV component, is sufficiently expressive. This section introduces TPL as a tool to define translation between trust schemes. A general introduction of TPL can be found in chapter 12.6 of D2.14 [4].

A TPL program consists of a set of clauses, each of which has the form:

```
conclusion :- requirements.
```

The conclusion is a *predicate*, and the requirements are a (possibly empty) set of comma-separated predicates. Typically, a clause is read as "if the requirements are true, then the conclusion is true". As an example, consider the *add*-predicate that calculates the sum of two numbers, *X* and *Y*, and unifies it with a third number, *Z*.

```
add(X,Y,Z) :- Z is X + Y.
```

As mentioned in chapter 12.6 of D2.14 [4], TPL is valid Prolog code, so this *add*-clause can be loaded by Prolog, and it is then possible to have Prolog evaluate, for example, *add(2,3,S)*, which would yield *S = 5*, or, *add(3,5,8)*, which would yield *true*.

6.3.1 A TPL fragment

To demonstrate the utility of TPL in defining translation we discuss here how in general translation can be formulated using the translation from a hypothetical EU trust scheme to a hypothetical US trust scheme:

```
translate(EU,US) :-
        translate_level(EU,US),
        translate_person(EU,US).
```

Here EU and US are both variables representing certificates in an EU and US trust schemes, and we define the predicate *translate(EU,US)* that says for which values of EU and US the translation

is valid. Typically, the ATV will use this by fixing a value for the first argument, EU, and the reasoning yielding a concrete value (if any exists) for US that represents the translation.

Here we have specified that the complete translation requires the translation of two attributes, an ordinal trust level of the scheme, and an "in-person" attribute. These translations will be specified in further clauses for *translate_level(...)* and *translate_person(...)* below. We can thus read the clause as “if the *level*-attribute translates from a first scheme to a second one, and the *person*-attribute translates from that first scheme to that second scheme, then the first scheme translates to the second scheme”.

In general, the translation of an entire tuple-based certificate/scheme is thus broken down into smaller parts of the translations of single attributes. Note that we also give here the entire certificate/scheme as argument to the translations, so they may rely on more than one attribute in the first scheme to yield a value in the second scheme.

As long as we do not specify the *translate_level*- and *translate_person*-predicates, the translation is abstract. However, it is possible to extend the TPL-specification to also define these predicates. Note that the upper-case, green words are *variables*, while the lower-case ones are *constants*.

```
translate_level(EU,US) :-
    extract(EU,loa,3),
    extract(US,level,b).
translate_level(EU,US) :-
    extract(EU,loa,4),
    extract(US,level,a).
translate_person(EU,US) :-
    extract(EU,inperson,true),
    extract(US,person,true).
translate_person(EU,US) :-
    extract(EU,inperson,false),
    extract(US,person,false).
```

We introduce here another predicate *extract* that allows us to access the attributes of a scheme without referring to the details of the concrete certificate formats. Thus, for every certificate format that we want to work with, we have to define the extract predicate to obtain the concrete value of a given attribute. For instance *extract(EU,inperson,X)* should look up in the certificate EU what value the attribute "inperson" has and bind it to the variable X. If we directly specify a concrete value instead of X, e.g., *extract(EU,inperson,true)*, then we require that the value of the attribute is "true".

Notice that the first two clauses have the same conclusion: `translate_level(EU,US)`. The meaning of this is that there are two ways to satisfy the predicate. One way is for the *loa*-attribute of the *EU*-scheme to be *3* and the *level*-attribute of the US-scheme to be *b*. The other way is for the *loa*-attribute of the *EU*-scheme to be *4* and the *level*-attribute of the US-scheme to be *a*.

It is also possible to express a conditional translation. For example, the following clause says that "*EU loa 3* translates to *US level b* if the *EU*-scheme has the *inperson*-attribute, and the value of that attribute is *true"*.

```
translate_conditional_level(EU,US) :-

    extract(EU,inperson,true),

    extract(EU,loa,3),

    extract(US,level,b).
```

Using the translate-*predicate* is not restricted to checking whether two known schemes translate to each other. Similarly to the *add*-predicate, where the third argument could be left as variable – i.e. *add(2,3,S)* – to calculate the sum, it is possible to leave one of the argument-schemes as a variable, and have the program calculate the translation from the first scheme.

This, however, will result in an incomplete (and probably incorrect) translation if some attribute that should be considered is left out of the specification by mistake. We can alleviate this with the following clause. Notice here that an underscore, "_", is used to mean *anything*.

```
underspecified(SCHEME) :-

    translate(_,SCHEME),

    extract(SCHEME,_,error).
```

This should express that there is a translation to a scheme that has the value *error* in some attribute, meaning that the translation failed to provide a value for that attribute.

The underspecified predicate serves as a way to check that any specified translate clause does not consider every attribute. So, it should *not* be a part of a specification. Rather, it is a tool that can be used to find problems with a specification.

The given example of a translation is for tuple-based schemes. The specification is of course much easier for ordinal schemes. While it is possible to handle ordinal trust schemes as a special case of a tuple-based scheme, we anticipate here that in most cases the ordinal value will be encoded into a scheme name. So instead of the hypothetical EU scheme we may instead have schemes "eu-loa-1", .., "eu-loa-4", and the US schemes "us-a" .. "us-c". We can then define the ordinal translation simply as:

```
translation(EU,US) :-

        extract(EU,schemename,EUNAME),
```

```
        extract(US,schemename,USNAME),

        ordinalTranslation(EUNAME,USNAME).

ordinalTranslation("eu-loa-1","us-c").

ordinalTranslation("eu-loa-2","us-c").

ordinalTranslation("eu-loa-3","us-b").

ordinalTranslation("eu-loa-4","us-a").
```

Here we assume that both schemes have an "attribute" *schemename* (but we could also use another predicate name for this). In fact, we anticipate the specification of most translation will be such simple "translation tables" like the predicate *ordinalTranslation* here, that simply relates the concrete values that correspond to each other. The advantage of TPL is that we can easily integrate this with more complicated translations (e.g. when some case allows a "better" level in the target scheme when some attribute of the input is fulfilled). Moreover, we can often easily abbreviate such tables by saying for instance that an attribute in the target scheme is an aggregation from some input scheme attributes, e.g., a resulting level is the minimum of three input levels.

6.3.2 TPL in a Tuple-Based Trust Level Translation

As Tuple-Based representations of Trust Scheme Publications are expected to be much larger, Translation of Tuple-Based Trust Scheme Publications is provided by Tuple-Based Trust Translation Lists, which are themselves provided by a Trust Translation List Provider. The Trust Translation List Provider is a component of the TTA. For discovery of the Tuple-Based Trust Translation Lists, which are associated with a Trusted Scheme, retrieval of the name of the Tuple-Based Trust Translation List is sufficient. The Tuple-Based Trust Translation List indicates the Translation between tuples of a Trusted Trust Scheme and a Recognized Trust Scheme. A Translation is hereby always specified by a variable for the Trusted Trust Scheme, and a variable for the Recognized Trust Scheme. The identifying names of the Trusted Trust Scheme and the Recognized Trust Scheme are indicated by an additional attribute in the Tuple-Based Trust Translation List.

A Tuple-Based Translation between a Trusted Scheme, and a Recognized Scheme, is always introduced by the following Syntax. Hereby, the Trusted and the Recognized Scheme are each identified by a Variable (in the following $\texttt{Variable}_{\texttt{TrustedScheme}}$ for the Trusted Scheme, and $\texttt{Variable}_{\texttt{RecognizedScheme}}$ for the Recognized Scheme). For identification of the Trusted and the Recognized Scheme, the Scheme Name (`schemename`) of each is extracted from the Recognized and Trusted Scheme. The extracted Scheme Name must hereby either indicate the Name of the Trusted Scheme, or the Name of the Recognized Scheme.

```
translate(Variable_TrustedScheme,Variable_RecognizedScheme) :-

        extract(Variable_TrustedScheme,schemename,name_Trusted),
```

extract(*Variable*$_{RecognizedScheme}$, schemename, name$_{Recognized}$),

translation_condition_1(*Variable*$_{TrustedScheme}$, *Variable*$_{RecognizedScheme}$),

translation_condition_2(*Variable*$_{TrustedScheme}$, *Variable*$_{RecognizedScheme}$),

translation_condition_n(*Variable*$_{TrustedScheme}$, *Variable*$_{RecognizedScheme}$).

Translation between the Tuples of the Trusted Scheme, and the Recognized Scheme are then each defined by a translation condition (translation_condition_1 to translation_condition_n). Each translation condition hereby refers to the translation of one tuple of the Trusted Scheme and the Recognized Scheme. Translation between a Trusted Scheme, and a Recognized Scheme is only possible if all translation conditions apply. A translation condition applies, if both the condition in the trusted and the recognized scheme are fulfilled:

translation_condition_1(*Variable*$_{TrustedScheme}$, *Variable*$_{RecognizedScheme}$) :-

extract(*Variable*$_{TrustedScheme}$, Attribute_A$_{TrustedScheme}$, Value_A$_{TrustedScheme}$),

extract(*Variable*$_{RecognizedScheme}$, Attribute_B$_{RecognizedScheme}$, Value_B$_{RecognizedScheme}$).

The listing above would indicate that translation_condition_1 is fulfilled, if the Trusted Scheme includes the attribute Attribute_A$_{TrustedScheme}$ with the value Value_A$_{TrustedScheme}$, and the Recognized Scheme includes the attribute Attribute_B$_{RecognizedScheme}$ with the value Value_B$_{RecognizedScheme}$.

There can also be multiple translation conditions for Tuples of Tuple-Based Trust Scheme Publications:

translation_condition_2(*Variable*$_{TrustedScheme}$, *Variable*$_{RecognizedScheme}$) :-

extract(*Variable*$_{TrustedScheme}$, Attribute_A$_{TrustedScheme}$, Value_A$_{TrustedScheme}$),

extract(*Variable*$_{RecognizedScheme}$, Attribute_B$_{RecognizedScheme}$, Value_B$_{RecognizedScheme}$).

translation_condition_2(*Variable*$_{TrustedScheme}$, *Variable*$_{RecognizedScheme}$) :-

extract(*Variable*$_{TrustedScheme}$, Attribute_A$_{TrustedScheme}$, Value_C$_{TrustedScheme}$),

extract(*Variable*$_{RecognizedScheme}$, Attribute_B$_{RecognizedScheme}$, Value_B$_{RecognizedScheme}$).

In this case, the translation condition applies if the Trusted Tuple-Based Trust Scheme Publication contains a Tuple with the attribute Attribute_A$_{TrustedScheme}$ and either the value Value_A$_{TrustedScheme}$ or the value Value_C$_{TrustedScheme}$, and the Recognized Tuple-Based Trust Scheme Publication contains a Tuple with the Attribute Attribute_B$_{RecognizedScheme}$ and the value Value_B$_{RecognizedScheme}$.

translation_condition_3(*Variable*$_{TrustedScheme}$, *Variable*$_{RecognizedScheme}$) :-

```
        extract(Variable_TrustedScheme,Attribute_A_TrustedScheme,Value_A_TrustedScheme),

        extract(Variable_RecognizedScheme,Attribute_B_RecognizedScheme,Value_B_RecognizedScheme).

translation_condition_3(Variable_TrustedScheme,Variable_RecognizedScheme) :-

        extract(Variable_TrustedScheme,Attribute_A_TrustedScheme,Value_A_TrustedScheme),

        extract(Variable_RecognizedScheme,Attribute_B_RecognizedScheme,Value_C_RecognizedScheme).
```

In this case, the translation condition applies if the Trusted Tuple-Based Trust Scheme Publication contains a Tuple with the $Attribute\_A_{TrustedScheme}$ and the value $Value\_A_{TrustedScheme}$, and the Recognized Tuple-Based Trust Scheme Publication contains a Tuple with the Attribute $Attribute\_B_{RecognizedScheme}$ and either the value $Value\_B_{RecognizedScheme}$ or the value $Value\_C_{RecognizedScheme}$. This is a rather unusual case, since the translation can be any of two possibilities, i.e., it is not deterministic. Such specifications can make sense in logical approaches that are using non-determinism and constraints. TPL allows for such non-deterministic specifications, but it is also possible to check a specification whether it is deterministic to warn the policy designer in case the non-determinism was accidental.

```
translation_condition_4(Variable_TrustedScheme,Variable_RecognizedScheme) :-

        extract(Variable_TrustedScheme,Attribute_A_TrustedScheme,Value_A_TrustedScheme),

        extract(Variable_TrustedScheme,Attribute_C_TrustedScheme,Value_C_TrustedScheme),

        extract(Variable_RecognizedScheme,Attribute_B_RecognizedScheme,Value_B_RecognizedScheme).
```

In this case, the translation condition applies if the Trusted Tuple-Based Trust Scheme Publication contains a Tuple with the attribute $Attribute\_A_{TrustedScheme}$ and value $Value\_A_{TrustedScheme}$ and a Tuple with the attribute $Attribute\_B_{RecognizedScheme}$ and value $Value\_B_{RecognizedScheme}$. Additionally, the Trusted Tuple-Based Trust Scheme Publication must contain a Tuple with the attribute $Attribute\_C_{TrustedScheme}$ and the value $Value\_C_{TrustedScheme}$.

```
translation_condition_5(Variable_TrustedScheme,Variable_RecognizedScheme) :-

        extract(Variable_TrustedScheme,Attribute_A_TrustedScheme,Value_A_TrustedScheme),

        extract(Variable_RecognizedScheme,Attribute_C_RecognizedScheme,Value_C_RecognizedScheme),

        extract(Variable_RecognizedScheme,Attribute_B_RecognizedScheme,Value_B_RecognizedScheme).
```

In this case, the translation condition applies if the Trusted Tuple-Based Trust Scheme Publication contains a Tuple with the attribute $Attribute\_A_{TrustedScheme}$ and value $Value\_A_{TrustedScheme}$ and a Tuple with the attribute $Attribute\_B_{RecognizedScheme}$ and value $Value\_B_{RecognizedScheme}$. Additionally, the Recognized Tuple-Based Trust Scheme Publication must contain a Tuple with the attribute $Attribute\_C_{RecognizedScheme}$ and the value $Value\_C_{RecognizedScheme}$.

This way, there would be five specified cases in which the translation conditions apply, and these conditions can be combined together in TPL as shown at the start of the section, however one must carefully consider the implications for the translation process, especially in the eventuality that the combined clauses may become non-deterministic.

Using this representation, any condition that applies to a translation between a Trusted Tuple-Based Trust Scheme, and a Recognized Tuple-Based Trust Scheme can be indicated in the Tuple-Based Trust Translation List.

6.4 Description of the TTA components

In this section all the internal components of a Trust Translation Authority are described. In Figure 5: Functional components it is possible to have a general view of the main components. Notice a TTA is working for a given trust scheme publishing its trust level translations, and a TTL provider works for a given trust level of such trust scheme being translated.

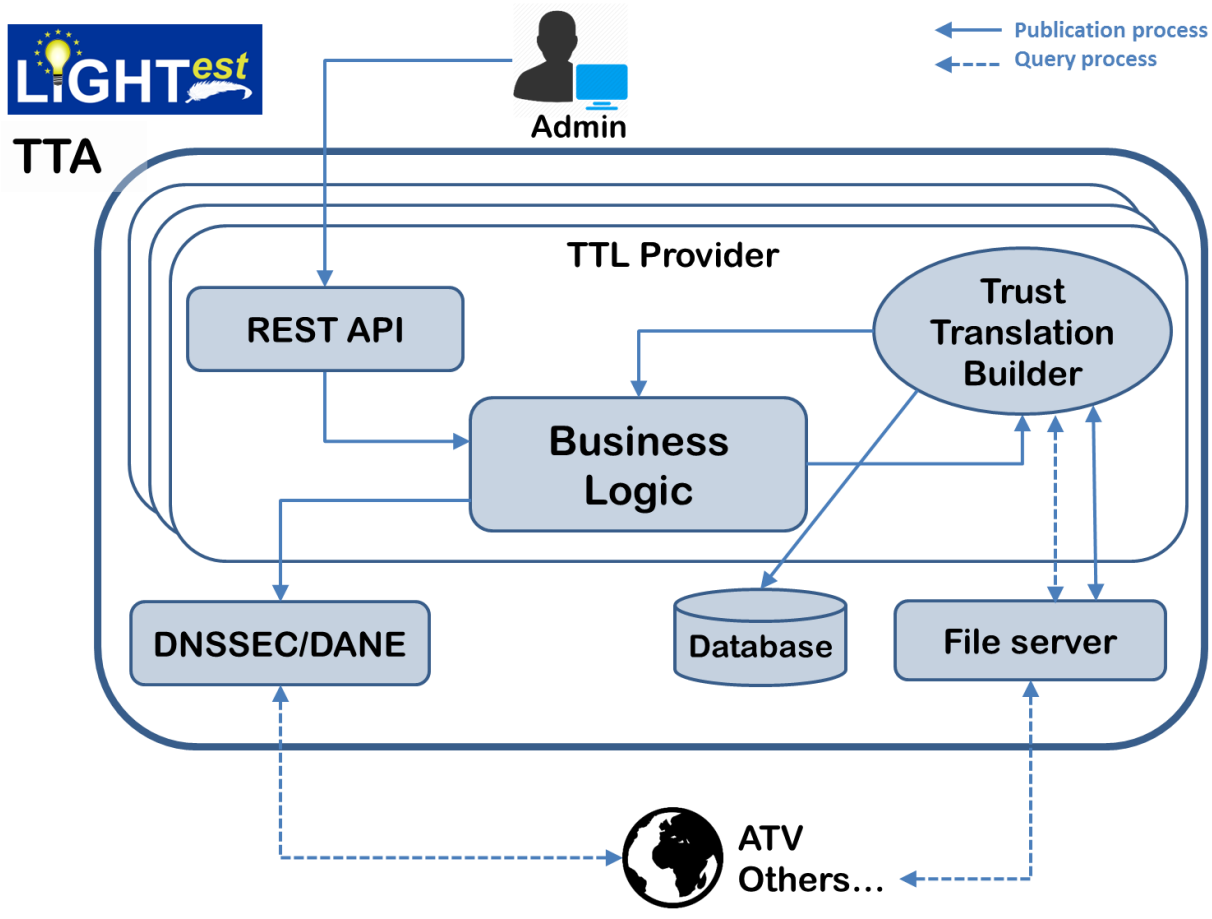

Figure 5: Functional components of the TTA

6.4.1 Trust Translation List Providers

Each TTL provider is responsible for serving the information contained in the files with the trust levels of the recognized (target) trust schemes for the trust level of the trusted (source) scheme.

Multiplicity happens with this component due to the possibility of having more than one trust scheme operator in the same TTA.

Each Trust Translation List Provider consists of the following components:

6.4.1.1 REST API

This component is an application programming interface (API) to be used by the administrator of the TTA to publish for the first time the translations or modify the existing ones. It can be considered the main data input module as it is used to populate the TTA.

In order to create a new translation for a trusted scheme, the REST API needs to obtain the information from such schemes. This is one of the services of this component of the TTA, to know the availability of the trust levels to be translated among the existing trust schemes.

The way of getting mentioned information from the TSPA is the ordinary mechanism to be used by any client. That is, a DNS query to the TSPA to get the related information about the published trust schemes and their trust levels.

Then, once the level of the trusted scheme is selected, the REST API is in charge of starting the process of the registration of the desired translation.

This API is implemented with architectural style of Representational State Transfer (REST), having the following API primitives according to their functionality:

- Retrieve a Trust Scheme. The function to request a Trust Scheme from the TSPA.
- Retrieve Trust Scheme Level information. With this procedure it is possible to obtain the information of the Trust Level of the Trust Scheme retrieved previously from TSPA.
- Add new translation declaration. This is the main input channel for the TTA, allowing the administrator user to insert a new Trust Translation List in the system.
- Retrieve translation declaration. This is used to obtain the information included in a specific declaration of a translation for any administration purpose. It is important to highlight this is not the external interface outside the TTA.
- Remove translation declaration. In case it is needed to remove a declaration of translation from the TTA, this is the procedure to be used.

6.4.1.2 Business Logic

From an architectural perspective this component is considered the core of the TTA as it oversees all the processes performed inside. Business Logic receives the input information from the API REST and invokes the rest of the components in the registration process. Note when querying any TTA for a trust translation, the Business Logic has no role in the query: it is a DNS query, with faster protocol to provide the client with the related answer.

6.4.1.3 Trust Translation Builder

This module is in charge of creating the trust translation as it is, transforming the input received from the data input of the REST API when the translation is being published. The behaviour of this component consists of the generation of a JSON format document with all the data extracted from the agreement that is stored in memory to be sent to the Database component.

In addition, this procedure creates the corresponding text file with a signature to be stored by File Server component. The generation of the different types of text file, XML and TPL formats, are being performed by submodules invoked by Trust Translation Builder.

- XML Generator

It is a subtask that creates a representation of the Trust Translation List in a XML format. The XML File Generator writes an XML file with the information expressing a given trust translation.

Both the trust level name and the trust scheme name have been published in LIGHT^{est} framework by the TSPA.

- TPL Generator

It is another subtask that creates a representation of the Trust Translation List this time in a TPL format. The TPL File Generator writes a TPL file with the information expressing a given trust translation.

Both the trust level name and the trust scheme name have been published in LIGHT^{est} framework by the TSPA.

6.4.2 Internal database

Once a translation is provided an internal database component will store all data extracted from the trust translation agreements. It could be used to regenerate any Trust Translation List in case it is necessary. This component is outside the Trust Translation List Provider and only one common instance is needed at least. This must be accessible from all TTL Providers.

6.4.3 File server

This component manages the storage of the Trust Translation Lists in text file format. It is a server platform that can be implemented using any commercial solution, with the basic requirements of being capable to provide TSL and SSL protocols to secure the communication.

Note it is possible to get more than one file for a given trust translation. For example, we can think about several bilateral agreements between the eSeal in eIDAS ([6]) and other trust schemes having been stored in LIGHT^{est}. As a consequence, a given trust level of the eSeal in eIDAS could have been translated into several trust levels of the other trust schemes. Since we have a final file (xml/tpl) for one trust level to another trust level, we will get as many files as trust translations have been recorded in LIGHT^{est} for that source trust level.

6.4.4 DNS interface

The DNS module provides two functionalities. First, it updates the corresponding structure of the DNS/DANE ([7]) tree related to each new trust translation in the TTA. This function can be only invoked by the TTA internals, it is not public at all.

On the other hand, the DNS module provides a public interface in order to query a TTA for a given trust level translation. The result will be expressed in TPL and XML formats:

```
;; QUESTION SECTION: Client/ATV to the TTA
_translate._trust.trusLevelOfTrustSchemeToBeTranslated.  IN  URI

;; ANSWER SECTION: from the TTA
_translate._trust.trusLevelOfTrustSchemeToBeTranslated.   IN  URI
              https://lightest.eu/ttl_trusLevelOfTrustSchemeToBeTranslated
              .tpl
_translate._trust.trusLevelOfTrustSchemeToBeTranslated.   IN  URI
              https://lightest.eu/ttl_trusLevelOfTrustSchemeToBeTranslated
              .xml
```

6.5 Data model of the TTA

6.5.1 Description of data model for the translations

The key entity in the data model of a TTA is the **Agreement**. Since a translation between trust levels of different trust schemes needs a bilateral legal agreement between the two trust scheme authorities (or trust scheme providers), the Agreement entity is the core of the data model.

An agreement consists of a name, the trust scheme providers (or trust scheme publishing authorities) which sign the agreement, a creation date, an expiration date, a status, and a list of pairs of trust levels (source trust level, target trust level), meaning pairs of trusted trust level, recognized trust level. Note the status of the agreement describes the current state of the agreement signed, as it could be revoked but still needed to verify transactions based on past agreements.

A trust level is the concatenation of the trust scheme name plus the trust service name, following the eIDAS style.

6.5.2 TPL samples

For the simple example (an **ordinal trust level**) we are studying, the TTA will provide two TPL files indicating two possible trust translations for the *advanced* level of the eIDAS eSeal: one translating to the *medium* eSeal of the X Company, and another translating to the *qualified* eSeal of the Y Company. That means these two TPL files:

```
/* ttl_advancedESealEIDAS_1.tpl */

translates_to_eIDAS_advanced(eSeal) :-

  extract(eSeal, schemename, x_co),
```

```
    extract(eSeal, level, medium).


/* ttl_advancedESealEIDAS_2.tpl */

translates_to_eIDAS_advanced(eSeal) :-

  extract(eSeal, schemename, y_co),

  extract(eSeal, level, qualified).
```

The TPL for a translation of a **tuple-based trust level** is expressed following this hypothesis. Relating to the eID card emitted and recognized in a given eIDAS infrastructure (note that this trust scheme is ordinal –high, substantial, low --, not tuple-based), the FIDO Alliance [8] and a certain eIDAS government agreed that such identification documents with High level are valid for the attestation process. However, the government will not allow the use of that FIDO authenticator (so, the agreement is only in one direction). The trust level that FIDO translates is known as "FIDOUAF_1_5" for this example in LIGHT^{est} framework. The values have been extracted from [9]:

- Fingerprint based user verification allowing up to 5 registered fingers, with false acceptance rate of 0.002% and rate limiting attempts for 30 seconds after 5 false trials.
- Authenticator is embedded with the FIDO User device.
- The authentication keys are protected by TEE and are restricted to sign valid FIDO sign assertions only.
- The (fingerprint) matcher is implemented in TEE.
- The Transaction Confirmation Display is implemented in a TEE.
- The Transaction Confirmation Display supports display of "image/png" objects only.
- Display has a width of 320 and a height of 480 pixels. A bit depth of 16 bits per pixel offering True Color (=Color Type 2).
- The zlib compression method (0). It doesn't support filtering (i.e. filter type of=0) and no interlacing support (interlace method=0).
- The Authenticator can act as first factor or as second factor, i.e. isSecondFactorOnly = false.
- It supports the "UAFV1TLV" assertion scheme.
- It uses the ALG_SIGN_SECP256R1_ECDSA_SHA256_RAW authentication algorithm.
- It uses the ALG_KEY_ECC_X962_RAW public key format (0x100=256 decimal).
- It only implements the TAG_ATTESTATION_BASIC_FULL method (0x3E07=15879 decimal).
- It implements UAF protocol version (upv) 1.0 and 1.1.

Expressing by tuples:

```
<trustlevel trustlevelname="FIDOUAF_1_5">

   <tuplelist>
```

```
            <tuple
                  attributename="userVerification"
                  attributevalue="Fingerprint based, up to 5 registered
fingers, false acceptance rate of 0.002%, rate limiting attempts for 30
seconds after 5 false trials">
            </tuple>
            <tuple
                  attributename="AuthenticatorEmbeddedWithFIDOUserDevice"
                  attributevalue="true">
            </tuple>
            <tuple
                  attributename="AuthenticationKeysProtection"
                  attributevalue="TEE">
            </tuple>
            <tuple
                  attributename="FingerprintMatcher"
                  attributevalue="TEE">
            </tuple>
            <tuple
                  attributename="TransactionConfirmationDisplay"
                  attributevalue="image/png">
            </tuple>
            <tuple
                  attributename="Width-HeightDisplay"
                  attributevalue="320-480pixels">
            </tuple>
            <tuple
                  attributename="CompressionMethod"
                  attributevalue="zlib">
```

```
            </tuple>
            <tuple
                  attributename="isSecondFactorOnly"
                  attributevalue="false">
            </tuple>
            <tuple
                  attributename="AssertionScheme"
                  attributevalue="UAFV1TLV">
            </tuple>
            <tuple
                  attributename="AuthenticationAlgorithm"
                  attributevalue="ALG_SIGN_SECP256R1_ECDSA_SHA256_RAW">
            </tuple>
            <tuple
                  attributename="PublicKeyFormat"
                  attributevalue="ALG_KEY_ECC_X962_RAW">
            </tuple>
            <tuple
                  attributename="MethodImplemented"
                  attributevalue="TAG_ATTESTATION_BASIC_FULL">
            </tuple>
            <tuple
                  attributename="UAFversion"
                  attributevalue="1.0-1.1">
            </tuple>
      </tuplelist>
   </trustlevel>
```

The translation in TPL is specified by means of this file: the FIDO Alliance is, hypothetically, accepting the *qualified* eIDAS electronic identity as an attestation valid when a FIDOUAF_1_5 is required.

```
/* ttl_FIDOUAF_1_5_1.tpl */

translate_identity(EIDAS, FIDOUAF_1_5) :-

    extract(EIDAS, schemename, eidas),

    extract(FIDOUAF_1_5, schemename, fidouaf_1_5),

    translate_qual(EIDAS, FIDOUAF_1_5).

translate_qual(EIDAS, FIDOUAF_1_5) :-

    extract(EIDAS, eIdentity_level, qualified),

    extract(FIDOUAF_1_5, userVerification, "Fingerprint"),

    extract(FIDOUAF_1_5, userVerificationUp, "5"),

    extract(FIDOUAF_1_5, userVerificationFalseAcceptPercent, "0.002"),

    extract(FIDOUAF_1_5, userVerificationFalseTrials, "5"),

    extract(FIDOUAF_1_5, authenticatorEmbeddedWithFIDOUserDevice, true),

    extract(FIDUAF_1_5, authenticationKeysProtection, "TEE"),

    extract(FIDOUAF_1_5, fingerprintMatcher, "TEE"),

    extract(FIDO_UAF_1_5, keys_protected_by_TEE_SE, "TEE"),

    extract(FIDOUAF_1_5, transactionConfirmationDisplay, "image/png"),

    extract(FIDOUAF_1_5, width-HeightDisplay, "320-480pixels"),

    extract(FIDOUAF_1_5, compressionMethod, "zlib"),

    extract(FIDOUAF_1_5, isSecondFactorOnly, false),

    extract(FIDOUAF_1_5, assertionScheme, "UAFV1TLV"),

    extract(FIDOUAF_1_5, authenticationAlgorithm,
"ALG_SIGN_SECP256R1_ECDSA_SHA256_RAW"),

    extract(FIDOUAF_1_5, publicKeyFormat, "ALG_KEY_ECC_X962_RAW"),

    extract(FIDOUAF_1_5, methodImplemented, "TAG_ATTESTATION_BASIC_FULL"

    extract(FIDOUAF_1_5, uafversion, "1.0" || "1.1").
```

6.5.3 XML samples

For the same example of the translation of an ordinal trust level as before, the TTA will also provide two XML files indicating two possible trust translations for the *advanced* level of the eIDAS eSeal: one translating to the *medium* eSeal of the X Company, and another translating to the *qualified* eSeal of the Y Company. That means these two XML files:

```
<!--- ttl_advancedESealEIDAS_1.xml -->
<trustlevel_translation>
      <agreement> "the agreement" </agreement>
      <tspa_source> "eIDAS" </tspa_source>
      <tspa_target> "XCo" </tspa_target>
      <trustlevel_source> "advanced_eSeal_eIDAS" </trustlevel_source>
      <trustlevel_target> "medium_eSeal_XCo" </trustlevel_target>
      <creation-date> 2018-06-01 </creation-date>
      <expiration-date> 2019-06-01 </expiration-date>
      <activation-date> 2018-06-02 </activation-date>
      <status> "Available" </status>
</trustlevel_translation>

<!--- ttl_advancedESealEIDAS_2.xml -->
<trustlevel_translation>
      <agreement> "another agreement" </agreement>
      <tspa_source> "eIDAS" </tspa_source>
      <tspa_target> "YCo" </tspa_target>
      <trustlevel_source> "advanced_eSeal_eIDAS" </trustlevel_source>
      <trustlevel_target> "qualified_eSeal_YCo" </trustlevel_target>
      <creation-date> 2018-07-30 </creation-date>
      <expiration-date> 2019-07-30 </expiration-date>
      <activation-date> 2018-07-31 </activation-date>
      <status> "Available" </status>
```

```
</trustlevel_translation>
```

The XML for a translation of a **tuple-based trust level** is expressed below. The same supposition about the agreement between FIDO Alliance and an eIDAS government is used: The imaginary FIDOUAF_1_5 trust level recognizes as a valid attestation the *qualified* level of an eIDAS electronic identity.

```
/* ttl_FIDOUAF_1_5_1.xml */

<trustlevel_translation>

      <agreement> "the agreement" </agreement>

      <tspa_source> "FIDOUAF" </tspa_source>

      <tspa_target> "eIDAS" </tspa_target>

      <trustlevel_source> "FIDOUAF_1_5" </trustlevel_source>

      <trustlevel_target> "qualified_eIdentity_eIDAS"
      </trustlevel_target>

      <creation-date> 2018-08-01 </creation-date>

      <expiration-date> 2019-08-01 </expiration-date>

      <activation-date> 2018-08-02 </activation-date>

      <status> "Available" </status>

</trustlevel_translation>
```

7 Trust Translation Sequences of the TTA

It is the TTA which has the representation of a bilateral agreement. Any Trust Scheme Provider can hereby negotiate with other Trust Scheme Providers on whether their schemes trust each other, and in what way. The outcomes of these negotiations are then provided to the TTA, which represents the translation scheme for automated processing in LIGHT^{est}. This means the TTA becomes a function that allows the interoperability of trust schemes published by different entities, even across different trust domains, by defining the relation between the trust scheme levels.

To perform this function the TTA is able to provide two main different services. The first service is targeted to register the trust translation data to make it publicly available (**publication**), while the second service is the **discovery**, a mechanism that allow clients (mainly the ATV) to query that published information. This query process is performed through the DNS interface.

7.1 Publishing Trust-related information in TTA

The next sections are the steps the TTA will follow to publish a Trust Translation List and makes it publicly available. Notice the administrator has previously chosen the Trust Translation List Provider, where the translation information is going to be kept.

7.1.1 Administrator uses the REST API

1. *Select the first trust scheme in the agreement, hereinafter* ***trust scheme A****,* to be translated.
    a. Select Trust Scheme. REST API queries TSPA (through DNS protocol) to extract the selected Trust scheme published and retrieves the information from it, mainly the trust levels that featured that trust scheme.
    b. Select Trust Level. Among the different trust levels of the trust scheme formerly obtained, the Administrator has to select the one that is in the bilateral agreement to be translated.
2. *Select the second trust scheme in the agreement, hereinafter* ***trust scheme B*** as the target translation
    a. Select Trust Scheme. This queries TSPA using the DNS protocol to extract the required Trust Scheme published and retrieves the information from it (its trust levels, in particular).
    b. Select Trust Level. Among the different trust levels of the trust scheme formerly obtained, the Administrator has to select the one that is in the bilateral agreement to be translated
3. Once both Trust Levels are selected, the TTL Provider where the trust translation will be published is chosen.
4. Then, the administrator invokes the publishing service for the generation of the Trust Translation List for the pair of trust levels selected. All the information will be transferred to the next module, Business Logic, in the same TTL provider.

7.1.2 Business Logic invokes the generation of the translation

In this step, after the Business Logic receives the input data it will send the information to the related Trust Translation Builder in the corresponding TTL provider.

7.1.3 Trust Translation Builder generates the trust translation

The TT builder manages the creation of the Trust Translation List invoking the different sub-procedures to do those tasks.
When the information of the translation is being reived by the TT Builder, it transforms all this information into a JSON format, more convenient to be stored in memory.
This formatted information is, as well, immediately archived in the database. The intention of this action is to have a method for recovering any Trust Translation List in case of any further necessity.
It is after those storing process when the TT Builder is calling to the corresponding sub-procedures to generate the file. This file generator will create the files (.tpl and .xml) with the information of the translation.
It is very important to highlight that for each trust level just translated, there will be a TPL file and an XML file individually. This means that in the case there is more than one Recognized Trust Level for a single Trusted Level, the TTA will produce different .tpl and .xml files for each translation.

7.1.4 File server uploads the translation files

Trust translation builder invokes the File server services to store the files created previously by each file type generator. When the File server receives the files, it uploads each file in the root folder of the server. There is no need to create a hierarchy on the file structure of the server, as every file could be named with its own intuitive name to be searched by. When all the operations are finished and confirmed, the control is returned to the Business Logic.

7.1.5 DNS interface

Once the files have been successfully generated and stored, Business Logic will invoke DNS module to publish the URI of the Trust Translation List into the DNS. This module provides two operations: one to perform the signature needed for DANE and, the other is the publication of the translation itself. DNS zone files are updated with the names of those translations files.

7.2 Discovery of Trust Translation Lists

The discovery process is the mechanism to be used in order to retrieve information of a Trust Translation List from a TTA. The way an entity can obtain the information of a translation declaration in the LIGHT^{est} framework. The following Figure 6 shows a sequence diagram, as presented previously in D2.14 [4], which involves the TTA.

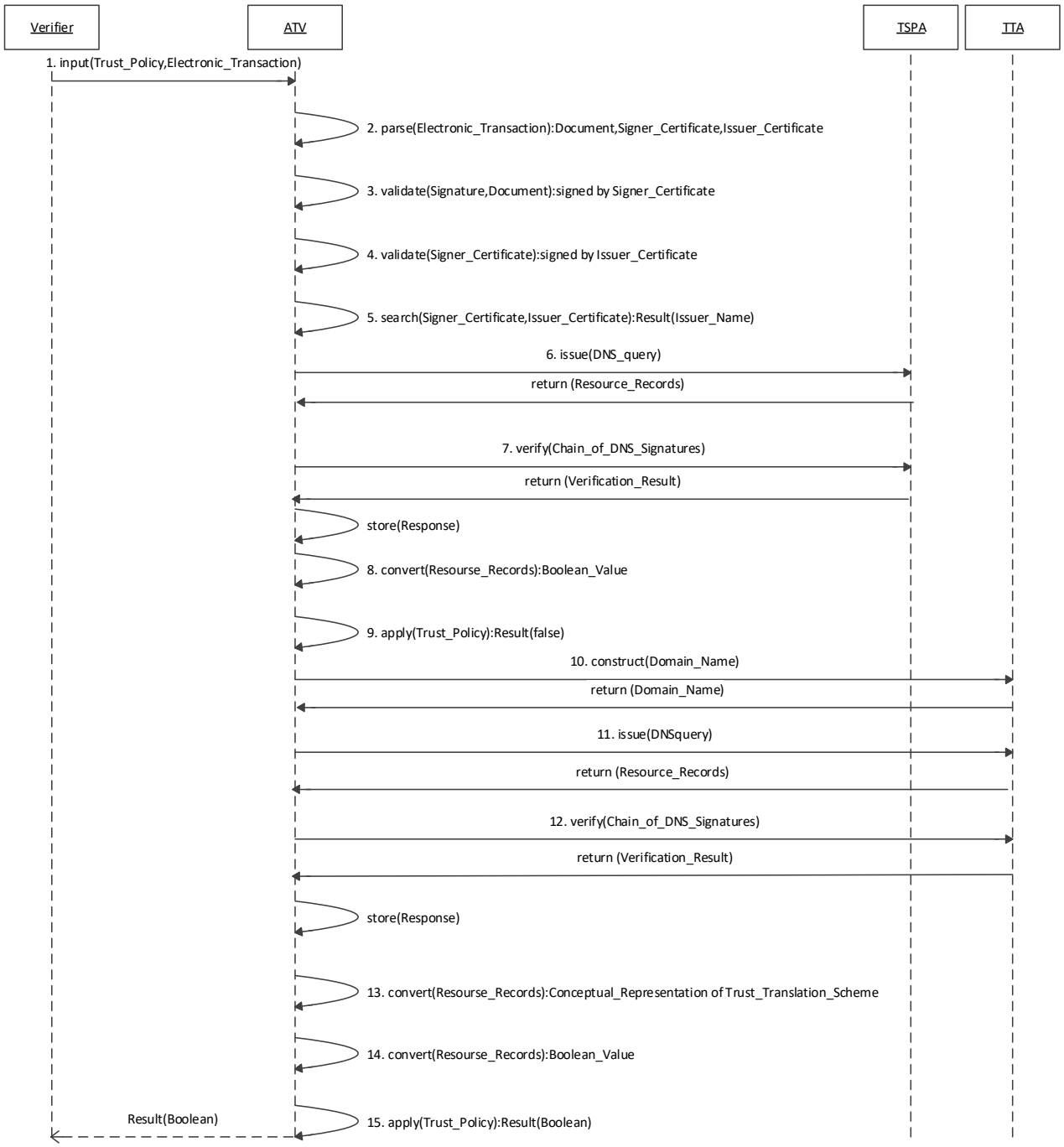

Figure 6 Trust Translation of a qualified signature (boolean) (extracted from D2.14)

Noticeably, the ATV first retrieves the corresponding trust scheme identifier(s) from the TSPA (Step 6). Using the trust scheme identifier, the TTA constructs the corresponding translation scheme identifier for the provided trust scheme identifier(s) (Step 10). In the next Step 11, a DNS query is sent to the TTA, upon which all target schemes are being provided to the ATV. Finally, the Chain of DNS signatures is being verified (Step 12).

Step 10 of the sequence involves the construction of a trust scheme identifier, by transforming any domain name labels into one String, whereas each "." is being converted to "__" (see D2.14). Step 12 is considered part of the existing DNSSEC [7] implementation, and therefore omitted in the following considerations.

Note the *Step 11*, which is more detailed in the next figure. Tuple-Based Trust Translations can be obtained via Tuple-Based Trust Translation Lists, which can be obtained from the Trust Translation List Provider after Discovery of the Trust Translation List Provider for a Trusted Trust Scheme (Steps 1 & 2) (see Figure 7). The ATV queries the Tuple Based Trust Translation List by using the Name (TBTTL_Name) that has been previously obtained from the DNS Name Server (Steps 5 & 6). By using the certificate fingerprint (Fingerprint_TTL_Provider), the ATV is now able to verify the authenticity of the list by validating the signature, and verifying that the used certificate corresponds to the fingerprint stored in the DNS Name Server.

Therefore, the exchanges that take place between the ATV and the TTA are:

a. ATV→TTA/DNS: The ATV contacts the TTA and provides the trust level of the trusted (source) Scheme Name to be translated.
b. TTA/DNS→ATV: The DNS component of TTA provides the information from File Server regarding the corresponding Trust Translation List Provider.
c. ATV→TTA/TTL Provider (File server): The ATV contacts the File Server and requests the Trust Level data of the Trusted (source) Scheme Name.
d. TTA/File server→ATV: The File Server provides a set of files indicating which trust levels are the recognized (target) ones for the provided trust level of the trusted (source) Scheme Name. Each file signed with the Trust Translation List Provider Certificate.
e. ATV→TTA/DNS: The ATV contacts the TTA and provides the trust level of the trusted (source) Scheme Name.
f. TTA/DNS→ATV: The TTA provides the SMIMEA record for the trust level of the trusted (source) Scheme Name.
g. ATV: The ATV verifies whether the certificate used for signing the trust translation list was valid.

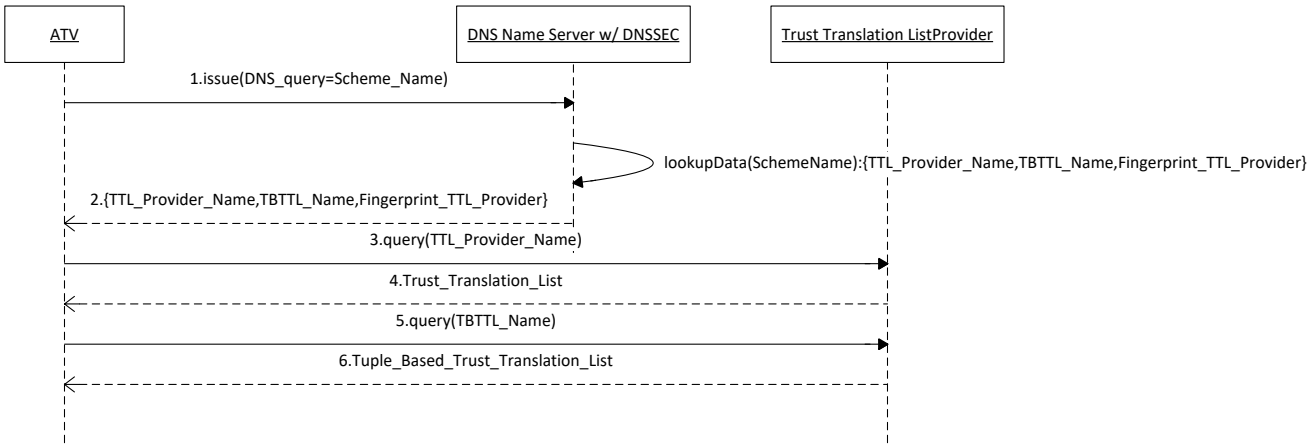

Figure 7 Sequence diagram of steps to obtain boolean, ordinal, and tuple-based trust translation lists

In summary, the main function of any TTA is to provide a list of trust levels of other trust schemes (recognized or target ones) for a given trust level of a (trusted or source) trust scheme. The procedure is known as *discovery* and the related flow is described following:

A LIGHT^{est} client wants to know the trust levels which it could trust on for a certain trust level of its trust scheme:

1. The LIGHT^{est} client builds a DNS/DANE query and sends it to the corresponding TT Authority.
2. The TTA replies with several DNS/DANE messages where some URI pointers to files with translation information are returned.
3. The LIGHT^{est} client makes the HTTP requests to such URIs to get the translation information contained in the downloading files. Each client selects the kind of translation file to be obtained from the TTA, since TPL and XML formats are available. For example, the ATV selects the TPL files, since it is ready for understanding TPL. Any other client could select the XML files instead.
4. After reading the files, the LIGHT^{est} client could store such information *locally* for next similar queries.

The discovery process always requires querying the DNS Name Server with DNSSEC extension and the Trust Translation List Provider. The latter provides a signed statement of association of a trust level of a trusted scheme with the TTL returned.

8 LIGHT^{est} scenario for trust translation

8.1 Mobile ID trust translation

One of the major use cases where trust translation can be applied in a practical way is the mobile ID (or derived ID) use case that is also addressed within LIGHT^{est}. As outlined in deliverable D7.2 [10], LIGHT^{est} develops a mobile ID derivation scheme that allows the establishment of mobile ID credentials that are coupled to a real-life identity. Additionally, the scheme allows for trust propagation of the corresponding Level of Assurance (LoA) of the derived identity. The LoA mainly depends on the LoA of the primary ID and the strength of the authentication mechanism.

The basis of the mobile ID scheme used in LIGHT^{est} is the FIDO protocol, as outlined in deliverable D3.1 [11]. FIDO is based on core elements of PKI (public/private keys) and avoids using full-blown PKI, including certificate chains, wherever possible. However, it can be extended and integrated into ID schemes by using certificates over the public key that binds the FIDO credentials to a verified identity, as described in deliverable D7.2 [10]. If this concept is applied to a derived mobile ID scheme, the complexity can quickly increase if the involved parties are part of different trust schemes. As has been outlined in deliverable D7.1 (Figure 1 in that document [12]), the relevant roles are:

- the primary identity issuer, who has issued the primary ID,
- the secondary identity issuer, who verifies the primary identity and creates the binding to the FIDO credentials,
- the manufacturer of the authenticator, who provides attestation about the type of authenticator and its security properties, and;
- the relying party, that needs to consume the derived mobile ID and has to ensure compliance to its relevant trust scheme.

Based on these foundations, the following scenario is realistic, in which trust translation applies. Without losing generality and in order to outline the main principles, it is kept relatively simple with one trust translation step. If several entities of those listed above are located in different trust schemes, additional trust translation steps may be required.

In this scenario, a U.S. citizen wants to subscribe to an e-government service that is subject to the eIDAS [6] trust scheme. Since the citizen is an employee of a U.S. public administration, he or she can provide a PIV derived identity (see [13] concerning the PIV trust scheme). The e-government web service uses FIDO as an authentication scheme and the U.S. citizen owns a smartphone of a Korean manufacturer with a FIDO authenticator that can facilitate the Trusted Execution Environment (TEE) of this smartphone for credential storage. An overview of this scenario is shown in Figure 9.

According to the LIGHT^{est} ID derivation scheme (D7.2 [10]), the following steps will now occur:

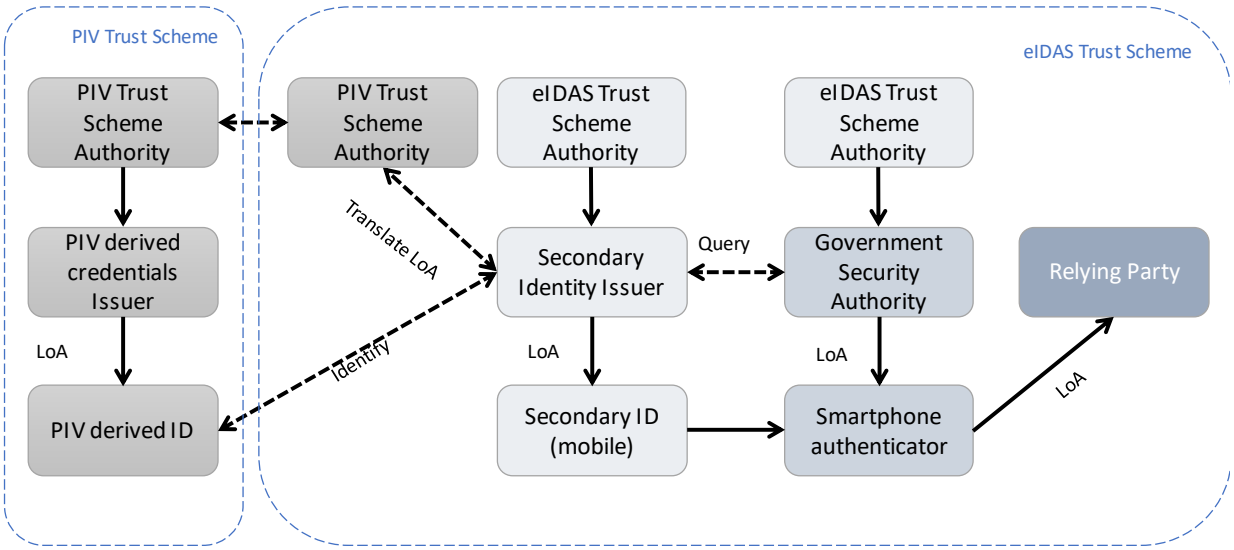

Figure 8: Trust translation scenario between the PIV derived credentials trust scheme and the eIDAS trust scheme for a mobile ID use case.

1. The U.S. citizen registers at the e-government service. In this initial registration step the FIDO key pair is created, the public key uploaded to the FIDO server of the e-government service and the private key is securely stored in the TEE of the U.S. citizen's smartphone. The e-government service receives a signature over the registration request, performed with the attestation key of the smartphone authenticator.
2. Since the e-government service requires a verified identity, it will now re-direct the citizen to an ID provider (secondary ID issuer) with whom the service cooperates. In this step, the public key of the user and the attestation signature of the authenticator are handed over to the ID provider.
3. The ID provider now asks the U.S. citizen to provide an ID. Since PIV is supported by the ID provider, the citizen can present the derived PIV credentials on the mobile phone for identification.
4. The ID provider now has to determine the eIDAS LoA that the citizen can reach with the provided ID and the FIDO authenticator available on the smartphone. To determine the LoA of the (PIV) ID, the ID provider now has to ask for a trust translation of the LoA level from the PIV scheme to the eIDAS scheme. As a result, the corresponding eIDAS level of the ID is returned.
5. As a second component of the eIDAS LoA, the LoA of the presented authenticator has to be determined. If the authenticator LoA were published in another trust scheme (outside eIDAS), another trust translation step would be required at this point. For simplicity we assume that the authenticator LoA can be determined from within the eIDAS scheme. This could be done through an eIDAS certified trust service that publishes a list (or a set of required properties) of acceptable FIDO authenticators and their corresponding LoA. Such a trust service provider could be a government security authority, as an example.
6. With the results of step 4 and 5 the ID provider can now determine the overall LoA that can be reached by the U.S. citizen with the presented ID and the available FIDO TEE-

based authenticator. This LoA, together with the requested identity attributes and the public FIDO key of the citizen is compiled into a certificate that is issued by the ID provider.

7. The certificate is returned to the e-government service. The service can now store the certificate together with the FIDO public key and verify the user identity in future authentication steps without further involvement of the ID provider.

With the domain-oriented publication of trust information by LIGHT^{est} it is also possible that this concept is extended to domain-specific trust schemes. In this concrete use case scenario, it is possible for example that a domain-specific scheme, e.g. maintained by a national banking association and used for payment applications, publishes the trust information about achievable security levels with certain types of authenticators. The TEE-based authenticator of the Korean mobile phone manufacturer in this example could be listed there. This can be done either by publishing the Authenticator Attestation ID (AAID) for this specific authenticator or by publishing general authenticator properties that are reflected in the authenticator metadata (for more details see deliverable D3.3 [14]).

The domain-specific payment trust scheme would now claim conformance with eIDAS LoAs [6] for the listed authenticators. With trust translation this could now be translated into the eIDAS scheme to determine the achievable LoA of this specific authenticator.

9 Conclusion

This document shows the concepts needed to understand the translation between levels of trust schemes and their representation in LIGHT^{est}. After making an analysis, in a first stage of the project, of several trust schemes, we concluded they are three different ways of representing a trust scheme: Boolean, ordinal and tuple-based. This has been used in the final stage to evaluate the components needed to create and retrieve data from a translation and integrate everything to have a unique conceptual framework.

This framework, in form of a Trust Translation Authority, provides a very useful way to make all the translations between Trust Scheme Levels publicly available using DNS infrastructure, gaining all the advantages of this system. It includes mechanism to verify the authenticity of the information by means of DNSSEC and DANE extensions. This translation is a digital representation of a bilateral agreement signed between official authorities.

On the other hand, the Trust Translation Authority is structured in different modules, having a variety of possibilities when the LIGHT^{est} platform is deployed in real cases, giving adaptability and scalability to any type of use case.

10 References

[1] The LIGHTest Project, “D4.1- Conceptual Framework for Trust Scheme Translation (1),” Project Deliverable, 2017.

[2] The LIGHTest Project, D4.3 - DNS-based Publication of Trust Translation Schemes, Project Deliverable, 2017.

[3] The LIGHTest Project, D4.4 - Discovery of Trust Translation Authorities, Project Deliverable, 2018.

[4] LIGHTest Consortium, “Architecture and Technical Coordination, Deliverable D2.14,” 2017.

[5] The LIGHTest Project, D3.2 - Conceptual Framework for Trust Schmes (2), Project Deliverable, 2018.

[6] European Parliament, 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 and repealing Directive 1999/93/EC, European Parliament, Brussels, Belgium, Regulation 910/201, 2014.

[7] T. L. Project, D2.7 -Relevant DNSSEC Concepts and Basic Building Blocks, Project Deliverable, 2017.

[8] FIDO Alliance, “FIDO UAF Protocol Specification,” 02 02 2017. [Online]. Available: https://fidoalliance.org/specs/fido-uaf-v1.1-id-20170202/fido-uaf-protocol-v1.1-id-20170202.html. [Accessed 21 06 2017].

[9] FIDO Alliance, “FIDO Metadata Statements,” 02 02 2017. [Online]. Available: https://fidoalliance.org/specs/fido-uaf-v1.1-id-20170202/fido-metadata-statement-v1.1-id-20170202.html. [Accessed 091 01 2018].

[10] The LIGHTest Project, D7.2 - Definition of device system architecture and derivation scheme of mobile IDs, Project Deliverable, 2017.

[11] LIGHTest Consortium, “Conceptual Framework for Trust Schemes, Deliverable D3.1,” 2017.

[12] The LIGHTest Project, D7.1 - Definition of Requirements for derivation and attestation of mobile IDs, Project Deliverable, 2017.

[13] NIST, “FIPS PUB 201-2: Personal Identity Verification (PIV) of Federal Employees and Contractors,” 2013. [Online]. Available: http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.201-2.pdf. [Accessed 2017].

[14] The LIGHTest Project, “D3.3 - DNS-based publication of Trust Schemes,” Project Deliverable, 2018.

[15] ETSI TS 119 612, “Electronic Signatures and Infrastructures (ESI);Trusted Lists,” European Telecommunications Standards Institute; Technical Specification, Sophia Antipolis Cedex, V2.1.1 (2015-07), 2015.

[16] S. Wagner, S. Kurowski, U. Laufs and H. Rossnagel, “A Mechanism for Discovery and Verification of Trust Scheme Memberships: The LIGHTest Reference Architecture,” in *L. Fritsch et al. (Eds.): Open Identity Summit, Lecture Notes in Informatics (LNI), Gesellschaft für Informatik*, Bonn, 2017.

[17] FIDO Alliance, “FIDO Metadata Service,” 02 02 2017. [Online]. Available: https://fidoalliance.org/specs/fido-uaf-v1.1-id-20170202/fido-metadata-service-v1.1-id-20170202.html. [Accessed 09 01 2018].

[18] ISO, “ISO/IEC 29115:2013 Information technology -- Security techniques -- Entity authentication assurance,” 2013. [Online]. Available: http://www.iso.org/iso/catalogue_detail.htm?csnumber=45138. [Accessed 2016].

11 Project Description

LIGHTest project to build a global trust infrastructure that enables electronic transactions in a wide variety of applications

An ever increasing number of transactions are conducted virtually over the Internet. How can you be sure that the person making the transaction is who they say they are? The EU-funded project LIGHT^{est} addresses this issue by creating a global trust infrastructure. It will provide a solution that allows one to distinguish legitimate identities from frauds. This is key in being able to bring an efficiency of electronic transactions to a wide application field ranging from simple verification of electronic signatures, over eProcurement, eJustice, eHealth, and law enforcement, up to the verification of trust in sensors and devices in the Internet of Things.

Traditionally, we often knew our business partners personally, which meant that impersonation and fraud were uncommon. Whether regarding the single European market place or on a Global scale, there is an increasing amount of electronic transactions that are becoming a part of peoples everyday lives, where decisions on establishing who is on the other end of the transaction is important. Clearly, it is necessary to have assistance from authorities to certify trustworthy electronic identities. This has already been done. For example, the EC and Member States have legally binding electronic signatures. But how can we query such authorities in a secure manner? With the current lack of a worldwide standard for publishing and querying trust information, this would be a prohibitively complex leading to verifiers having to deal with a high number of formats and protocols.

The EU-funded LIGHT^{est} project attempts to solve this problem by building a global trust infrastructure where arbitrary authorities can publish their trust information. Setting up a global infrastructure is an ambitious objective; however, given the already existing infrastructure, organization, governance and security standards of the Internet Domain Name System, it is with confidence that this is possible. The EC and Member States can use this to publish lists of qualified trust services, as business registrars and authorities can in health, law enforcement and justice. In the private sector, this can be used to establish trust in inter-banking, international trade, shipping, business reputation and credit rating. Companies, administrations, and citizens can then use LIGHTest open source software to easily query this trust information to verify trust in simple signed documents or multi-faceted complex transactions.

The three-year LIGHT^{est} project starts on September 1st and has an estimated cost of almost 9 Million Euros. It is partially funded by the European Union's Horizon 2020 research and innovation programme under G.A. No. 700321. The LIGHT^{est} consortium consists of 14 partners from 9 European countries and is coordinated by Fraunhofer-Gesellschaft. To reach out beyond Europe, LIGHT^{est} attempts to build up a global community based on international standards and open source software.

The partners are ATOS (ES), Time Lex (BE), Technische Universität Graz (AT), EEMA (BE), G+D (DE), Danmarks tekniske Universitet (DK), TUBITAK (TR), Universität Stuttgart (DE), Open Identity Exchange (GB), NLnet Labs (NL), CORREOS (ES), IBM Danmark (DK) and Ubisecure (FI). The Fraunhofer IAO provides the vision and architecture for the project and is responsible for both, its management and the technical coordination.

The Fraunhofer IAO provides the vision and architecture for the project and is responsible for both, its management and the technical coordination.