# D2.14

## Reference Architecture

| Document Identification | |
|---|---|
| **Date** | 28.02.2017 |
| **Status** | Final |
| **Version** | Version 1.0 |

| | | | |
|---|---|---|---|
| **Related WP** | WP 2 | **Related Deliverable(s)** | none |
| **Lead Authors** | Heiko Roßnagel | **Dissemination Level** | PU |
| **Lead Participants** | FHG | **Contributors** | see list in document |
| **Reviewers** | Georg Wagner (TUG), Charles Sederholm (GS) | | |

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

LIGHT$^{est}$ develops a lightweight trust infrastructure providing parties of electronic transactions with automatic validation of trust based on their individual trust policies. To ease integration and improve availability on any system, LIGHT$^{est}$ makes use of the existing global Domain Name System (DNS) for publication, querying, and cross-jurisdiction translation of information relevant to make such decisions, including levels of assurance. Building on top of the existing global infrastructure of the domain name system and explicit efforts to reach international acceptance enable LIGHT$^{est}$ to offer truly “global trust lists”.

Scope of this deliverable is the description of the architecture of LIGHT$^{est}$. It refers to the fundamental macroscopic system structures to be realized in LIGHT$^{est}$. This document does not cover more detailed aspects of the overall system such as implementation details or data structures which are described in the dedicated work packages as referred to in section 11.

This document describes architectural principles and both functional and technical goals addressed by the architecture and gives a high level explanation of all involved components to be specified in detail and developed in the specific work packages. It also provides a terminology in order to create a common understanding and gives an overview of standards that are relevant in the context of LIGHT$^{est}$. For a better understanding, this document also provides scenario descriptions for the basic functionality of LIGHT$^{est}$ and gives examples of more sophisticated scenarios such as realised in the pilots.

## 2. Project Description

**LIGHT$^{est}$ 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 people 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 LIGHT$^{est}$ 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 1$^{st}$, 2016 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 Globalsign (FI). The Fraunhofer IAO provides the vision and architecture for the project and is responsible for both, its management and the technical coordination.

# 3. Document Information

## 3.1 Contributors

| Name | Partner |
|---|---|
| Brügger, Beat Peter | FHG |
| Roßnagel, Heiko | FHG |
| Laufs, Uwe | FHG |
| Lorenzo Rosa | ATOS |
| Alberto Crespo | ATOS |
| Miryam Villegas | ATOS |
| Sebastian Mödersheim | DTU |
| Rasmus Birkedal | DTU |
| Edona Fasllija | TUBITAK |
| Muhammet Yildiz | TUBITAK |
| Sven Wagner | USTUTT |
| Sebastian Kurowski | USTUTT |

## 3.2 History

| Version | Date | Author | Changes |
|---|---|---|---|
| Draft 0.1 | 04.11.2016 | | First draft, document structure, first chapters |
| Draft 0.2 | 23.11.2016 | | Finalized architectural goals section |
| Draft 0.3 | 30.11.2016 | | Integrated partner contributions |
| Draft 0.4 | 06.12.2016 | | |
| Draft 0.5 | 15.12.2016 | | Integrated partner contributions |
| Draft 0.6 | 10.01.2017 | | Added scenarios section, integrated partner contributions |
| Draft 0.7 | 20.01.2017 | | Integrated partner contributions |
| Draft 0.8 | 03.02.2017 | | Integrated partner contributions |
| Draft 0.9 | 10.02.2017 | | Finalized, first complete “final” draft for internal reviews |
| Draft 1.00 | 27.02.2017 | | Final draft ready for submission |
| | | | |

# 4. Table of Contents

## 4.1 Table of Figures



## 4.2 Table of Acronyms

| | |
|---|---|
| ATV | Automatic Trust Verifier |
| AdES | Advanced Electronic Signatures |
| CAdES | CMS Advanced Electronic Signature |
| CH | Switzerland |
| CRL | Certificate Revocation List |
| DANE | DNS-based Authentication of Named Entities |
| DBMS | Datenbankmanagementsysteme |
| DNS | Domain Name System |
| DNSSEC | Domain Name System Security Extensions |
| DNSKEY | DNS Key |
| DP | Delegation Publisher |
| DPs | Delegation Publishers |
| DS | Delegation Signer |
| EC | European Commission |
| eIDAS | Electronic IDentification And Signature |

| | |
|---|---|
| ESI | Electronic Signatures and Infrastructures |
| FIPS | Federal Information Processing Standard |
| LoA | Level of Assurance |
| NSEC | Next SECure record |
| PAdES | PDF Advanced Electronic Signature |
| PKI | Public Key Infrastructure |
| RFC | Request for Comments |
| RRs | Resource Records |
| RRSIG | Resource Record Digital Signature |
| SHAs | Secure Hash Algorithms |
| SSH | Secure Shell |
| SSL | Secure Sockets Layer |
| TLS | Transport Layer Security |
| TPL | Trust Policy Language |
| TRL | Technology Readiness Level |
| TSPA | Trust Scheme Publication Authority |
| TTA | Trust Translation Authority |
| TTAs | Trust Translation Authorities |
| TTF | Trust Translation Framework |
| TTL | Trust Translation List |
| TTS | Trust Translation Scheme |
| UPU | Universal Post Union |
| URL | Uniform Resource Locator |
| US | United States |
| XAdES | XML Advanced Electronic Signature Time-Stamp Protocol |

# 5. Introduction

The overall focus of the LIGHT$^{est}$ project is to develop a lightweight trust infrastructure providing parties of electronic transactions with automatic validation of trust based on their individual trust policies. By using an existing infrastructure of the global DNS for publication, querying, and cross-jurisdiction translation of information relevant to make such decisions, including levels of assurance, LIGHT$^{est}$ wants to enable the use of truly “global trust lists”.

With this approach LIGHT$^{est}$ will basically provide an infrastructure to enable the most important principles and driving factors of eIDAS on a global level.

The architecture of LIGHT$^{est}$ defines the macroscopic design of this infrastructure and defines the overall system’s components, their functionality and their interaction on a high level view while a more detailed definition of the components will take place in the specific work packages. In section 6 of this document, we describe the scope of this deliverable and provide a terminology in section 7. Section 8 describes the architectural goals and principles applied in the architecture. Section 9 gives an overview of the standards related to the LIGHT$^{est}$ project and section 10 provides outline of the architecture. In section 11 and 12 we describe the components of LIGHT$^{est}$ and show their interaction in several scenarios.

# 6. Scope

Scope of this deliverable is a description of the architecture of LIGHT$^{est}$. It refers to the fundamental macroscopic system structures to be realized in LIGHT$^{est}$. This includes all high level specifications made with an abstraction on conceptual level or from a technical point of view on component level. This includes everything fundamental to understanding the overall system in its environment and excludes everything that is on a more detailed level.

This document does not address details of the single components on design or implementation level, which means that details such as data structures, data formats or details about the developed formal language(s) are not part of this deliverable but are part of the deliverables of the specific work packages. These work packages are referred to in section 11 for each component.

Starting with an overview of the architectural goals and concepts, the document creates a common understanding about the underlying concepts and the components within the project and for all stakeholders. This document also provides a terminology and component description in section 7. In order to demonstrate the component's functionality, the information flow and the interaction between the LIGHT$^{est}$ components, section 12 provides a high level description for the basic scenarios and examples of more sophisticated scenarios such as realised in the pilots.

# 7. Terminology

## 7.1 Purpose

Participants of LIGHT$^{est}$ use several terms that refer to concepts. A common understanding of important concepts is crucial for efficient communication among partners and thus the overall success of the project. For this reason, this section collects definitions of the most important concepts. It is intended to be used as glossary in the other deliverables.

## 7.2 Concepts of LIGHT$^{est}$

### 7.2.1 Entity

An Entity is a person, organization, or thing that is enrolled in a Trust Scheme and certain attributes of which are certified by a Trust Scheme Authority.

### 7.2.2 Trust Domain

A Trust Domain is a domain operated under the responsibility of a Trust Scheme Authority. It defines (typically by constraints) the set of Entities that are eligible to enroll, and the set of attributes that describes trust-relevant aspects of the enrolled Entities.

### 7.2.3 Trust List

A trust list provides relevant attributes of enrolled entities. A trust lists is usually signed by an issuing authority with an electronic signature to prove their trustworthiness.

Different types of trust lists do exist. For example, a boolean trust list provides a boolean value for each entity. An entity can either be trusted or not trusted. As another example, an ordinal trust list provides an ordinal value for each entity. Typically, typical value for an ordinal value is a Level of Assurance (LoA).

### 7.2.4 Trust Scheme

A Trust Scheme is operated by a Trust Scheme Authority and comprises the organizational, regulatory/legal, and technical measures to assert trust-relevant attributes about enrolled Entities in a given domain of trust. A Trust Scheme operates in a given Trust Domain and typically has a declared or implied purpose.

Regarding the operator of a given Trust Scheme Authority, there are two major types of Trust Schemes:

**Authority-Based Trust Schemes:** In Authority-Based Trust Schemes, an Authority issues regulations and conditions that are necessary for them to certify certain attributes. Often the Trust Scheme Authority uses supervision to ascertain that an Entity complies with all regulations and conditions. Another organizational function of the Authority-Based Trust Schemes is the enrollment of eligible Entities.

**Reputation-Based Trust Schemes:** In Reputation-Based Trust Schemes, a third party trusted by verifiers collects and publishes reputation data on Entities. The current status of a Trust Scheme is described by its Trust Scheme Data that is typically published in one or multiple ways (see Trust Scheme Publication).

**Trust Scheme Data:** Trust Scheme Data represent the current state of a Trust Scheme. It is a dataset managed by the Trust Scheme Authority, for example in a DBMS, typically recording also change events over time in order to allow to reconstruct the past state of this data.

**Trust Scheme Publication:** A Trust Scheme Publication makes the complete or a subset of the Trust Scheme Data available to verifiers. A Publication uses a specific means of communications including Trust Lists and LIGHT$^{est}$ Infrastructure.

A Publication can contain different aspects of the Trust Scheme Data including:

- **Historical publications:** These include the full set of change events and make it possible to determine the status of the Trust Scheme Data at different positions in time.
- **Snapshot publications:** Report the status of the Trust Scheme data at a given point in time.
- **Sampled publications:** Report the state of the Trust Scheme Data at the point of time when it was last queried.
- **Real time publications:** Report the state of the Trust Scheme Data at the point of time of a query.

**Temporal Model:** The *Temporal Model* is used for the retrieval of current data (present) and historic data (past). The temporal model is also responsible for the description of data freshness. Using *Sampled Publications* is a way to keep and provide historic data. Boolean and ordinal Trust Lists are typically sampled snapshot publications. Since the sampling point lies back in time, their validity period is typically limited to mitigate the effects of potential changes since the snapshot was taken.

A publication of a Trust Scheme on the LIGHT$^{est}$ trust infrastructure may be either real time, if the Trust Scheme Data is directly queried, or sampled, in case that the LIGHT$^{est}$ publication is simply based on (sampled) Trust Lists.

Based on the data of a Trust Scheme Publication, it's possible to distinguish between three basic types:

**Boolean Trust Scheme Publications:** This type of publication returns a Boolean result (True or False). As an implementation convention, instead of explicitly stating the Boolean value, every Entity listed in a publication is usually assumed to have the same value: True (=trusted) in the case of white lists, False (=untrusted) in the case of black lists.

**Ordinal Trust Scheme Publications:** This type of publication returns an ordinal value, which is contained by an enumeration of valid values. A typical example of an ordinal publication are Levels of Assurance or a reputation ranking. The defined result value sets for example could be:

- [low, medium, high]
- [level1, level2, level3, level4]
- [0-stars, 1-star, 2-stars, 3-stars, 4-stars, 5-stars]

**Generic Trust Scheme Publications*:*** All other Trust Scheme Publications, for which a Boolean of Ordinal representation of the contained data is insufficient, can be realized as a Generic Trust Scheme Publication. This publication type can define any tuple of attributes required to describe the properties of a given entity. An example of a Generic Trust Scheme Publication is a business register that lists a set of relevant attributes for a company, such as for example:

- date of foundation
- legal form of company
- social capital

Note that Boolean and Ordinal Trust Scheme Publications are special cases of the Generic Trust Scheme Publication. Though, these two special types are assumed to occur very frequently which makes it important to distinguish between them for better ease of use and ease of implementation.

**Trust Scheme Publication Authoritiy:** A LIGHT$^{est}$ Scheme Publication Authority is a DNSSEC software with an enhancing layer to fulfill the LIGHT$^{est}$ purposes. Mechanisms to publish the Trust Schemes using the existing ones of the DNS system, are to be developed.

### 7.2.5 Electronic Transaction

An Electronic Transaction is the object whose trustworthiness is evaluated by a verifier.

The simplest possible Electronic Transaction is a single document that is cryptographically associated with an electronic identity, e.g., through the mechanism of electronic signature.

In the more general case, an Electronic Transaction is a container (of a given format) that contains several documents or sub-containers. Optionally, documents and containers are associated with an electronic identity, e.g., via electronic signature.

An Electronic Transaction or the digital identities (e.g., certificates) associated with its parts can also contain "Discovery Data" such as membership claims that point to the Trust Schema that certifies its trustworthiness.

An example of an Electronic Transaction is a purchase order: A container contains the actual purchase order, signed by the issuing organization (with a seal) or by an employee who is authorized by a delegation. In addition, the container contains a letter of credit signed by a bank. All kind of additional "attachments" can be imagined.

### 7.2.6 Trust Policy

A Trust Policy is a recipe that takes an Electronic Transaction and potentially multiple Trust Schemes as input and creates a single Boolean value (trusted [yes/no]) as output. A Trust Policy

can optionally return a documentation of the result which for example is an explanation why the Electronic Transaction is not trustworthy.

### 7.2.7 Trust Policy Language

A Trust Policy Language is a formal language with well-defined semantics that is typically based on a mathematical formalism and is used to express the recipe of a trust policy. To serve its role in LIGHT$^{est}$, a trust policy language must have the following elements:

- a mechanism to uniquely identify trust schemes (e.g., through a Fully Qualified Domain Name of the DNS)
- a mechanism to look up an entity (electronic identity) in a given trust scheme
- a mechanism to translate trust schemes across trust domains
- a mechanism to express that delegation is allowed and how many delegation steps are admissible.
- a mechanism to address the various parts within an electronic transaction (to express constraints on these parts)
- mechanisms to extract specific data values from parts that are contained in an electronic transaction

The current suggestion to achieve this is a simple but powerful language in the style of Prolog/Horn clauses. The advantages of this language are:

- It is trivial to formalize all simple policies that are based on a kind of enumeration
- It offers an easy mechanism to describe relations between concepts, e.g. what criteria need to be satisfied to fulfill a certain standard, logical combinations of policies (and/or/not)
- It is ideal for concepts like delegation and black-listing (for this reason for instance the access control policy languages like SECPAL and DKAL by Microsoft are based similarly on Horn clauses)
- The language comes directly with a clear formal meaning including an evaluation procedure, i.e., specifications are directly "executable".
- The language is powerful enough because it is Turing complete (every computable policy can be expressed).
- The evaluation can be made to directly trigger necessary queries to servers, e.g., using DNSSec, and process their answer; thus the bulk of the ATV can directly be encoded into the language, either as a prototype/testing reference or even as the final product.
- For the average users we can either provide design patterns for their policy or even interface to a simpler (possibly graphical) language that they can use more intuitively but that is limited in expressive power. In this way one may be able to use LIGHT$^{est}$ without any learning curve in 99% of all cases, but when one wants to express something really non-standard (the remaining 1% of cases), the language still allows that.

### 7.2.8 Trust Translation Scheme

A trust translation scheme is a function, defined on the base of a legally accepted policy that enables the interoperability of trust schemes published by different entities and communities (and possibly across different trust domains) by defining the relationships between the levels of trust of the different trust schemes. To allow interoperability, agreements between different communities on how to translate between their concepts are encouraged: for example a translation from a trust scheme with levels {1,2,3} to one with levels {A,B,C,D}.

LIGHT$^{est}$ requires a mechanism to refer to trust schemes that are translated according to a given trust translation scheme. This is highly specific to LIGHT$^{est}$ and it is very unlikely to find an existing language mechanism that fits. This required language element can be satisfied with simple constructs. For example, a “native” trust scheme could be constructed by a function such as “translate(<foreign trust scheme id>, <translation scheme id>)”.

### 7.2.9 Trust Translation Framework

A *Trust Translation Framework* is the meta trust translation scheme, the conceptual framework to represent arbitrary trust translation schemes.

The LIGHT$^{est}$ TTF consists of suitable translation algorithms, abstract syntax for expressing the different formats of the trust translation lists, legal aspects, etc. This TTF is supporting all the rationale about translation in LIGHT$^{est}$. Trust translation is considered under different types of trust schemes of all types (Boolean, ordinal and generic).

### 7.2.10 Trust Translation List

A Trust Translation List is comprised of trust policies (translation algorithms) and trust schemes. It is a ternary list of (trustPolicy, sourceSchema, targetSchema). The LIGHT$^{est}$ TTF provides trust translation lists.

### 7.2.11 Trust Translation Authority

LIGHT$^{est}$ Trust Translation Authority is a DNSSEC software with an enhancing layer to aim the LIGHT$^{est}$ purposes. Mechanisms to publish the trust translation schemes using the existing ones of the DNS system, are to be developed.

Discovery structures based on DNS, are to be studied in order to be used by verifiers to discover Trust Translation Authorities but the goal is not let anything be discovered but directly provided by the TTA.

**Related to publishing to a TTA:** Different approaches are needed for simple and historical Trust Schemes. A suitable sub-domain structure has to be designed in order to facilitate querying Trust Translation data. The design has to use the standard DNS delegation mechanisms for support of the subsidiarity principle which is often applied by Trust Scheme Authorities.

**Related to the discovery structures*:*** The necessity of finding the appropriate Trust Translation Authority in the global namespace of domain names makes the creation of discovery

mechanisms essential. These will include pointers from Trust Scheme Publication Authorities as well as index domains operated by Trust Translation Authorities.

# 8. Architectural Principles and Goals

The LIGHT$^{est}$ architecture addresses both organisational and technical goals resulting from the requirement of enabling a globally scalable trust infrastructure that integrates the existing technical and organisational environment considering the given constraints.

## 8.1 General Principles and goals

The LIGHT$^{est}$ architecture follows several general principles, which are:

**Subsidiarity and no change of data ownership:** Following the principle of subsidiarity, a main goal of the LIGHT$^{est}$ architecture is to consequently enable participants to stay in control regarding all relevant areas, such as data ownership or trust decisions. Especially, to avoid issues regarding data ownership and responsibility for data, the ownership has to be unchanged and publishers stay in control and in responsibility of their own data.

**Minimisation of adoption barriers:** In order to achieve transfer the results of the project into the existing real world environment, the architecture has the goal to reduce organisational adoption barriers as well as technical barriers regarding integration into the existing technical environment. Furthermore, the architecture aims at simplicity and structural consistency (self-documentation) in order to achieve an easier adoption of the technology.

**Reuse of existing software and infrastructure:** To lower technical adoption barriers and to reach the expected level of technology readiness, the LIGHT$^{est}$ architecture has the goal to make extensive use of existing infrastructure. The use of mature and widespread technology is intended to reduce efforts regarding the utilisation of LIGHT$^{est}$ for organisations intending to publish their trust data. The reuse of the existing DNS infrastructure, including the global agreement of its governance and management of its existing single, global trust root, the world-wide organization composed of name registries, the global highly available infrastructure consisting of root and top-level-domain servers, the mature and well-tested communications protocols, and a variety of compliant DNS servers from multiple vendors including open source offerings. Organizations that intend to publish trust schemes, translations schemes, and/or delegations can reuse their existing DNS servers (with security extension) or the existing outsourcing of this functionality. Furthermore, the reuse of DNS, well-known protocols and existing software aims at the goal of increasing acceptance and lowering adoption barriers on the user side.

**Separation of concerns and abstraction:** The architecture has to follow a modular approach (Edsger et al., 1976) to allow the distribution of efforts among the consortium and achieve an easier collaboration by the reduction of technical dependencies. In addition, modularity can help

to reduce efforts for maintenance and optimization of the different components, e.g. by allowing a reduction of complexity and information hiding (Parnas, 1972).

A related approach followed by the architecture is the reduction of complexity by abstraction (Shaw et al.,1995). Hiding details of the specific implementation of specific system components allows easier maintenance as well as easier usage by other components.

## 8.2 Technical Principals and Goals

The architecture especially focusses on the following technical goals:

**Distributed System:** Considering the general principles and requirements described below, the architecture has to be distributed to allow the realisation of the required separations e.g. to allow subsidiary, distributed ownership of data and distributed control and responsibilities.

**Extensibility:** Because of the wide range of possible applications and the variety of LIGHT$^{est}$ users, an incremental approach is required regarding the extension the overall system, e.g. by realising new use cases or extending the amount of users. It is important that the overall system can handle these extensions without modifications on architecture level.

**Scalability:** Because of the large number of possible users and application fields, one of the main technical goals is scalability (Tanenbaum et al., 2007).

**Security Screening:** In the domain of trust infrastructure, security of course is vital. In addition to the solutions that inherently are part of LIGHT$^{est}$, there also have to be mechanisms that allow security screening.

**Fault tolerance and high availability:** Addressing the large scale overall approach of LIGHT$^{est}$ and its relevance for the targeted use cases and applications, high availability and fault tolerance have to be addressed by the architecture (Lee, Anderson., 2012).

**Maturity:** Given the targeted TRL of this project, a high level of maturity has to be considered in the determinations of the LIGHT$^{est}$ architecture. This concern for example is addressed in the architecture by the extensive reuse of mature technology.

**Traceability:** In the domain of trust decisions, it is required to be able to trace and reconstruct the base of these decisions. According to that, traceability of historic information is an important requirement.

# 9. Standards

This section gives an overview of the international standards that apply to the LIGHT$^{est}$ architecture, with a brief explanation of their role in the architecture.

## 9.1 Level of Assurance

The following important standard defines the level of assurance that is used for entity authentication and checking trust compliance in ordinal-type trust schemes used throughout the LIGHT$^{est}$ architecture.

Involved components: all except 11.3, 11.6, and 11.9.

**ISO/IEC 29115:2013** Information technology -- Security techniques -- Entity authentication assurance framework

This standard provides a framework for managing entity authentication assurance in a given context. In particular, it

- specifies four levels of entity authentication assurance
- specifies criteria and guidelines for achieving each of the four levels of entity authentication assurance
- provides guidance for mapping other authentication assurance schemes to the four LoAs
- provides guidance for exchanging the results of authentication that are based on the four LoAs
- provides guidance concerning controls that should be used to mitigate authentication threats.

## 9.2 DNS and DNSSEC Extensions

The following important standards apply to the whole LIGHT$^{est}$ architecture as they define the DNS and the DNSSEC extensions upon which the architecture is based; in particular, the resource records definitions crucial for trust scheme publication (WP3), translation (WP4), delegation (WP5), and verification (WP6). DNSSEC uses cryptographic keys and digital signatures to provide authentication of DNS data. Information that is retrieved from the DNS and that is validated using DNSSEC is thereby proved to be the authoritative data.

Involved components: all except 11.10, 11.11, 11.12, and 11.13.

**RFC1035** Domain names - implementation and specification. P.V. Mockapetris. November 1987. (Format: TXT=125626 bytes) (Obsoletes RFC0973, RFC0882, RFC0883) (Updated by RFC1101, RFC1183, RFC1348, RFC1876, RFC1982, RFC1995, RFC1996, RFC2065, RFC2136, RFC2181, RFC2137, RFC2308, RFC2535, RFC2673, RFC2845, RFC3425,

RFC3658, RFC4033, RFC4034, RFC4035, RFC4343, RFC5936, RFC5966, RFC6604, RFC7766) (Also STD0013) (Status: INTERNET STANDARD) (DOI: 10.17487/RFC1035)

This RFC (with the corresponding updates) describes the details of the Domain Name System and protocol, providing a mechanism for naming resources in such a way that the names are usable in different hosts, networks, protocol families, internets, and administrative organizations.

**RFC4033** DNS Security Introduction and Requirements. R. Arends, R. Austein, M. Larson, D. Massey, S. Rose. March 2005. (Format: TXT=52445 bytes) (Obsoletes RFC2535, RFC3008, RFC3090, RFC3445, RFC3655, RFC3658, RFC3755, RFC3757, RFC3845) (Updates RFC1034, RFC1035, RFC2136, RFC2181, RFC2308, RFC3225, RFC3597, RFC3226) (Updated by RFC6014, RFC6840) (Status: PROPOSED STANDARD) (DOI: 10.17487/RFC4033)

This RFC (with the corresponding updates) introduces the Domain Name System Security Extensions (DNSSEC), adding data origin authentication and data integrity to DNS, describing their capabilities and limitations, also discussing the services provided.

**RFC4034** Resource Records for the DNS Security Extensions. R. Arends, R. Austein, M. Larson, D. Massey, S. Rose. March 2005. (Format: TXT=63879 bytes) (Obsoletes RFC2535, RFC3008, RFC3090, RFC3445, RFC3655, RFC3658, RFC3755, RFC3757, RFC3845) (Updates RFC1034, RFC1035, RFC2136, RFC2181, RFC2308, RFC3225, RFC3597, RFC3226) (Updated by RFC4470, RFC6014, RFC6840, RFC6944) (Status: PROPOSED STANDARD) (DOI: 10.17487/RFC4034)

This RFC (with the corresponding updates) defines the public key (DNSKEY), delegation signer (DS), resource record digital signature (RRSIG), and authenticated denial of existence (NSEC) resource records, describing the purpose and format of each resource record with examples.

**RFC4035** Protocol Modifications for the DNS Security Extensions. R. Arends, R. Austein, M. Larson, D. Massey, S. Rose. March 2005. (Format: TXT=130589 bytes) (Obsoletes RFC2535, RFC3008, RFC3090, RFC3445, RFC3655, RFC3658, RFC3755, RFC3757, RFC3845) (Updates RFC1034, RFC1035, RFC2136, RFC2181, RFC2308, RFC3225, RFC3597, RFC3226) (Updated by RFC4470, RFC6014, RFC6840) (Status: PROPOSED STANDARD) (DOI: 10.17487/RFC4035)

This RFC (with the corresponding updates) describes the DNSSEC protocol modifications, defining the concept of a signed zone, along with the requirements for serving and resolving by using DNSSEC, and allowing a security-aware resolver to authenticate both DNS resource records and authoritative DNS error indications.

**RFC4025** A Method for Storing IPsec Keying Material in DNS. M. Richardson. March 2005. (Format: TXT=25408 bytes) (Status: PROPOSED STANDARD) (DOI: 10.17487/RFC4025)

This RFC describes a new resource record for storing public keys for use in IP security (IPsec) systems. The record also includes provisions for indicating what system should be contacted when an IPsec tunnel is established with the entity in question.

**RFC4255** Using DNS to Securely Publish Secure Shell (SSH) Key Fingerprints. J. Schlyter, W. Griffin. January 2006. (Format: TXT=18399 bytes) (Status: PROPOSED STANDARD) (DOI: 10.17487/RFC4255)

This RFC describes a method of verifying Secure Shell (SSH) host keys using Domain Name System Security (DNSSEC), also defining a new DNS resource record that contains a standard SSH key fingerprint.

### 9.3 DANE protocol

The following standards apply to the whole LIGHT$^{est}$ architecture, in particular for WP3 to WP6 and define the DANE protocol, which adds TLS encryption to DNSSEC. DANE allows DNS, secured by DNSSEC, to indicate which TLS/X.509 certificate is the right one to use according to the right trust anchor. This reduces the attack footprint of TLS significantly. A new DNS resource record TLSA is defined and indicates the correct server certificate. It must be DNSSEC signed to be valid.

Involved components: all except 11.10, 11.11, 11.12, and 11.13.

**RFC6698** The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA. P. Hoffman, J. Schlyter. August 2012. (Format: TXT=84034 bytes) (Updated by RFC7218, RFC7671) (Status: PROPOSED STANDARD) (DOI: 10.17487/RFC6698)

This RFC (with the corresponding updates) defines the DANE protocol for DNS, enabling administrators of domain names to specify the keys used in that domain's TLS servers, improving on previous protocols which relied on third parties to certify the keys, without requiring changes in the TLS server software.

**RFC7671** The DNS-Based Authentication of Named Entities (DANE) Protocol: Updates and Operational Guidance. V. Dukhovni, W. Hardaker. October 2015. (Format: TXT=80496 bytes) (Updates RFC6698) (Status: PROPOSED STANDARD) (DOI: 10.17487/RFC7671)

This RFC clarifies and updates the DANE TLSA specification (RFC 6698), based on subsequent implementation experience, also giving guidance for implementers, operators, and protocol developers who want to use DANE records.

**RFC6394** Use Cases and Requirements for DNS-Based Authentication of Named Entities (DANE). R. Barnes. October 2011. (Format: TXT=29477 bytes) (Status: INFORMATIONAL) (DOI: 10.17487/RFC6394)

This RFC describes a set of use cases in which the DNS and DNS Security Extensions (DNSSEC) could be used to make assertions that support the TLS authentication process,

focusing on TLS server authentication, but also covering TLS client authentication for applications where TLS clients are identified by domain names.

### 9.4 Certificate formats

The following standard specifies the formats for public key certificates, certificate revocation lists, attribute certificates, and the certification path validation algorithm, that are used throughout the whole LIGHT$^{est}$ architecture to validate electronic transactions and resource records.

Involved components: all except 11.10 and 11.11.

**RFC5280** Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, W. Polk. May 2008. (Format: TXT=352580 bytes) (Obsoletes RFC3280, RFC4325, RFC4630) (Updated by RFC6818) (Status: PROPOSED STANDARD) (DOI: 10.17487/RFC5280)

This RFC (with the corresponding updates) profiles the X.509 v3 certificate and X.509 v2 certificate revocation list (CRL) for use in the Internet.

### 9.5 Secure connection methods

The following standard describes the secure connection method used by the verifier (WP6) and the resource record authoring tools (WP3, 4, 5) to query and manage the LIGHT$^{est}$ architecture records within the DNS.

Involved components: 11.1, 11.2, 11.4, 11.5, 11.7, 11.8, and 11.14.

**RFC2818** HTTP Over TLS. E. Rescorla. May 2000. (Format: TXT=15170 bytes) (Updated by RFC5785, RFC7230) (Status: INFORMATIONAL) (DOI: 10.17487/RFC2818)

This RFC (with the corresponding updates) describes how to use TLS to secure HTTP connections over the Internet, documenting the practice of layering HTTP over SSL (the predecessor to TLS) and distinguishing secured traffic from insecure traffic by the use of a different server port.

### 9.6 Hashing algorithms

The following standard also applies from WP3 to WP6 and defines the hashing algorithms employed to match the signature in the certificate of the DANE TLSA record.

Involved components: all except 11.10 and 11.11.

**RFC6234** US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF). D. Eastlake 3rd, T. Hansen. May 2011. (Format: TXT=236573 bytes) (Obsoletes RFC4634) (Updates RFC3174) (Status: INFORMATIONAL) (DOI: 10.17487/RFC6234)

This RFC defines the suite of Secure Hash Algorithms (SHAs), including four beyond SHA-1, as part of a Federal Information Processing Standard (FIPS), namely SHA-224, SHA-256, SHA-

384, and SHA-512. It supplies samples of open source code performing these SHA hash functions, supporting input strings of arbitrary bit length.

## 9.7 Advanced electronic signatures

The three following standards cover advanced electronic signatures (AdES) supported by PKI and public key certificates, and can be used for any transaction between an individual and a company, between two companies, between an individual and a governmental body, etc. They are environment-independent and can be applied to any environment e.g. smart cards, GSM SIM cards, special programs for electronic signatures, etc. They apply to the signature of the electronic transaction document to be examined in the ATV (WP6), according to the CMS, XML, or PDF format.

Involved components: 11.12, 11.13, and 11.14.

**ETSI EN 319 122-1 V1.1.1 (2016-04)** Electronic Signatures and Infrastructures (ESI); CAdES digital signatures; Part 1: Building blocks and CAdES baseline signatures.

This standard meets applicable requirements from Regulation (EU) No 910/2014 [i.13].

**ETSI EN 319 132-1 V1.1.1 (2016-04)** Electronic Signatures and Infrastructures (ESI); XAdES digital signatures; Part 1: Building blocks and XAdES baseline signatures.

This standard meets applicable requirements from Regulation (EU) No 910/2014 [i.1].

**ETSI EN 319 142-2 V1.1.1 (2016-04)** Electronic Signatures and Infrastructures (ESI); PAdES digital signatures; Part 2: Additional PAdES signatures profiles.

This standard includes evidence as to the signature's validity even if the signer or verifying party later attempts to deny (i.e. repudiates; see ISO/IEC 10181-4 [i.1]) the validity of the signature. Thus, this standard can be used for any document encoded in portable document format (PDF).

## 9.8 Discovery mechanisms

The following documents investigate discovery mechanisms, including DNS/DNSSEC, and are useful to identify needs and requirements for a global discovery mechanism, such as the LIGHT$^{est}$ architecture aims to implement.

Involved components: 11.3, 11.6, and 11.9.

**ETSI GS INS 006 V1.1.1 (2011-11)** Identity and access management for Networks and Services; Study to Identify the need for a Global, Distributed Discovery Mechanism

This document investigates the current landscape on the identity management area and evaluates the need for a new discovery mechanism, focusing on gap analysis for a global distributed discovery mechanism of identifiers, providers and capabilities.

**ETSI GS INS 010 V1.1.1 (2014-03)** Identity and access management for Networks and Services; Requirements of a global distributed discovery mechanism of identifiers, providers and capabilities

This document investigates the requirements of a global discovery mechanism of identifiers, providers and capabilities and examines if any existing systems or mechanism meet these requirements and can - fully or partially - support the design of its architecture.

### 9.9 Trust Policy Languages

In the area of trust management, more and more developers and researchers begin to understand that we need to describe not only concrete trust lists and the like, but that we need to describe precisely our policy of trust in order to reduce the size of specification, and rather make the rationale behind a trust policy visible and accessible to (automated) reasoning. This is a similar development that we have seen in access control where it is beneficial to move from simple flat access control lists to more sophisticated access control policy models and languages. However, while there are already many models for trust policies, there are only few actual languages to describe such models. The most relevant such language is in our opinion the language proposed by De Capitani et al. (S. De Capitani, 2012). They define an SQL-style language in order to integrate trust management with data base management (in particular classical access control management). The idea is that the entire design is geared towards relational databases and the use in web applications. Note that the terms *trust management* and *trust policy* are still very close to access control in this work. Basic concepts of their language are:

- authorities: identities who can issue certificates (e.g. department of health)
- authority classes: classes of authorities characterized by certain attributes (e.g. class hospital)
- trust tables: characterizing a class of identities (e.g. physicians of a hospital) by the authoritative classes who can the certify the attributes of the trust table (e.g. class hospital)
- (Worachet Uttha, 2014 ) trust policy: associates subjects a particular role (for access control) based on certified attributes (e.g. cardiologist as a physician with specialization attribute cardiology).

Besides this work by De Capitani there are similar works that seem to be subsumed. To mention are here the number of works that are only indirectly dealing with trust management, by defining access control policies (Worachet Uttha, 2014 ). While such languages are thus not directly applicable in LIGHT$^{est}$, our language design draws a lot from them, due to many similar traits such as defining policies (based on properties of the subjects), the fluctuation (subjects enter and leave or delegate), and the need to evaluate the policy to obtain a yes/no decision. Like the language we propose, many such works employ a Prolog/Horn-clause representation (Gurevich, 2008), i.e., clauses of the form "goal <= premises" thus positively defining that a goal (like a level of assurance) is fulfilled if all of the given premises are fulfilled (which may trigger further evaluations). Clauses are not exclusive, i.e., when the premises are not fulfilled, one can try the

next one that matches one's goal. The advantages are that this representation is for many cases easily readable ("logical") and can be directly executed by an interpreter.

Involved components: 11.10, 11.11, and 11.14.

# 10. Overview of LIGHT$^{est}$ Reference Architecture

This section gives an overview of the LIGHT$^{est}$ architectural components. Figure 1: The LIGHT$^{est}$ Reference Architecture shows the LIGHT$^{est}$ reference architecture with all the major software components. It illustrates how a verifier can validate a received Electronic Transaction based on her individual Trust Policy and queries to the LIGHT$^{est}$ reference trust infrastructure.

Verifiers use Policy Authoring and Visualization Tools to state their individual Trust Policy. These tools support non-technical decision makers understanding and creating Trust Policies that can be applied by the ATV. In a cross-jurisdiction setting, different trust schemes are used to describe conceptually equivalent aspects. To make it easy to verifiers, TTAs, provide the necessary translation data to map the levels of assurance of the foreign Trust Scheme to its equivalent in the domestic Trust Scheme. For example, an American authentication security of Level 3 could be mapped to the eIDAS level substantial. Very often, data records that compose an Electronic Transaction are not directly signed by the legal entity responsible for it (e.g., using a company seal), but by a natural person that acts as an authorized representative for the former based on a delegation. The architecture therefore foresees the component of DPs that permit verifiers to query delegations and mandates. All server components are implemented as DNS name servers. Organizations intended to publish Trust Schemes, Translations Schemes, and/or Delegations can reuse their existing DNS servers (with security extension) or the existing outsourcing of this functionality. In the same way as the DANE standard RFC7671 (see section 9.3) uses the DNS security extension to derive trust in TLS server certificates, LIGHT$^{est}$ derives trust in Trust Scheme, Translation, and Delegation data. Chains of trust can be stored as receipts that can be validated at a later point in time.

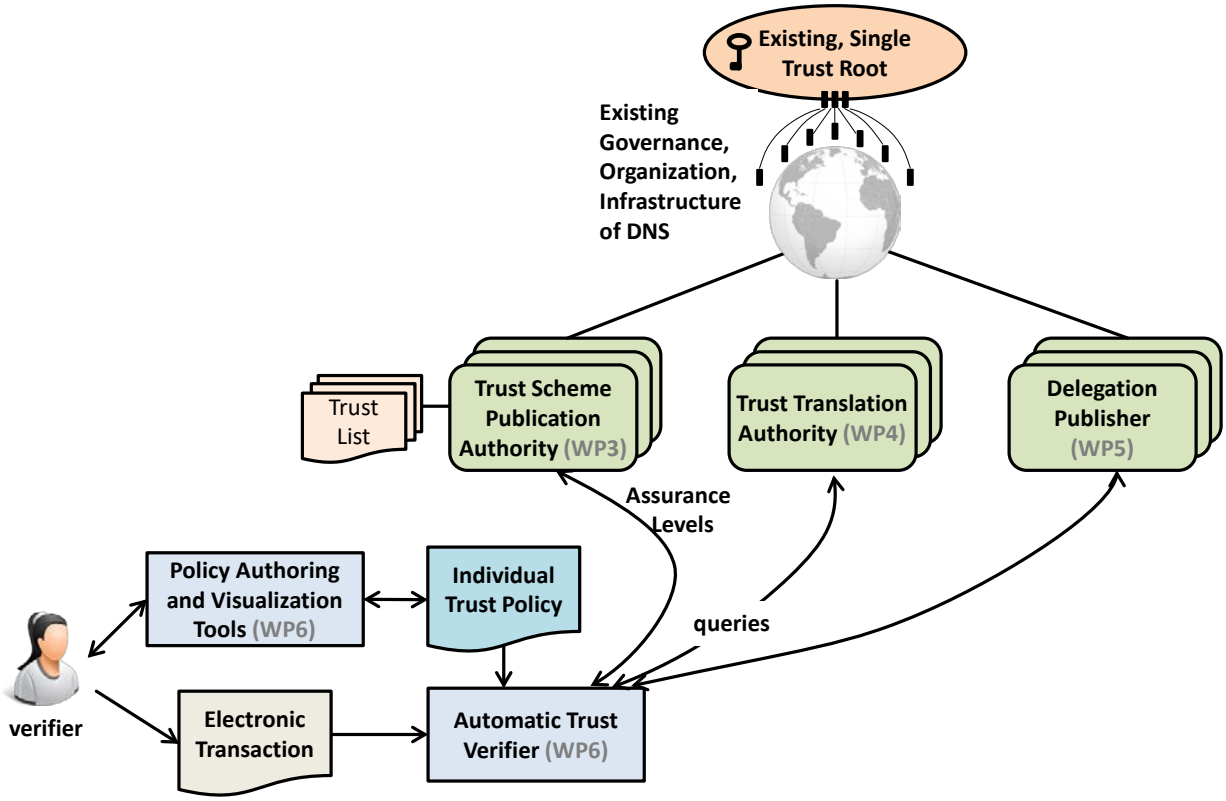


Figure 1: The LIGHT$^{est}$ Reference Architecture

# 11. Description of Components

This section describes all architectural components of LIGHTest in detail.

## 11.1 Trust Scheme Publication Authority (TSPA)

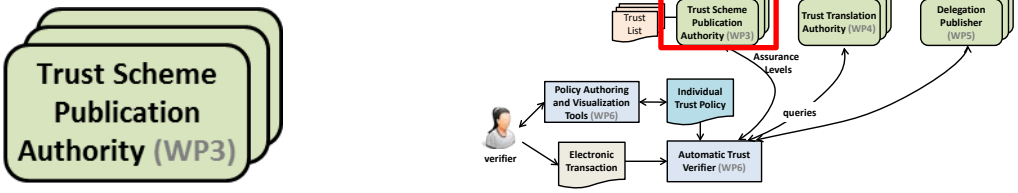


**Figure 2: Trust Scheme Publication Authority**

A Trust Scheme Publication Authority (see Figure 2: Trust Scheme Publication Authority) operates an off-the-shelf DNS Name Server with DNSSEC extension. A server publishes multiple Trust Lists under different sub-domains of the Authority's domain name. Further information on TSPA will be provided in the deliverables of Work Package 3.

## 11.2 Utilities to Load selected Trust Lists into a TSPA

The utilities parse selected Trust List formats and write or load equivalent DNS Zone files. The "zone file writer" sub-component can be used for multiple utilities and expose a conceptual view that is defined in Task 3.1 of the Work Package 3. The "parser" sub-component is specific to the Trust List format.

## 11.3 Utilities to Construct DNS-based Discovery Structures for TSPAs

The utilities to construct DNS-based discovery structures for Trust Scheme Publication Authorities will be presented in detail in the Work Package 3. One of the examples of the tasks that perform such utilities is the construction of index domains or zones.

## 11.4 Trust Translation Authority (TTA)

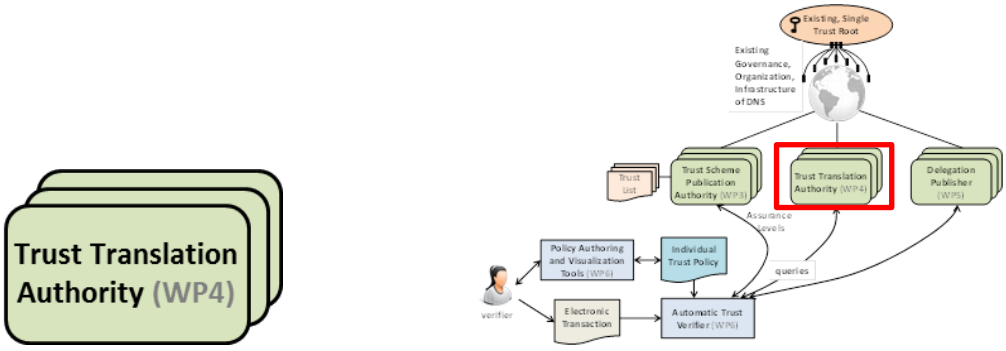


**Figure 3: Trust Translation Authority**

A Trust Translation Authority (see Figure 3: Trust Translation Authority) operates a standard DNS Name Server with DNSSEC extension. A server publishes trust data under different sub-domains of the Authority's domain name. Trust Translation Lists express which authorities from other Trust Domains are trusted. Trust Translation Authority will be described in details in the deliverables of the Work Package 4.

## 11.5 Utilities to Load selected Trust Translation Data into a TTA

The utilities parse and query input data and write or load equivalent DNS Zone files. The "zone file writer" sub-component can be used for multiple utilities and expose a conceptual view that is defined in Task 3.1. More information will be provided in the Work Package 4. The "parser/query" sub-component is specific to the data format.

## 11.6 Utilities to Construct DNS-based Discovery Structures for TTAs

The utilities to construct DNS-based discovery structures for Trust Translation Authorities will be presented and described in the Work Package 4. One of the examples of the tasks that perform these utilities is the construction of index domains or zones.

## 11.7 Delegation Publisher (DP)


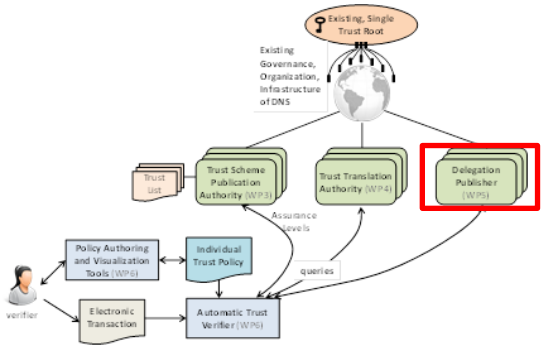


**Figure 4: Delegation Publisher**

A Delegation Publisher (see Figure 4: Delegation Publisher) operates an off-the-shelf DNS Name Server with DNSSEC extension. A server publishes multiple delegations under different sub-domains of the organization's domain name. Further information on DP will be provided in the deliverables of the Work Package 5.

## 11.8 Utilities to Load selected Delegation Data into a DP

The utilities parse and query input data and write or load equivalent DNS Zone files. The "zone file writer" sub-component can be used for multiple utilities and expose a conceptual view that is defined in Task 3.1. More information will be provided in the Work Package 5. The "parser/query" sub-component is specific to the data format.

## 11.9 Utilities to Construct DNS-based Discovery Structures for DPs

The utilities to construct DNS-based discovery structures for Delegation Publishers will be presented and described in the Work Package 5. One of the examples of the tasks that perform these utilities is the construction of index domains or zones.

## 11.10 Policy Authoring and Visualization Tools

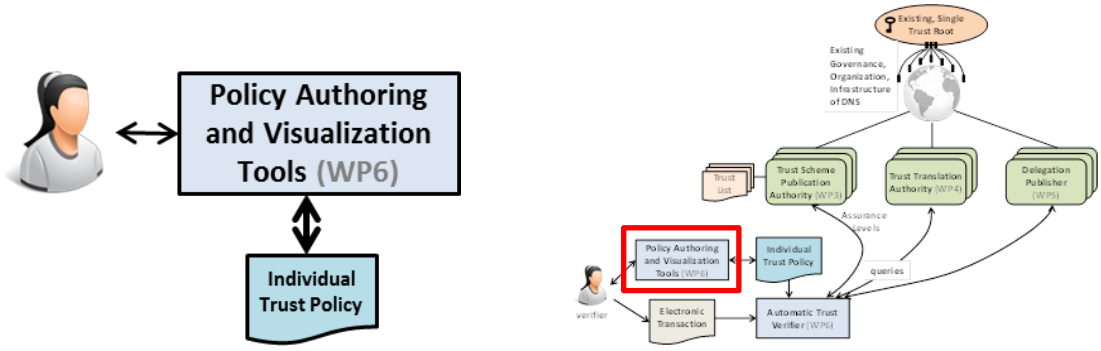


**Figure 5: Policy Authoring and Visualization Tools**

Policy Authoring and Visualization Tools (see Figure 5: Policy Authoring and Visualization Tools) form a part of an interactive software, for example one of several desktop/web applications. They make it easy for non-technical users to visualize and edit a Trust Policy. More information on Policy Authoring and Visualization Tools will be provided in the Work Package 6.

## 11.11 Trust Policy

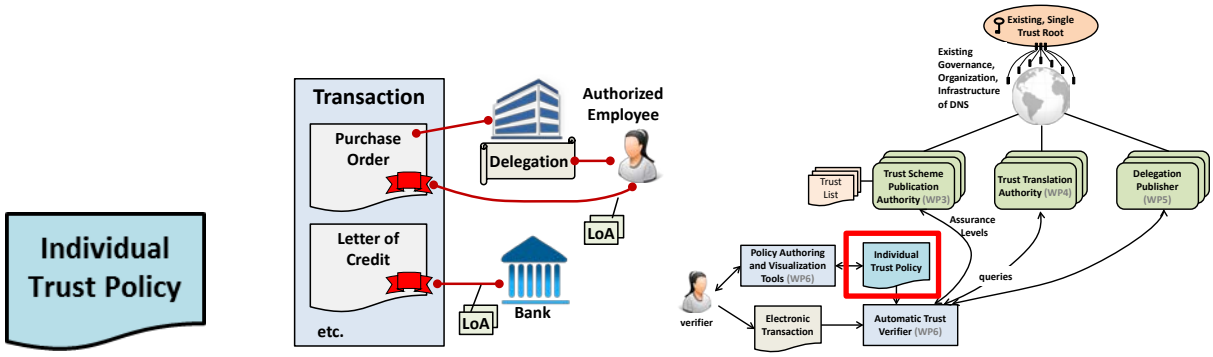


**Figure 6: Trust Policy**

Trust Policy (see Figure 6: Trust Policy) provides formal instructions how to validate trustworthiness of a given type of a transaction. It always states which Trust Lists from which Authorities should be used. The deliverables of the Work Package 6 will present more detailed information on this component.

## 11.12 Electronic Transaction

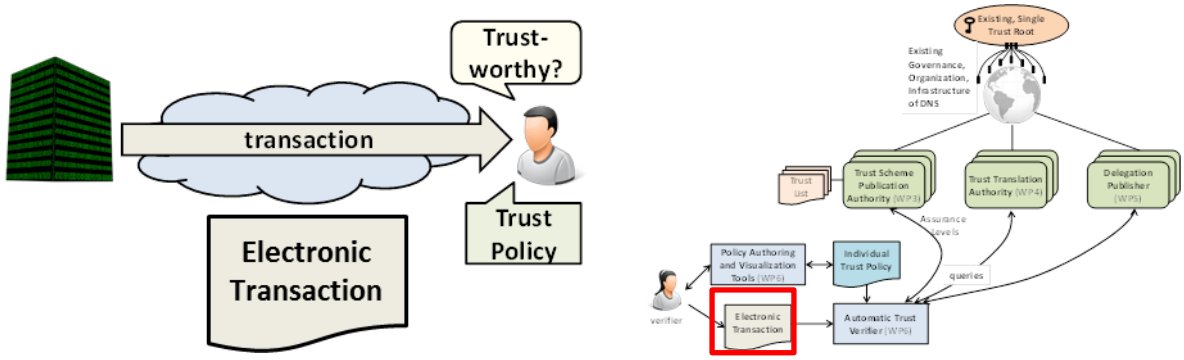


**Figure 7: Electronic Transaction**

An Electronic Transaction (see Figure 7: Electronic Transaction) is something that we receive from a remote digital agency: it might be a digitally signed document, a multi-part purchase order, etc. Work Packages 6 and 9 will present more information and a selection of concrete Electronic Transactions that can be used to demonstrate LIGHT$^{est}$.

## 11.13 Parser(s) for Electronic Transaction

A parser for Electronic Transaction is a sub-component that can be plugged into the Automatic Trust Verifier and handles a specific format of an Electronic Transaction. Further description will be presented in the Work Packages 6 and 9.

## 11.14 Automatic Trust Verifier (ATV)

Automatic Trust Verifier (see Figure 8: Automatic Trust Verifier) takes an Electronic Transaction and Trust Policy as input. The ATV provides as outputs if the Electronic Transaction is trustworthy [y/n] and optionally with explanation of its reasoning (in particular if not trustworthy). It uses a pluggable parser for Electronic Transactions as sub-component. More information will be provided in the deliverables of the Work Package 6.

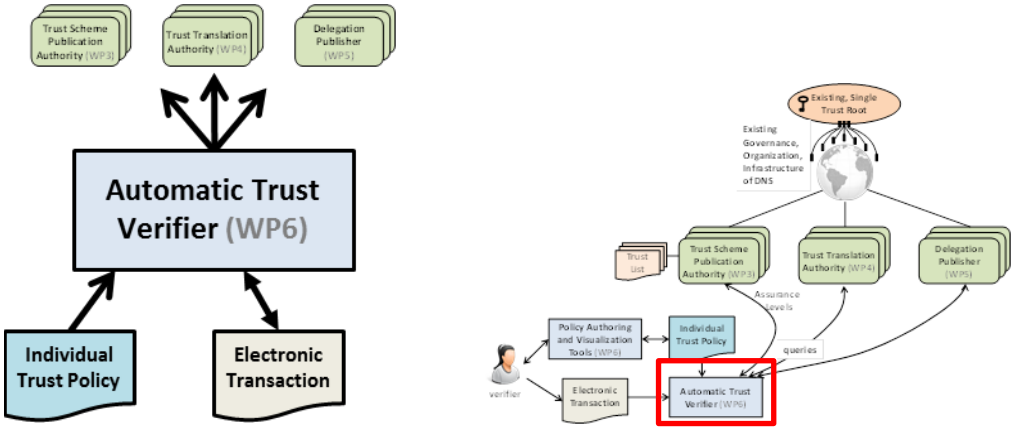


**Figure 8: Automatic Trust Verifier**

# 12. Scenarios

## 12.1 Purpose

This section presents examples of usage scenarios of LIGHT$^{est}$ that illustrate different aspects including the flows in the architecture. These concrete scenarios act as a basis to extract the abstract concepts and to understand the architecture better.

Actors in these scenarios are the following:

- Verifier: Verifier of an electronic transaction who feeds individual trust policies and electronic transactions as input to ATV
- ATV: Automatic Trust Verifier is the LIGHT$^{est}$ Component that verifies whether the electronic transaction satisfies the verifier's trust policy.
- TSPA: Trust Scheme Publication Authority manages multiple trust schemes and publishes trust scheme data to verifiers
- TTA: Trust Translation Authority provide the necessary translation data to map the levels of assurance of the foreign trust scheme to its equivalent in the domestic trust scheme in a cross-jurisdiction setting.
- DP: Delegation Publisher permit verifiers to query delegations and mandates in case data records that compose an electronic transaction are not directly signed by the legal entity responsible for it, but by a natural person that acts as an authorized representative for the former based on a delegation.

## 12.2 Trust Publication

The following scenarios illustrate Trust Scheme Publication.

### 12.2.1 Qualified Signature

This is a relatively simple scenario that illustrates trust scheme publication for qualified signatures.

**Situation:**

Assume that a verifier and signer are both located in the EC/eIDAS trust domain and the eIDAS trust domain contains the actual eIDAS trust scheme. This could for example be managed in the following domain name structure:

- *trust.ec.europa.eu*
  - *signature*
    - *TrustScheme*
      - actual eIDAS trust scheme for qualified signature

**Electronic Transaction:**

For the electronic transaction, the assumption is made that the electronic transaction is simply a signed document. Furthermore, the certificate used to sign the document contains a link to the trust list (Trust Membership Claim) for easier discovery such as "Issuer Alt Name: PX2NO4LVPA4WHCBLYXHIKRWVRE.qualified.trust.ec.eu" that points to the DNS resource records of the native trust scheme for qualified signatures. In addition, this trust scheme lists the certificate as qualified.

**Boolean Trust Scheme**

Assume that trust policy simply states that the signature of the document is trusted if the issuer of the certificate is listed in *TrustScheme.signature.trust.ec.europa.eu*.

### Information flow in the Architecture:

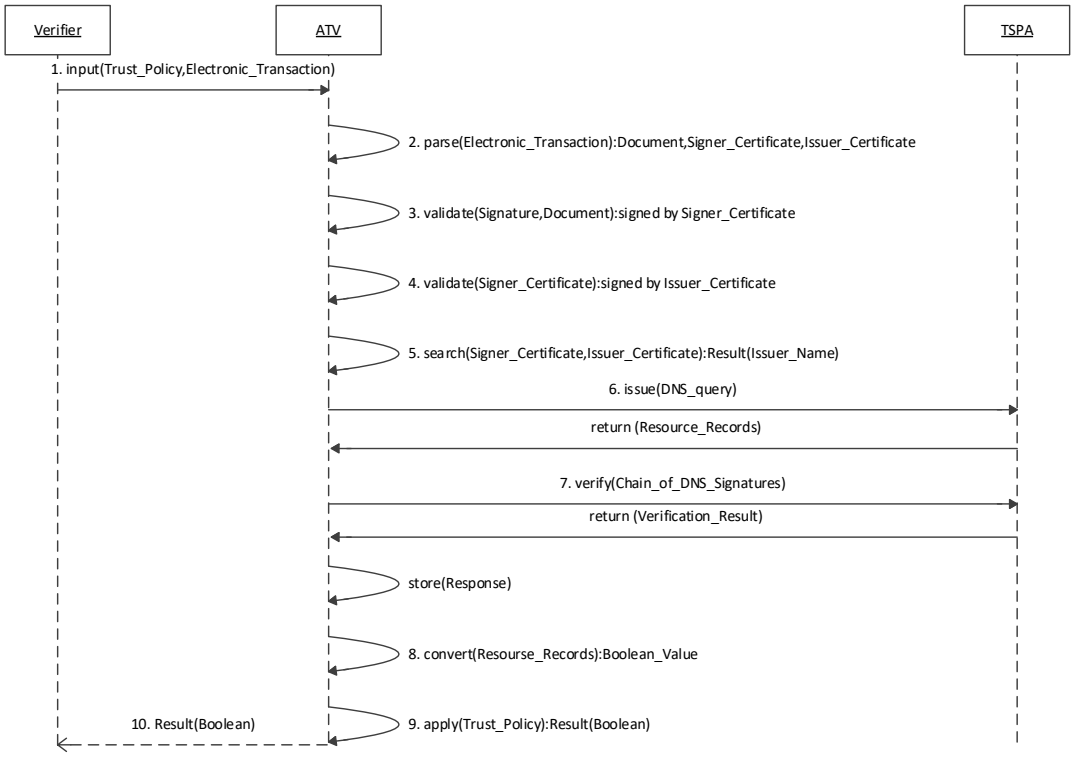


**Figure 9: Sequence Diagram for Trust Publication of a Qualified Signature (Boolean)**

1. The verifier feeds both, the Trust Policy and the Electronic Transaction into the Automatic Trust Verifier (ATV).
2. The ATV first parses the electronic transaction and yields the document, the signer certificate and the issuer certificate.
3. The ATV validates the signature on the document to make sure it is signed by the signer certificate.
4. The ATV validates that the signer certificate is signed by the issuer certificate.
5. The ATV searches the signer certificate and the issuer certificate for discovery information.
    - The ATV finds a Trust Membership Claim in the signer certificate:
        - "Issuer Alt Name: "*XYZ.qualified.trust.admin.ec*"
6. The ATV issues a DNS query for all relevant Resource Records (RRs) for Boolean Trust Schemes (as determined in Task 3.2) for *XYZ.qualified.trust.admin.eu*
7. The ATV verifies the chain of signatures from the DNS trust root of the DNS response using a validating resolver and stores the response as a "receipt" for future justification of its decision.
8. The ATV converts the RRs of the response into a boolean value. (Decoding the Encoding from the conceptual value to its DNS zone file representation).

9. The ATV looks at the trust policy and detects that the trust scheme, TrustScheme.signature.trust.ec.europa.eu.is trusted.
10. The overall result of applying the trust policy to the electronic transaction is therefore Trusted

### Ordinal Trust Scheme

Assume that trust policy simply states that the signature of the document is trusted if the issuer of the certificate is listed in a Trust Scheme according to XYZ.qualified.trust.admin.ec or higher (or equal).

### Information flow in the Architecture:

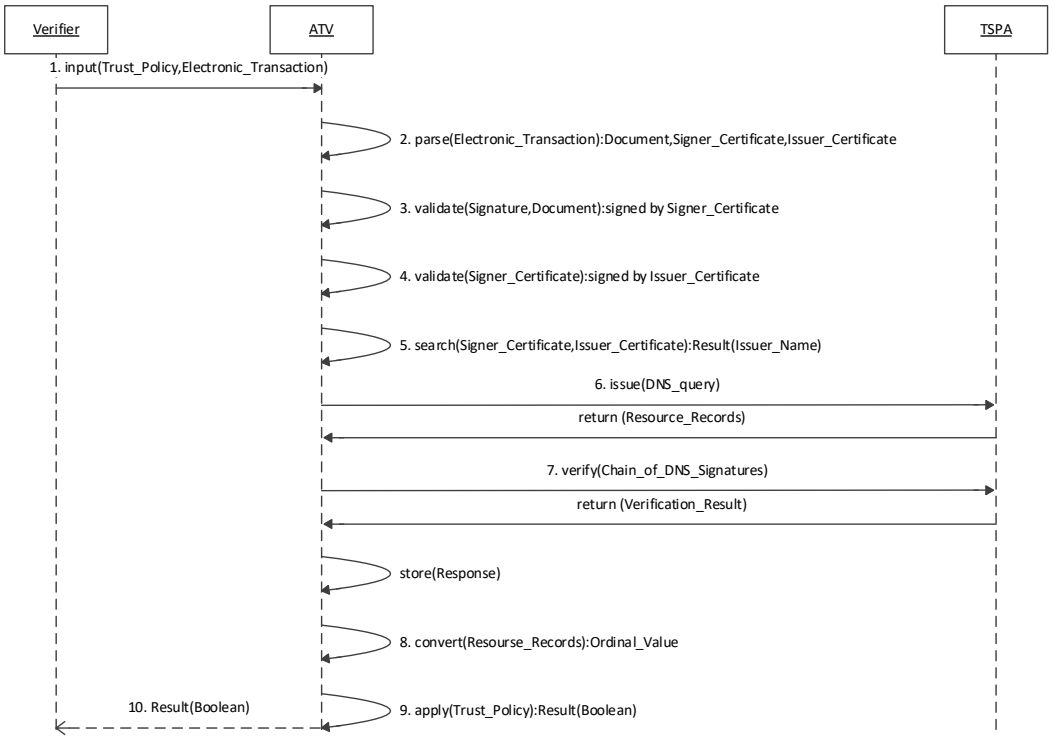


**Figure 10: Sequence Diagram for Trust Publication of a Qualified Signature (Ordinal Values)**

1. The verifier feeds both, the Trust Policy and the Electronic Transaction Identification into the Automatic Trust Verifier (ATV).
2. The ATV first parses the electronic transaction and yields the document, the signer certificate and the issuer certificate.
3. The ATV validates the signature on the document to make sure it is signed by the signer certificate
4. The ATV validates that the signer certificate is signed by the issuer certificate.
5. The ATV searches the signer certificate and the issuer certificate for discovery information.

- The ATV finds a Trust Membership Claim in the signer certificate:
  - "Issuer Alt Name: "*XYZ.qualified.trust.admin.ec*"

6. The ATV issues a DNS query for all relevant Resource Records (RRs) for Ordinal Trust Schemes for *XYZ.qualified.trust.admin.eu*
7. The ATV verifies the chain of signatures from the DNS trust root of the DNS response using a validating resolver and stores the response as a "receipt" for future justification of its decision.
8. The ATV converts the RRs of the response into an ordinal (Decoding the Encoding from the conceptual value to its DNS zone file representation).
9. The ATV looks at the trust policy and detects that the trust scheme, *XYZ.qualified.trust.admin.ec*.is trusted.
10. The overall result of applying the trust policy to the electronic transaction is therefore trusted.

The overall result of applying the trust policy to the electronic transaction is therefore Trusted.

**Tuple-Based Trust Scheme**

Assume that trust policy simply states that the signature of the document is trusted if the issuer of the certificate is listed in a Trust Scheme according to XYZ.qualified.trust.admin.ec with additional requirements on e.g. which authentication mechanism was used.

### Information flow in the Architecture:

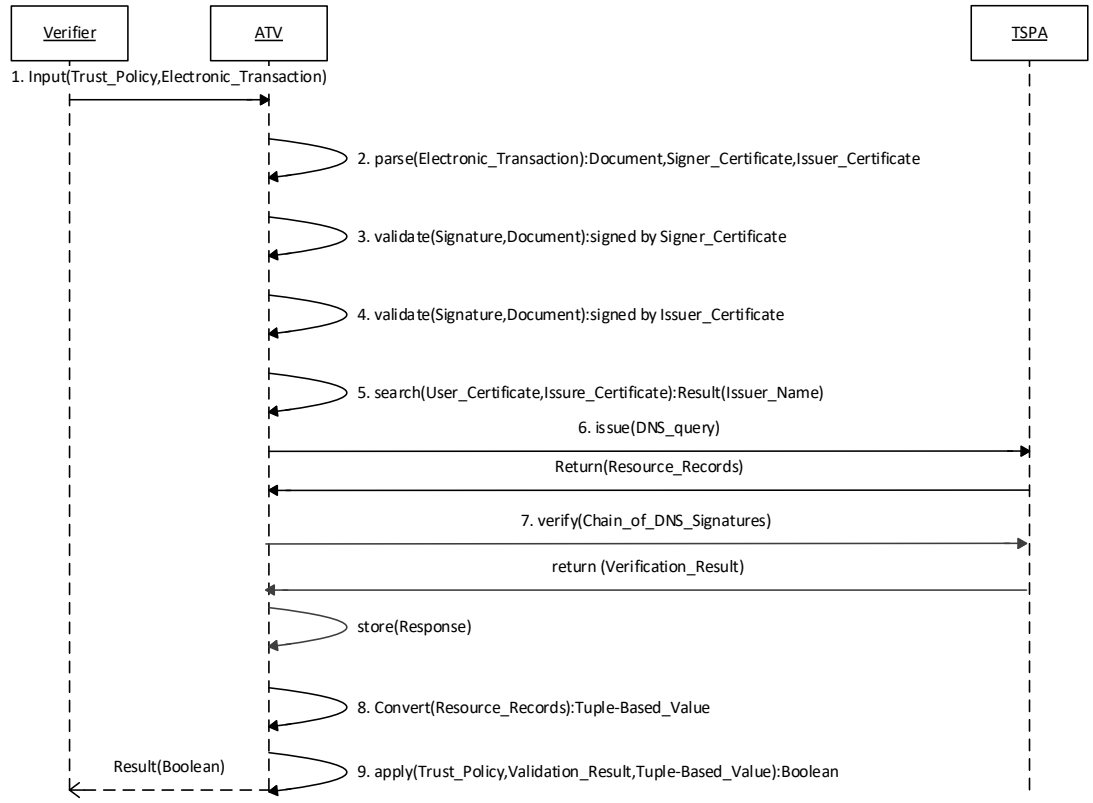


**Figure 11: Sequence Diagram for Trust Publication of a Qualified Signature (Tuples)**

1. The verifier feeds both, the Trust Policy and the Electronic Transaction Identification into the Automatic Trust Verifier (ATV).
2. The ATV first parses the electronic transaction and yields the document, the signer certificate and the issuer certificate.
3. The ATV validates the signature on the document to make sure it is signed by the signer certificate
4. The ATV validates that the signer certificate is signed by the issuer certificate.
5. The ATV searches the signer certificate and the issuer certificate for discovery information.
   - The ATV finds a Trust Membership Claim in the signer certificate:
     - "Issuer Alt Name: "*XYZ.qualified.trust.admin.ec*"
6. The ATV issues a DNS query for all relevant Resource Records (RRs) for Tuple-Based Trust Schemes for *XYZ.qualified.trust.admin.ec*
7. The ATV verifies the chain of signatures from the DNS trust root of the DNS response using a validating resolver and stores the response as a "receipt" for future justification of its decision.

8. The ATV converts the RRs of the response into multiple tuple-based values (2 factor authentication: True; Identity Proofing: InPerson) (Decoding the Encoding from the conceptual value to its DNS zone file representation).
9. The ATV looks at the trust policy and detects that the trust scheme, *XYZ. qualified.trust.admin.ec* is trusted.
10. The overall result of applying the trust policy to the electronic transaction is therefore Trusted.

### 12.2.2 Qualified Seals

The scenario for this trust service is very similar to the Qualified Signature, since the main difference between electronic signature and electronic seal is based on the nature of the creator (a natural person or a legal person/public sector body).

However, we would have to be into account there are other classifications in the industry based on the provider of the certificate, whether it is qualified or not, and on the provider of the seal, whether it is qualified or not.

The information flow in the architecture is identical to the one of the qualified signature described above.

### 12.2.3 Qualified Identity

The following scenario describes a trust scheme publication for qualified identities.

**Situation:**

Assume that a verifier and signer are both located in the EC/eIDAS trust domain and the eIDAS trust domain contains the actual eIDAS trust scheme. A Boolean Trust Scheme is used. Other types (ordinal, tuple) could be realized analogically to 12.2. This could for example be managed in the following domain name structure:

- *trust.ec.europa.eu*
  - *identity*
    - *TrustScheme*
      - actual eIDAS trust scheme for qualified identity

**Electronic Transaction:**

For the electronic transaction, the assumption is made that the electronic transaction is simply a signed document. Furthermore, the certificate used to sign the document contains a link to the trust list (Trust Membership Claim) for easier discovery such as "Issuer Alt Name: PX2NO4LVPA4WHCBLYXHIKRWVRE.qualified.trust.ec.eu" that points to the DNS resource records of the native trust scheme for qualified identities. In addition, this trust scheme lists the certificate as qualified.

### Trust Policy

Assume that trust policy simply states that the identity of the document is trusted if the issuer of the certificate is listed in *TrustScheme.identity.trust.ec.europa.eu*.

### Information flow in the Architecture:

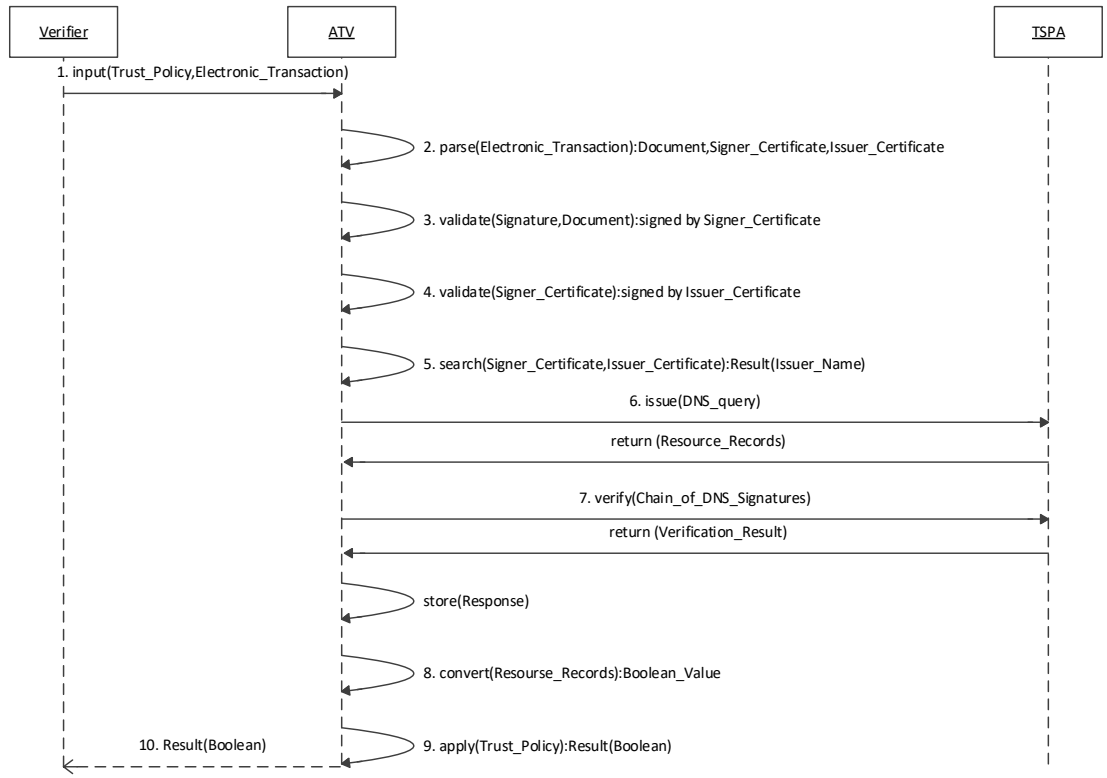


**Figure 12: Sequence Diagram for Trust Publication of a Qualified Identity (Boolean)**

1. The verifier feeds both, the Trust Policy and the Electronic Transaction into the Automatic Trust Verifier (ATV).
2. The ATV first parses the electronic transaction and yields the document, the signer certificate and the issuer certificate.
3. The ATV validates the signature on the document to make sure it is signed by the signer certificate.
4. The ATV validates that the signer certificate is signed by the issuer certificate.
5. The ATV searches the signer certificate and the issuer certificate for discovery information.
    - The ATV finds a Trust Membership Claim in the signer certificate:
        - "Issuer Alt Name: "*XYZ.qualified.trust.admin.ec*"

6. The ATV issues a DNS query for all relevant Resource Records (RRs) for Boolean Trust Schemes for *XYZ.qualified.trust.admin.eu*
7. The ATV verifies the chain of signatures from the DNS trust root of the DNS response using a validating resolver and stores the response as a "receipt" for future justification of its decision.
8. The ATV converts the RRs of the response into a boolean value. (Decoding the Encoding from the conceptual value to its DNS zone file representation).
9. The ATV looks at the trust policy and detects that the trust scheme, TrustScheme.identity.trust.ec.europa.eu.is trusted.

The overall result of applying the trust policy to the electronic transaction is therefore Trusted.

#### 12.2.4 Qualified Timestamp

This scenario shows trust scheme publication for qualified timestamps.

**Situation:**

Assume that a verifier and signer are both located in the EC/eIDAS trust domain and the eIDAS trust domain contains the actual eIDAS trust scheme. A Boolean Trust Scheme is used. Other types (ordinal, tuple) could be realized analogically to 12.2. This could for example be managed in the following domain name structure:

- *trust.ec.europa.eu*
  - *timestamp*
    - *TrustScheme*
      - actual eIDAS trust scheme for qualified timestamp

**Electronic Transaction:**

For the electronic transaction, the assumption is made that the electronic transaction is simply a signed document. Furthermore, the certificate used to sign the document contains a link to the trust list (Trust Membership Claim) for easier discovery such as "Issuer Alt Name: PX2NO4LVPA4WHCBLYXHIKRWVRE.qualified.trust.ec.eu" that points to the DNS resource records of the native trust scheme for qualified timestamps. In addition, this trust scheme lists the certificate as qualified.

**Trust Policy**

Assume that the trust policy simply states that the timestamp is trusted if the source of the time data is listed in *TrustScheme.timestamp.trust.ec.europa.eu*.

### Information flow in the Architecture:

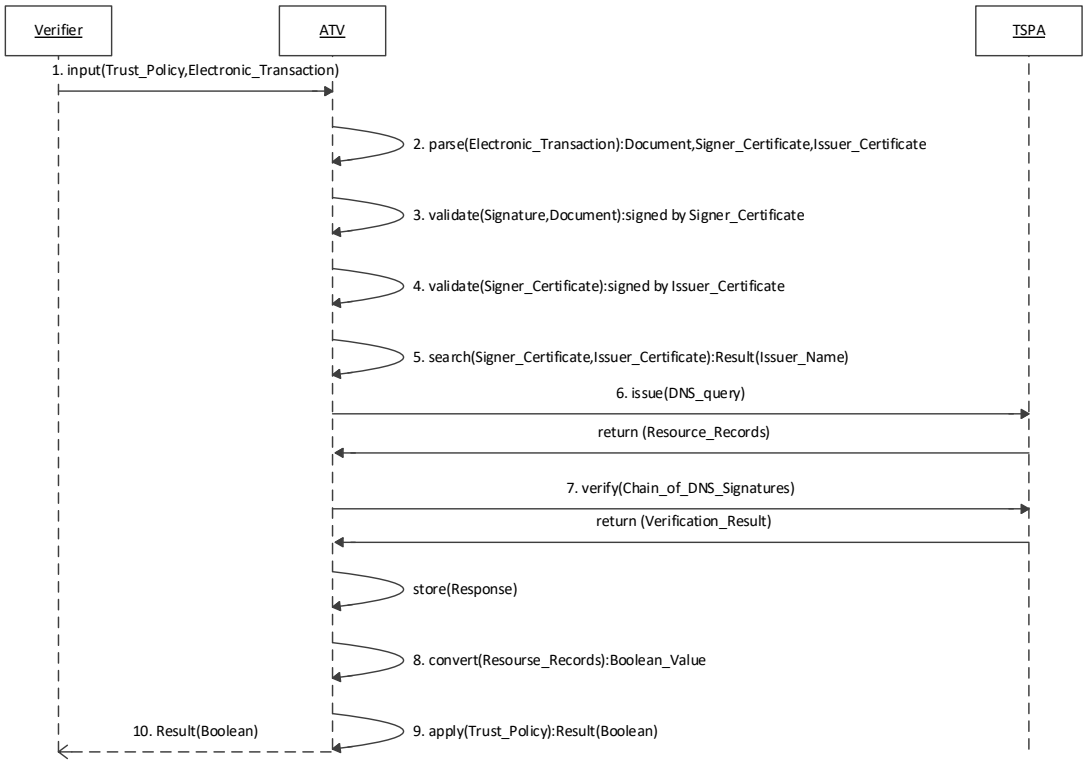


**Figure 13: Sequence Diagram for Trust Publication of a Qualified Timestamps (Boolean)**

1. The verifier feeds both, the Trust Policy and the Electronic Transaction into the Automatic Trust Verifier (ATV).
2. The ATV first parses the electronic transaction and yields the document, the signer certificate and the issuer certificate.
3. The ATV validates the signature on the document to make sure it is signed by the signer certificate.
4. The ATV validates that the signer certificate is signed by the issuer certificate.
5. The ATV searches the signer certificate and the issuer certificate for discovery information.
    - The ATV finds a Trust Membership Claim in the signer certificate:
        - "Issuer Alt Name: "*XYZ.qualified.trust.admin.ec*"
6. The ATV issues a DNS query for all relevant Resource Records (RRs) for Boolean Trust Schemes for *XYZ.qualified.trust.admin.eu*
7. The ATV verifies the chain of signatures from the DNS trust root of the DNS response using a validating resolver and stores the response as a "receipt" for future justification of its decision.

8. The ATV converts the RRs of the response into a boolean value. (Decoding the Encoding from the conceptual value to its DNS zone file representation).
9. The ATV looks at the trust policy and detects that the trust scheme, TrustScheme.timestamp.trust.ec.europa.eu.is trusted.

The overall result of applying the trust policy to the electronic transaction is therefore Trusted.

## 12.3 Trust Translation

The following scenarios illustrate Trust Scheme Translation. In contrast to Trust Publication as described in 12.2., the main difference of this scenario is that the trust services do not provide original trust information but translate information from existing sources into equivalent or similar representations (depending on the context) but expressed using other schemes.

### 12.3.1 Qualified Signature

This relatively simple scenario illustrates trust scheme translation for qualified signatures.

**Situation:**

Assume that a verifier is located in the EC/eIDAS trust domain. Furthermore, the EC has negotiated bilateral agreements with important countries on the mutual recognition of qualified electronic signatures (e.g. Switzerland, United States, Canada). The eIDAS trust domain contains both, the actual eIDAS trust scheme and several trust translation schemes, one for each country. This could for example be managed in the following domain name structure:

- *trust.ec.europa.eu*
  - *signature*
    - *TrustScheme*
      - actual eIDAS trust scheme for qualified signature
    - *TranslationSchemes*
      - CH-translationScheme: qualified__trust__admin__ch
        - under this "host label", a single TXT RR with the recipe is available (plus the RRs for DNSSEC)
      - US-translationScheme
      - CA-translationScheme

For simplification, all above countries have a single trust scheme for qualified signatures (possibly using subsidiarity as done in Europe).

**Electronic Transaction:**

For the electronic transaction, the assumption is made that the electronic transaction is simply a signed document. Furthermore, the certificate used to sign the document contains a link to the trust list (Trust Membership Claim) for easier discovery such as "Issuer Alt Name: PX2NO4LVPA4WHCBLYXHIKRWVRE.qualified.trust.admin.ch" that points to the DNS resource records of the native trust scheme for qualified signatures. In addition, this trust scheme lists the certificate as qualified.

### Trust Policy

Assume that trust policy simply states that the signature of the document is trusted if the issuer of the certificate is listed in TrustScheme.signature.trust.ec.europa.eu or any of the Translation Schemes under TranslationSchemes.signature.trust.ec.europa.eu.

### Trust Translation Schemes

Assume that the trust translation scheme for each country contains two entries:

- the foreign trust scheme that is referenced by its domain name:
  - qualified.trust.admin.ch
    - if this is already encoded in the domain name component of the translation scheme (see below), the translation scheme may contain only the recipe for the conversion below).
- the recipe for the conversion trust scheme values from the foreign domain into the native domain:
  - equivalent
    - note: equivalent is the most common recipe for translation of boolean schemes.

## Information flow in the Architecture:

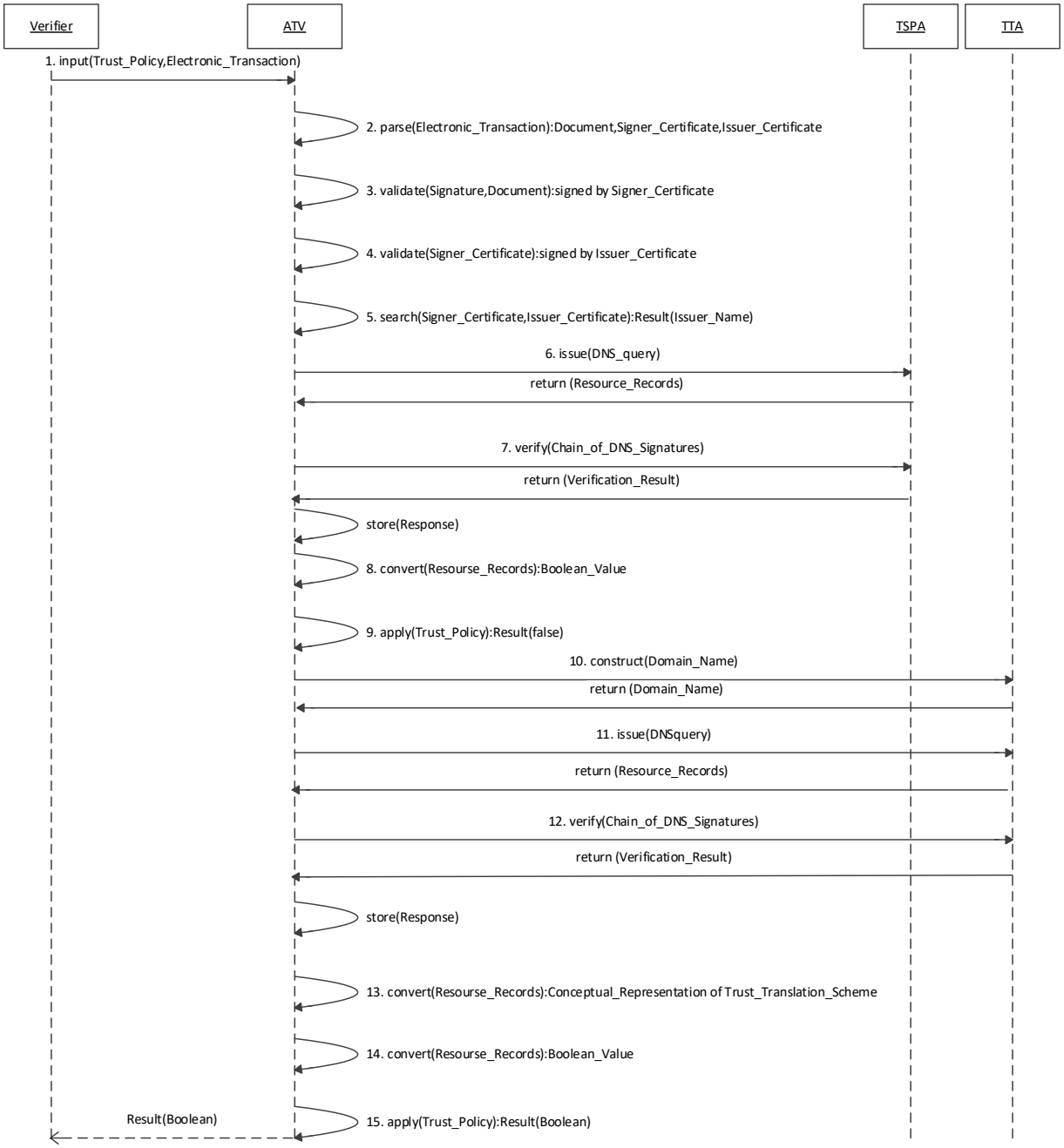


**Figure 14: Sequence Diagram for Trust Translation of a qualified signature (boolean)**

1. The verifier feeds both, the Trust Policy and the Electronic Transaction into the Automatic Trust Verifier (ATV).
2. The ATV first parses the electronic transaction and yields the document, the signer certificate and the issuer certificate.
3. The ATV validates the signature on the document to make sure it is signed by the signer certificate.
4. The ATV validates that the signer certificate is signed by the issuer certificate.
5. The ATV searches the signer certificate and the issuer certificate for discovery information.
    - The ATV finds a Trust Membership Claim in the signer certificate:
        - "Issuer Alt Name: *PX2NO4LVPA4WHCBLYXHIKRWVRE.qualified.trust.admin.ch*"
6. The ATV issues a DNS query for all relevant Resource Records (RRs) for Boolean Trust Schemes (as determined in Task 3.2) for *PX2NO4LVPA4WHCBLYXHIKRWVRE.qualified.trust.admin.ch*
7. The ATV verifies the chain of signatures from the DNS trust root of the DNS response using a validating resolver and stores the response as a "receipt" for future justification of its decision.
8. The ATV converts the RRs of the response into a boolean value.
9. The ATV looks at the trust policy and detects that the trust scheme qualified.trust.admin.ch is not TrustScheme.signature.trust.ec.europa.eu.
10. The ATV therefore constructs a domain name in order to look for a matching trust translation scheme in TranslationSchemes.signature.trust.ec.europa.eu.
    - This domain name is *qualified__trust__admin__ch.TranslationSchemes.signature.trust.ec.europa.eu*
        - Note that the Fully Qualified Domain Name of the Swiss trust scheme is converted into a single domain name component by converting '.' into '__'.
11. The ATV issues a DNS query for the relevant RRs (determined in Task 4.2) for *qualified__trust__admin__ch.TranslationSchemes.signature.trust.ec.europa.eu*
12. The ATV verifies the chain of signatures from the DNS trust root of the DNS response using a validating resolver and stores the response as a "receipt" for future justification of its decision.
13. The ATV converts the received RRs into a conceptual representation of the trust translation scheme
14. The ATV applies the translation recipe of the trust translation scheme to the value found in step 8
    - since the recipe is "equivalence", the Boolean True is converted to True
15. The overall result of applying the trust policy to the electronic transaction is therefore Trusted

In this scenario of Trust Translation of a qualified signature a Boolean Trust Scheme is used.

Other types (ordinal, tuple: see chapter 12.2) could be realized analogically including the points 9 to 14 of the above Information flow in the Architecture accordingly.

### 12.3.2 Qualified Seals, Identities and Timestamps

The scenario for the qualified seal is very similar to the Qualified Signature, since the main difference between electronic signature and electronic seal is based on the nature of the creator (a natural person or a legal person/public sector body). So the translation result can be equivalent or not.

However, we would take into account there are other classifications in the industry based on the provider of the certificate, whether it is qualified or not, and on the provider of the seal, whether it is qualified or not.

The information flow in the architecture is identical to the one of the qualified signature described above. Likewise, the scenarios for Qualified Identities and Timestamps are very similar to the signature scenario. Also, only the data / data structures (=provided attributes) differ. The formal representation of this data will be defined in WP4. The information flow is identical to the information flow of the qualified signature described above.

## 12.4 Trust Delegation

The following scenario illustrates Trust Delegation Scheme.

**Situation:**

Assume that a verifier and signer are both located in the EC/eIDAS trust domain and the eIDAS trust domain contains the actual eIDAS trust scheme. This could for example be managed in the following domain name structure:

- *trust.ec.europa.eu*
  - *signature*
    - *TrustScheme*
      - actual eIDAS trust scheme for qualified signature

Furthermore it is assumed the electronic transaction is not directly signed by the legal entity responsible for it (e.g., using a company seal), but by a natural person that acts as an authorized representative for the former based on a delegation.

**Electronic Transaction:**

For the electronic transaction, the assumption is made that the electronic transaction is a purchase order. Furthermore, the certificate used to sign the document contains a link to the trust list (Trust Membership Claim) for easier discovery such as "Issuer Alt Name: PX2NO4LVPA4WHCBLYXHIKRWVRE.qualified.trust.ec.eu" that points to the DNS resource

records of the native trust scheme for qualified purchases. In addition, this trust scheme lists the certificate as qualified.

In a second step, the delegation from the purchase order is verified. The certificate used to sign the document contains a link to the Purchasing List of Company XYZ for easier discovery such as "Issuer Alt Name: PX2NO4LVPA4WHCBLYXHIKRWVRE.purchasing.CompanyXYZ.com" that points to the DNS resource records of the native trust scheme for qualified signatures.

### Trust Policy

Assume that trust policy simply states that the signature of the document is trusted if (i) the issuer of the certificate is listed in TrustScheme.signature.trust.ec.europa.eu and (ii) the delegation of CompanyXYZ is valid.

## Information flow in the Architecture:

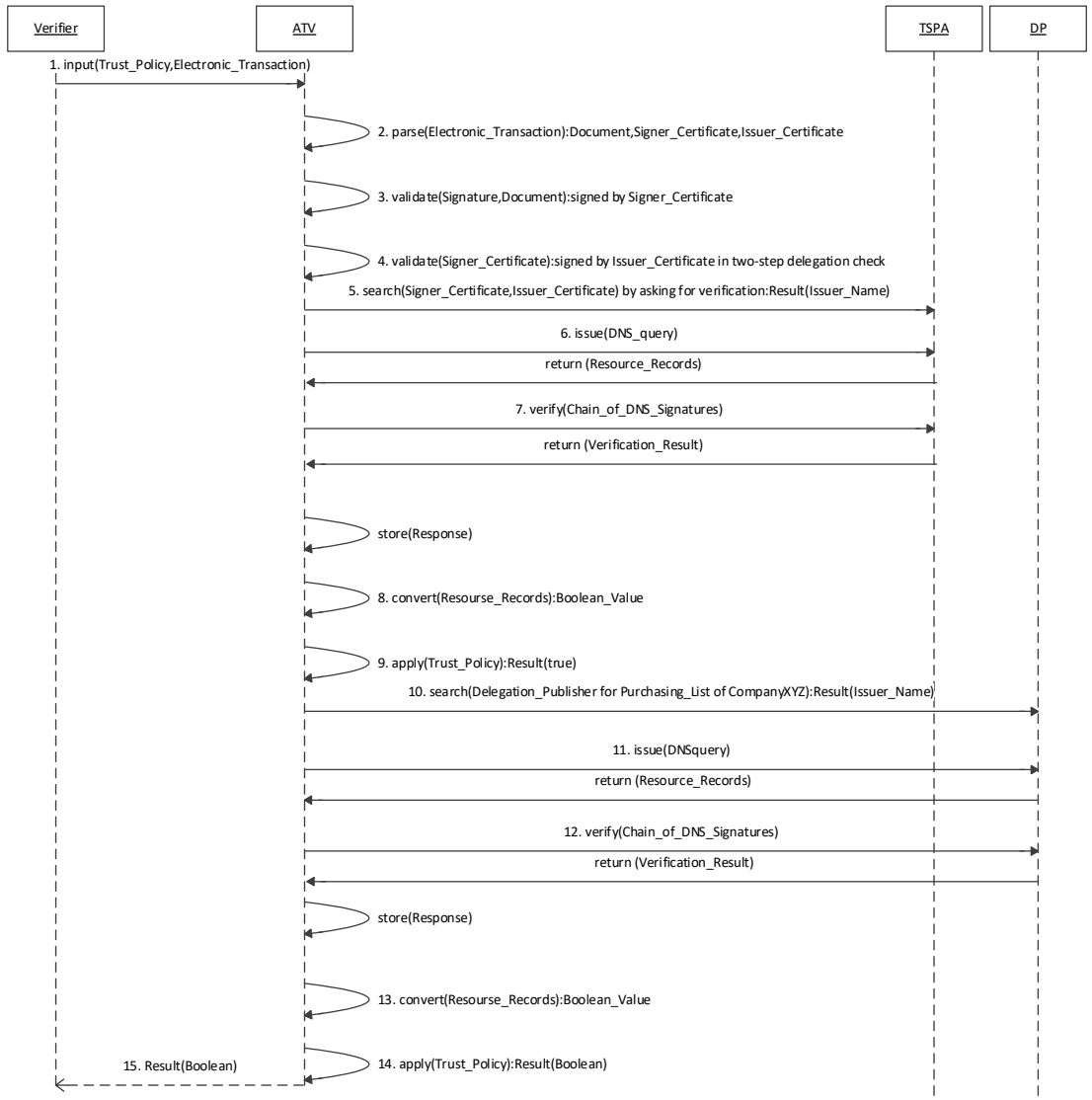


**Figure 15: Sequence Diagram for Trust Delegation Scenario**

1. The verifier feeds both, the Trust Policy and the Electronic Transaction into the Automatic Trust Verifier (ATV).
2. The ATV first parses the electronic transaction and yields the document, the signer certificate and the issuer certificate.
3. The ATV validates the signature on the document to make sure it is signed by the signer certificate.

4. The ATV validates that the signer certificate is signed by the issuer certificate in a two-step delegation check.

5. The ATV searches the signer certificate and the issuer certificate for discovery information. The ATV searches for this by asking for verification from the Trust Scheme Publication Authority.
    - The ATV finds a Trust Membership Claim from the Trust Scheme Publication Authority in the signer certificate:
        i. "Issuer Alt Name: AFJFDAKdjFKAGJGfkdS.qualified.trust.ec

6. The ATV issues a DNS query for all relevant Resource Records (RRs) for Boolean Trust Schemes (as determined in Task 3.2) at the Trust Scheme Publication Authority for "*AFJFDAKdjFKAGJGfkdS.qualified.trust.ec* "
7. The ATV verifies the chain of signatures from the DNS trust root of the DNS response using a validating resolver and stores the response as a "receipt" for future justification of its decision.
8. The ATV converts the RRs of the response into a boolean value. (Decoding the Encoding from the conceptual value to its DNS zone file representation).
9. The ATV looks at the trust policy and detects that the trust scheme AFJFDAKdjFKAGJGfkdS.qualified.trust.ec is trusted
10. The ATV tries to verify the delegation from the purchase order. It looks for discovery information about the Delegation Publisher for the Purchasing list of CompanyXYZ.
    - The ATV finds a Delegation Claim in the Purchasing list of CompanyXYZ:
        - "Issuer Alt Name: *AFJFDAKdjFKAGJGfkdS*.purchasing.CompanyXYZ.com
11. The ATV issues a DNS query for all relevant Resource Records (RRs) for Delegation (as determined in Task 5.2) for "*AFJFDAKdjFKAGJGfkdS.purchasing.CompanyXYZ.com* "
12. The ATV verifies the chain of signatures from the DNS trust root of the DNS response using a validating resolver and stores the response as a "receipt" for future justification of its decision.
13. The ATV converts the RRs of the response into a boolean value. (Decoding the Encoding from the conceptual value to its DNS zone file representation).
14. The ATV looks at the trust policy and detects that the delegation AFJFDAKdjFKAGJGfkdS.purchasing.CompanyXYZ.com is valid delegation of purchasing.CompanyXYZ.qualified.com
15. The overall result of applying the trust policy to the electronic transaction is therefore Trusted

## 12.5 Examples of Advanced Scenarios

The basic scenarios described in 12.1 – 12.3. illustrate basic applications of the trust infrastructure. This basic functionality (publish, translate, delegate) can be used to realise a wide range of more sophisticated scenarios. This section provides several examples of such scenarios. These scenarios can be either variants of the basic scenarios or a combination of different basic scenarios. Composing two trust services is a chaining process in which the output level of the inner trust service becomes the input level of the outer trust service. By the output level we mean the result of the translation in the schema target.

There is an example of such composition embedded in the Qualified Signature scenario, when the electronic document has to be parsed to obtain the signer certificate and the issuer certificate. In a situation like the following one, where several trust services are taking part, and notice how each trust service has to be translated in an ordered way.

**Qualified Delivery Services:** E-registered delivery can be realised using a combination of the scenarios signature and timestamps from section 12.2. This scenario will be demonstrated in detail in the Trustworthy Communication Services Pilot. This pilot integrates LIGHT$^{est}$ into the e-Correos platform. The pilot will assess and validate the following features of LIGHT$^{est}$ in an operational context:

- Use of an internal LIGHT$^{est}$ Trust Scheme Publication Authority (TSPA) for trust lists issued by Correos partners (e.g., from the UPU-Universal Post Union) and for managing external Identity Providers that issue access credentials.
- Use of external TSPAs for accessing the EU TSL and possibly trust lists for non-qualified trust services. Multiple levels of assurance need to be managed.
- Use of LIGHT$^{est}$ client libraries to access the above TSPAs and determine the level of assurance of the various involved certificates.
- Experimential use of LIGHT$^{est}$ to extend e-Correos use beyond Europe and use certificates from Turkey and the United States.

The pilot will also use delegation functionality in relation to the integration of mobile IDs.

**Qualified Website Authentication:** Another example is to use the LIGHT$^{est}$ infrastruture for authentication, such as user authentication for websites or e.g. authentication functionality for assets in the Internet of Things. Basic scenario for this is Trust Publication (see. 12.2.) with qualified identities. Additionally, Trust Translation could be used in order to authenticate third party users/things. It is imaginable that in some cases trust delegation also may make sense, e.g. if functionality of the authenticating system is allowed/intended to be delegated to other persons.

**Authentication with LoAs:** This example scenario is basically equivalent to the authentication example above but the involved trust schemes are of type ordinal and a trust translation scheme therefore needs a more complex recipe for the conversion of trust scheme values from the foreign domain to the native one. One possible approach here would be a lookup table.

## 12.6 Trust Evaluation Process

We illustrate the trust evaluation process and our trust policy language with a few examples. The language is based on Horn clauses that have the form

$$conclusion :- requirements$$

This loosely corresponds to a sentence of the form: “if the requirements on the right are all satisfied, I get the left-hand side conclusion as a result”. Consider as a specific example the sentence “I trust X if I trust someone that delegates to X”. This could be expressed as a Horn clause in the following way.

$$trust(X) :- trust(Y), delegate(Y, X).$$

In the above, $:-$ should be read as “if” in the sense of a sufficient (but not necessary) condition, i.e., if the requirements on the right-hand side are not met, there may still be another clause to derive that I trust X. The comma between the requirements should be read as "and". The clause should thus be read as “I trust X if I trust Y, and Y delegates to X”. Here the terms $trust$ and $delegate$ are not built-in parts of the (base) language, and the clause says nothing of their meaning in isolation, i.e. nothing is said of what it means to trust or delegate to something. However, we may consider having a library of the most important terms and concepts (so users do not have to start from scratch when specifying their own policy) and they may have also a distinguished meaning for our ATV.

The language would then consist of a set of such clauses, each having exactly one term (the *head* of the clause) before the $:-$ (the “if”), and zero or more terms separated by commas (the *body* of the clause) after the $:-$. Expressing trust by a bounded number of delegations in this language could be done using the following two clauses.

$$trust(X, N) :- trust(X).$$

$$trust(X, N) :- N > 0, delegate(Y, X), trust(Y, N - 1).$$

The first clause should express that I trust X through at most N steps of delegation if I trust X directly (without considering delegation). The second clause says that I trust X through at most N steps of delegation if N is greater than zero, Y delegates to X, and I trust Y through N-1 steps of delegation.

One big advantage of using this language is that it happens to be valid Prolog code. It is then possible to use a Prolog environment to evaluate the policy against a prototype, which, conveniently, can also be specified as Prolog code. The following is a prototype meant to express that I trust *a*, *b*, and *c* (this could represent that I trust these because they are listed in some trust list), and that *c* delegates to *d*, which delegates to *e*.

$$trust(a).$$

$$trust(b).$$

$$trust(c).$$

$$delegate(c, d).$$

$$delegate(d, e).$$

With the two clauses with $trust(X, N)$ in the head loaded into a Prolog environment together with the above prototype, the query $trust(e, 2)$. will return true, and $trust(e, 1)$. will return false.

## 12.7 Formalization of some Scenarios in TPL

The Trust Policy Language that we propose is capable of expressing both the policy itself and the operational aspects (like queries to a DNS server). To supplement the explanations of the previous subsections, we give a possible formalization of three scenarios. We begin with the most basic scenario:

### Boolean Trust Scheme Without Translation

This is the information flow and scenario first described in Section 12.2.1. We formalize it as a predicate of a text `Text` being supposedly signed by a `Signer`, these two parameters would normally be input to a query; as comments we have the step numbers from the information flow diagram above:

```
trustPublicationScenario(Text,Signer) :-
  % Steps 1.-5. checking document and signatures (local):
  document(signature(Text,PkSig)),
  certificate(SigCert),
    issuer(SigCert,Issuer),
    bearer(SigCert,Signer),
    pubkey(SigCert,PkSig),
    issuerkey(SigCert,PkIss),
    altIssuerName(SigCert,TrustMemClaim),
  % Step 6.-8. Query DNS
  queryBTS(TrustMemClaim,Issuer,PkIss),
  % Step 9. Check Trust policy
  scheme(TrustMemClaim,TrustScheme),
  trust(TrustScheme).
```

Here, the predicates `document` and `certificate` refer to facts that should be part of the current "knowledge base", i.e., that are fed into the automated trust verifier initially. The predicates `issuer` and the like are for extracting aspects of the certificate (e.g., who is the issuer of the certificate). This is done with such predicates in order to formulate it in a general way that is independent of a concrete credential format. To support a new credential format, one thus has to just define these predicates for it. Thus, these parts of the clause represent that we

check the existing credentials and extract the Trust Membership Claim from it, which is the URL we then use in the DNS query -- that is then expressed with the predicate `queryBTS(TrustMemClaim,Issuer,PkIss).` Note that conceptually, TPL (or Prolog) do not have a notion of input or output in these predicates. This is because either can be left a variable in a query and we then evaluate to find the possible solutions for these variables to make the query true. The query predicate would support in fact three different options for this:

1. Only the URL (`TrustMemClaim`) is the input, and the result of the query is the entire trust list with their certificates.
2. The URL and the Issuer name, and output is the public key if the issuer is in the trust list.
3. All three -- URL, Issuer name and public key -- are input, and the query only can succeed (if the issuer is part of the trust list and has the given public key) or fail.

Note that this abstracts from the precise mechanisms with RR-records and verifying the certificate chain to the top-level domain. Finally, we have to check that we even trust the given trust scheme. Usually the trust scheme name may not be identical with the URL of the `TrustMemClaim,` therefore we use an additional predicate `scheme` to extract the trust scheme name. Then we finally verify whether the TrustScheme actually satisfies our trust policy which can be in the example scenario simply an enumeration of trusted schemes e.g.:

```
trust(["TrustScheme","signature","trust","ec","europa","eu"]).
```

### Ordinal Trust Scheme Without Translation

This is the second information flow and scenario described in Section 12.2.1. Here in addition, we have to check an attribute, in the example that the method of identity proofing of the said certificate was "in person". To that end, we only modify the query predicate to return a set of Attributes as a result:

```
ordinalTrustPolicyScenario(Text,Signer) :-
  % Steps 1.-5. checking document and signatures (local):
  document(signature(Text,PkSig)),
  certificate(SigCert),
    issuer(SigCert,Issuer),
    bearer(SigCert,Signer),
    pubkey(SigCert,PkSig),
    issuerkey(SigCert,PkIss),
    altIssuerName(SigCert,TrustMemClaim),

  % Step 6.-8. Query DNS
  queryOTS(TrustMemClaim,Issuer,PkIss,Attrib),
  attribute(Attrib,identityProofing,inPerson),

  % Step 9.-10. Check Trust policy
  scheme(TrustMemClaim,TrustScheme),
  trust(TrustScheme).
```

Here the predicate `attribute` is used to extract attributes from the (potentially complex) set of attributes that the query returns. This is again done to make this specification independent of the concrete representation of the attributes (like a list of pairs of attribute name and attribute value), basically only requiring that there is an attribute called "identityProofing" and that its value is "inPerson". Similar things we can do with all kinds of attributes, including level of assurance. One may argue that this is actually mixing of a procedure (checking a document and obtaining certificates) with the actual trust policy (who is trusted, which attributes are required). Our trust policy language of course allows us in a nice way to separate these concerns, as we describe in more detail in Deliverable 2.4 on the Formal Concepts (that also contains a more detailed introduction to the Trust Policy Language). Here the purpose was to faithfully represent the information flows we have described in the previous subsections.

### Trust Translation Scheme Scenario

As a third example we look at the Boolean Trust Translation Scheme Scenario from Section 12.3. This is like the first scenario, but where the trust scheme is actually foreign (e.g. a Swiss trust scheme) and needs to be translated (e.g. into European qualified signature). The beginning is similar again to the Boolean trust scheme, and all the difference lies in the predicate `trustWithTranslation` that allows to accept the trust in this scheme if it is either directly trusted or can be translated into a trusted one:

```
trustTranslationScenario(Text,Signer) :-
  % Steps 1.-5. checking document and signatures (local):
  document(signature(Text,PkSig)),
  certificate(SigCert),
    issuer(SigCert,Issuer),
    bearer(SigCert,Signer),
    pubkey(SigCert,PkSig),
    issuerkey(SigCert,PkIss),
    altIssuerName(SigCert,TrustMemClaim),

  % Step 6.-8. Query DNS
  queryBTS(TrustMemClaim,Issuer,PkIss),

  % Step 9.-15. Check Trust policy
  scheme(TrustMemClaim,TrustScheme),
  trustWithTranslation(TrustScheme).
```

Here, the `trustWithTranslation` is defined by two clauses, namely if it is directly a trusted scheme, or otherwise we check if it can be translated into a scheme that we already trust in:

```
trustWithTranslation(TrustScheme) :- trust(TrustScheme).
trustWithTranslation(ForeignTrustScheme) :-
  trust(TrustScheme),
  translation(ForeignTrustScheme,TrustScheme).
```

Here `translation` is a predicate that invokes again a query similar to a query for native schemes as we had them before:

```
translation(ForeignTrustScheme,TrustScheme) :-
  encoding(ForeignTrustScheme,TrustScheme,URL),
  queryBTSTranslationScheme(URL).
```

Here, we first need to generate a URL out of the foreign trust scheme and the one we want to translate to; this is done by the predicate `encoding` for our example, the foreign trust scheme is `["admin","ch"]`, the (native) trust scheme is `["signature","trust","eu"]` and encoding would yield the URL `["XYZ__admin__ch","Translation","signature","trust","eu"]`.

# 13. Conclusions

This deliverable provided a High-level description of the LIGHT$^{est}$ Reference Architecture. It describes architectural principles and both functional and technical goals addressed by the architecture.

It defines the essential terms of the project, provides a description of the different components and highlights how these components interact with each other. The information flows for the different scenarios are shown using textual descriptions as well as UML sequence diagrams. The specific details for the components will be described in the respective deliverables of the development work packages.

# 14. References

(S. De Capitani, 2012) S. De Capitani, S. Foresti, S. Jajodia, S. Paraboschi, G. Psaila, P. Samarati: Integrating Trust Management and Access Control in Data-Intensive Web Applications. In: ACM Transactions on the Web, 1-43, 2012.

(Edsger et al., 1976) Edsger, Wybe, Dijkstra: A discipline of programming. Prentice Hall, 1976, ISBN 978-0-13-215871-8, p. 56.

(Gurevich, 2008) Y. Gurevich, I. Neeman: DKAL: Distributed-Knowledge Authorization Language. In: Microsoft Research, 2008.

(Lee, Anderson., 2012) Lee, Peter A.; Anderson, Thomas. Fault tolerance: principles and practice. Springer Science & Business Media, 2012

(Parnas, 1972) Parnas, D. L.: On the criteria to be used in decomposing systems into modules. In: Communications of the ACM. Vol. 15, Nr. 12, 1972, ISSN 0001-0782, p. 1053–1058, doi:10.1145/361598.361623.

(Shaw et al., 1995) M. Shaw, R. DeLine, D. V. Klein, T. L. Ross, D. M. Young, G. Zelesnik: Abstractions for software architecture and tools to support them. In: IEEE Transactions on Software Engineering. Vol. 21, Nr. 4, 1995, ISSN 0098-5589, p. 314–335, doi:10.1109/32.385970.

(Tanenbaum et al., 2007) Tanenbaum, Andrew S.; Van Steen, Maarten. Distributed systems. Prentice-Hall, p.3-16, 2007.

(Worachet Uttha, 2014) W. Uttha, C. Bertolissi, S. Ranise: Towards a Reference Architecture for Access Control in Distributed Web Applications. In: ESSoS Doctoral Symposium, Springer, 2014.