D5.4

Discovery of Delegations

Document Identification	
Date	23.04.2018
Status	Final
Version	Version 1.0

Related WP	WP 5	**Related Deliverable(s)**	D5.3
Lead Authors	Georg Wagner (TUG)	**Dissemination Level**	PU
Lead Participants	TUG	**Contributors**	FHG
Reviewers	G+D, NLNET		

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

This deliverable describes the discovery of delegations. The discovery of delegation is based on the two deliverables D5.1; the conceptual framework; [1] and D5.3; the publication of delegations; [2].

In Section 5, we describe our consolidated approach, which has been developed as a cooperation between WP 3, 4, and 5. This approach can also be found in [3], and [4]. It describes the general mechanisms for the publication, discovery, and authenticity of trust declarations.

In Section 6, we describe the ideas for the discovery of delegations. We start with a very naïve approach and refine it within this section to get our final discovery process. The naïve approach sends all data to the proxy. This discloses some information about the delegation provider, and can generate lots of data that needs to be transferred to the proxy. The second approach uses the public key of the proxy to only provide those delegations that match the proxy's public key. This would still allow an impostor to impersonate the proxy as the public key is available to everyone. The third approach then implements a challenge-response protocol between the proxy and the delegation provider to prove that the proxy is indeed the owner of the public key.

In Section 7, we provide pointers to relevant delegation data. We provide information about the Associated Signature Container (ASiC) and how delegations can be embedded in the context of this container. We also describe the discovery of a delegation by the Automated Trust Verifier (ATV) and its importance for the verification process.

In Section 8, we provide an example of the proposed discovery process described in Section 6. In our demonstration Alice; the mandator; gives a delegation to Bob; the proxy. The demonstration shows the creation of the delegation, as well as the publication at a delegation provider. Further, Bob needs to discover the delegation at the delegation provider. The demonstration describes the necessary steps Bob has to take and tries to give an insight into the data Bob receives and sees.

2. Document Information

2.1 Contributors

Name	Partner
Georg Wagner	TUG
Martin Hoffmann	NLNET
Olamide Omolola	TUG
Stefan More	TUG
Reinhard Lüftenegger	TUG

2.2 History

Version	Date	Author	Changes
0.1	23.04.2018	Georg Wagner	Initial Version
0.2	26.04.2018	Georg Wagner	Added chapter 6
0.3	28.04.2018	Georg Wagner	Added chapter 7
0.4	2.5.2018	Georg Wagner	Added chapter 8
0.5	8.5.2018	Georg Wagner	Copied text for chapter 5
0.6	9.5.2018	Georg Wagner	Summary, Scope and Conclusion added
1.0	25.5.2018	Georg Wagner	Added reviewers comments

3. Table of Contents

3.1 Table of Figures

3.2 Table of Tables

3.3 Table of Acronyms

ATV	Automated Trust Verifier
DP	Delegation Provider
TSPA	Trust Scheme Publication Authority
TTL	Trust Translation List
DNS	Domain Name System
DNSSEC	DNS Security Extensions
DANE	DNS based Authentication of Named Entries
OEBPS	Open EBook Forum Publication Structure
OCF	OEBPS Container Format
ODF	Open Document Format - Open Office
UCF	Universal Container Format
ASiC	Associated Signature Container
IDPF	International Digital Publishing Forum

4. Scope of the Deliverable

The focus on this document is on the discovery of delegations, which adds another cornerstone in the delegation lifecycle. The mechanisms described in this document are compatible with the publication of delegations as described in [1].

The discovery of a delegation uses both symmetric and asymmetric cryptography to protect the delegation from disclosure to an unauthorized third party.

When publishing a delegation, the delegation is encrypted with a random symmetric key. As this key is needed at the proxy's side for decryption, the key needs to be uploaded to the delegation provider. In order to protect the key from unauthorized disclosure, and thus protect the delegation from disclosure, the key needs to be encrypted so that it can only be opened by the proxy (or intermediary in some cases). Therefore, we use a secret that mandator and proxy both have. In this case it is the private key of the proxy's digital signature certificate. The mandator can encrypt the key used for delegation encryption and send it to the delegation provider, from which the proxy downloads the encrypted delegation and the encrypted key. As the proxy has the corresponding private key to decrypt the key for the delegation, the decryption of the key is straight forward, as well as the later decryption of the delegation.

This process is good enough, if the two parties mandator and proxy know which delegation is the issued one and in case of multiple delegations from the same mandator the correct one. In order to find the correct delegation we introduce a discovery process for delegations that can work on the encrypted data and show examples how this discovery process works in detail.

5. Terminology

This section contains the terminology used throughout the deliverable. It is important, that all readers of this document have the same understanding of the concepts. Furthermore, it makes communication between partners and participants easier. Parts of the terminology are reused from D2.14 [1] and D5.1 [2] to provide a comprehensive terminology dictionary in this document.

5.1 Delegation Provider

The Delegation Provider is a web component, which is used to publish delegations. It is used by other components by querying it. The Delegation Provider contains a repository for delegations and one for revoked delegations.

5.2 Verifier

Person or Entity that wants to check if a transaction is valid or not.

5.3 Automated Trust Verifier

Component used by the Verifier to verify documents. This component queries the delegation provider to verify the existence of the delegation and verifies that the delegation has not been revoked.

5.4 Mandator

Person or entity empowering another person or entity to act on behalf of itself. The Mandator is the creator of delegations and publishes delegations at the Delegation Provider of his choice.

5.5 Proxy

Person or entity empowered by a Mandator to act on behalf of another person or entity. The Proxy is the person or entity executing the delegation.

5.6 Intermediary

Person or entity empowered by a Mandator to find another person or entity to act on behalf of the Mandator. The intermediary acts as a link between Mandator and Proxy in the selection process. The parameters of the delegation are provided by the Mandator.

6. A Consolidated Approach to Publishing Trust related Information in the DNS

The trust framework created by LIGHTest verifies the trustworthiness of an electronic transaction by attempting to establish a chain of trust from a set of pre-configured, well-known trust sources to this transaction. The links in this chain are assurances that the trust into a source can be extended to another entity. This section looks at the basic properties these assurances exhibit independently of their concrete contents and introduces an underlying, fundamental framework.

Within the LIGHTest project, three working packages look at different aspects of such assurances. WP3 examines the most basic form where a trust source known as a trust scheme declares that some issuer of trust services, such as certificates or time stamps, conforms to the conditions and rules set out by the scheme and thus extends trust placed into it onto this trust service. WP4 assesses the relationship between multiple trust schemes allowing a trust scheme or some other trusted entity to declare if and how trust into one scheme extends to trust into another scheme. WP5, finally, looks into how individual entities can empower other entities to act on their behalf– extending trust into themselves onto that other entity within certain well-defined limits.

6.1 Trust Declarations

However different they may appear, each of these aspects follows a similar pattern: some entity makes a *trust declaration* stating that trust into a certain entity extends to another entity, possibly providing conditions and limits of such an extension of trust. To simplify further discussion, it will be helpful to label the three entities involved in the process. The entity issuing the declaration shall be the *originator,* the entity that is already trusted is the *source* and the entity trusted as a result of the declaration is the *target.*

Within the aspects discussed as part of the LIGHTest project, in many cases the originator of a declaration is identical to the source. This is certainly true for trust membership publication in WP3, where the trust scheme itself declares which trust services are a member. In the aspects of the other two work packages, similarly originator and source are most often identical. However, there may be use cases where this is not the case. For instance, a third party may declare a trust translation independently of the trust schemes that are source or target of this declaration. Similarly, a third party may declare a trust delegation. For instance, business registries often provide information about individuals that are allowed to sign on behalf of a company. Such information can be modeled as third-party trust delegation.

This definition specifies a single declaration to extend trust from exactly one source to exactly one target. In practice, the document formats used often contain multiple declarations according to this definition. For instance, the trusted lists defined in ETSI TS 119 612 [1] used for the declarations in WP3 contain a list of all the targets a trust scheme as a source wishes to extend trust to. This published form of declarations shall be called *trust declaration documents.* At least in principle each such document can contain one or many declarations with any number of sources and targets.

For a trust declaration to be considered when building the chain of trust during validation of an electronic transaction, the declaration itself needs to be trusted. If a declaration document is treated like any other electronic transaction, this trust can in turn be established through verification using the LIGHTest framework. That is, there needs to be a chain of trust from pre-configured trust sources to the originator of the declaration document for the particular aspect of the declaration in question.

Note that in most cases where the originator of the declaration is identical to the source of the declaration, this chain exists implicitly and no extra checks are needed. It may, however, be possible that conditions for trusting the source as such and trusting declarations made by it are different and thus need to be verified independently.

6.2 Publication of Trust Declarations

When using trust declarations for verifying an electronic transaction, a validator needs to construct a chain of trust declarations leading from any of the trusted entities to those entities appearing in the transaction. It does so by recursively finding and adding applicable trust declarations that have an originator that is either a trusted entity or the target of the declaration is already part of the chain. In order to do this in an unaided, automatic way, the validator needs a way to gain access to all declarations that are potentially usable in this process.

There are two fundamental strategies for the verifier to find declarations when they are needed: a declaration could either be actively supplied as part of the input or configuration or it is left to the verifier itself to discover it.

The prime example for the former case is that a declaration is provided as part of the electronic transaction to be verified. This is particularly useful if the creator of the transaction is aware that the declaration is necessary for verification. For instance, if an entity signs a document in their function as a proxy, the transaction will only ever verify if the declaration of trust delegation– the mandate– is known to the verifier. The proxy may very well include the mandate in the transaction right away.

For declarations that are potentially applicable to a large amount of transactions or if the sender of the transaction doesn't know the trusted entities the receiver will base their verification on, such a strategy isn't very practical. Instead, it is better to make the declarations available publicly and provide means for a verifier to discover how and where it can retrieve them. The verifier can then decide itself which declarations it needs and try and find them as needed.

While the most likely method for publishing currently is the Hypertext Transfer Protocol (HTTP), other protocols may become available in the future. An extensible standard exists to describe both the method used for accessing a resource and all necessary parameters for a successful retrieval in the form of a Uniform Resource Identifier (URI). It encodes all information into a single string which can be easily stored or transmitted.

6.3 Discovering Trust Declarations

In order to build a chain of trust from published declarations, the verifier needs to be able to discover their existence. Given a transaction and a set of already trusted entities, this chain can be built from two sides: either the verifier starts with a trusted entity, tries to discover all the declarations that have this entity as their source, and repeats this process until it arrives at a declaration that includes the transactions as its target or runs out of declarations to apply. Alternatively, it can start with the transaction, attempts to discover all declarations that have the transaction as their target, and continues by recursively trying to find declarations that have the sources of already discovered declarations as their target until it arrives at an already trusted entity as a declaration's source or, again, runs out of declarations.

In both cases, the verifier needs a mechanism to search for the URI of a declaration based on a given entity. Such a mechanism will have to take some information that identifies that entity as input. Given that entities are typically identified by X.509 certificates, it should be possible to include such identifying information in the entity's certificate. One of the possible options is to use a domain name as the entity's identifier. This has the advantage that the name can be used directly as a search input for the DNS, allowing to use the DNS as a global, highly available, distributed, and independently managed data store.

A domain name can be stored in a certificate either in the subject alternative name or issuer alternative name extensions. The subject alternative name indicates the domain name identifying the entity using the certificate while the issuer alternative name identifies the entity having issued the certificate.

Using this domain name as input, the URIs to the declarations for differing aspects should be stored in the DNS. As the DNS uses record types to distinguish between different types of information stored for a domain name, one option is to register an individual record type for each aspect of trust declarations. However, this would clutter the space of types. As the data stored is the same in every case – a URI – an alternative approach can be used whereby the aspect is encoded as a prefix to the domain name. This is already used for instance with the SRV record type for discovering the host name and port where a certain networking service is available for a domain. As such services sometimes are available over different transport protocols, a two-layer prefix of the form *_service._protocol* is used where the latter describes the transport protocol to be used and the former the networking service.

In keeping with this concept, LIGHTest proposes to use the DNS for providing pointers to all kinds of declarations via a pair of prefixes of the form *_aspect._application*. Here, the type of application for which a declaration is published is the second part of the prefix while the first part defines the particular aspect within that application. For LIGHTest itself, the application is, of course, trust-related declarations, which is identified via using the literal label *_trust* as the second part. Each of the aspects of trust declarations defines its own label to be used as the first part.

Under the name constructed by concatenating the prefix with the entity's own identifying domain name, the entity can now publish pointers to trust declarations of the corresponding aspect that relate to it. It can use the already existing URI resource record type for this purpose. This type is defined in section 4 of RFC 7553 [2]. Its record data contains exactly one URI. While section 5 of the RFC describes a different use of the record, this use is limited by a different set of prefixes, allowing its reuse for declaration publication based on the prefixes defined above.

If the entity in question is not the originator of a declaration it may not control the URI under which the declaration is published. In this case, it may be beneficial to only point to the originating entity rather than burden itself with tracking whether the originators URI has changed. This can be done easily if the originator is an entity identified by a domain name, too. In this case, instead of publishing URI resource records under its domain name prefixed by the declaration aspect, the entity will publish PTR resource records. Such record types, part of the original DNS specification in RFC 1035 [3], contain another domain name as their record data. If such records are present, they instruct the verifier to continue discovery for declaration at the entity identified by these domain names. Note that as these names in the PTR record's data refer to the specific application and aspect and as such are already prefixed with the correct declaration

prefix. While not used in the LIGHTest framework as yet, this would allow to provide references between different aspects.[1]

6.4 Authenticity of Trust Declarations

If a verifier retrieves a declaration from somewhere in the network, it needs to make sure that the data it received is indeed the declaration made by the originator. Since the URI resource records are stored under the domain name identifying the originator, it is reasonable to assume they are authentic if DNSSEC validation succeeds, guaranteeing that the URI is indeed the one intended by the originator.

In the next step, where the verifier contacts the server indicated in the URI, it needs to ensure that it communicates with the correct server. When using encrypted transport via the TLS protocol, the server will identify itself via a certificate. In order to deal with shortcomings of the traditional method of certificate verification, a protocol called DNS-based Authentication of Named Entities or DANE allows a server operator to publish information in DNS about the certificates used.

Since, however, the server is not necessarily operated under authority of the originator of the declaration – for instance because the declarations are hosted by a third party that provides better availability, this does not guarantee that the declaration received is indeed the one that the originator intended.

This final link can be provided if the declaration itself is a signed document. The originator can then publish the certificates that it uses for signing declarations of a certain aspect using a slightly adapted version of the DANE protocol.

To do so, it adds SMIME resource records under the same domain name it placed the URI records pointing to the declarations. These records define conditions a certificate has to fulfill to be accepted. By placing these records, the originator declares that all documents retrieved via the pointers have to verify considering these conditions.

[1] The initial version of the Consolidated Framework as presented in deliverable D3.3 proposed to let the PTR records refer to the domain name identifying the entity. After discussion, it was decided that the updated approach presented here is both more correct as it points to the exact name where discovery continues and, as mentioned, more flexible as a generic means of discovery.

7. Discovery of Delegations

In [1] we describe the publication of a delegation at the delegation provider. The publication process makes the delegation available for later discovery. Figure 1 shows the process of the publication and the discovery of the delegation. First, the mandator creates a delegation for the proxy, who represents the mandator in the end. After the selection of the proxy and the creation of the initial delegation file, the mandator creates a symmetric encryption key *k*. With this key, the mandator encrypts the delegation. Further, the mandator encrypts the key *k* with the proxy's public key. That way, the mandator and the proxy can share the delegation key without a lot of communication. Second, the mandator loads the encrypted delegation data, encrypted key, and the public key of the proxy onto the delegation provider. After this, the third step is already the discovery of the delegation. The proxy executes the discovery process, which we describe in the remainder of this section. The fourth and last step in this figure is the download of the delegation, which also happens during the discovery process of the delegation.

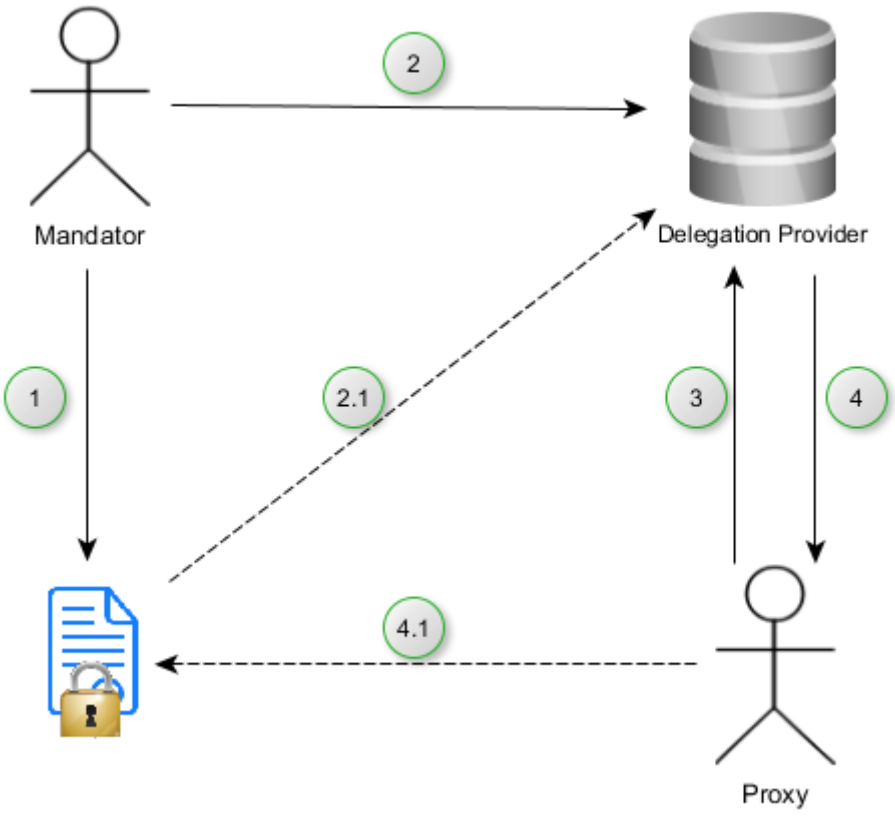

Figure 1: Process of Publication and Discovery of Delegations

7.1 Prerequisites

In order to discover delegations, we must meet the following prerequisites:

- A mandator and a proxy have a relation with each other. In this case the delegation.
- The delegation is encrypted with a symmetric key.
- The symmetric key is encrypted with the proxies public key.
- Both delegation and key are published at the delegation provider.

After the publication of the delegation we have to be able to discover it. Because we store everything encrypted, we need to either be able to search on encrypted data or provide support mechanisms to find the delegation. Two parties have to discover the delegation. One party is the proxy and the other one is the ATV. While the ATV needs to discover the delegation within the transaction, the proxy needs to discover it at the delegation provider and embed it in the signature container, so that the ATV is able to discover the delegation in this container.

In the remainder of this section we introduce some methods how discovery can work, and elaborate their usefulness for LIGHTest and describe the method we will finally implement.

7.2 The Naive Approach

This approach will disclose all saved delegations to the proxy, who then has to reconstruct the key and the delegation. The proxy gets all available delegation data from the delegation publisher and can create his own local mirror of delegation data. Revocation will become useless and a bad proxy can then go ahead and use this delegation provider instead, where the delegation is still valid.

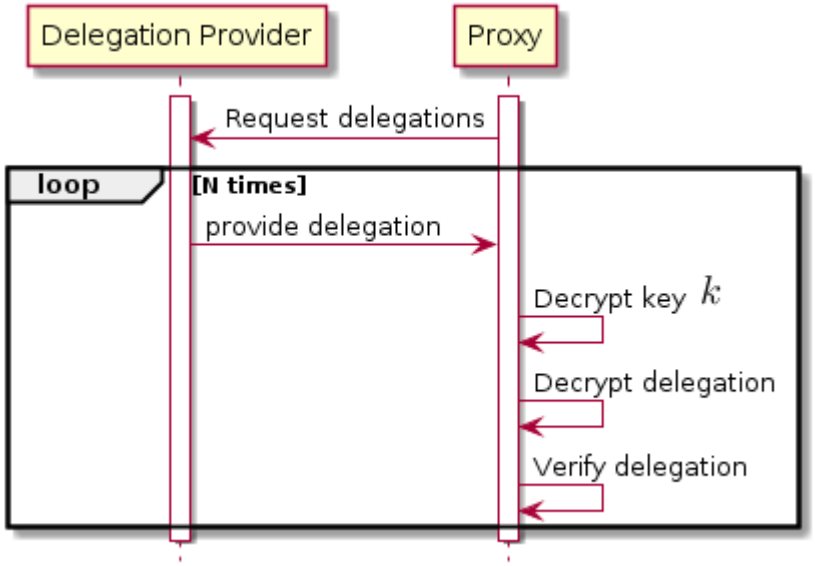

Figure 2: Naïve approach

To discover a delegation, the proxy requests all delegations at the delegation provider. Neither proxy nor delegation provider have any detailed information about the content of the encrypted delegation. The delegation provider sends all data to the proxy and hopes that the desired delegation is within the send data. The proxy receives a delegation and tries to decrypt the key with his secret key. With the help of the decrypted key, the proxy tries to decrypt the delegation itself and verifies the content. The content is valid if the decrypted data contains a valid XML structure as specified in [4].

We call this approach naïve, because the delegation provider discloses all delegations to the proxy. The proxy may only be able to successfully decrypt the delegations to which he holds the corresponding keys. Unfortunately all other delegations are transmitted to the proxy as well. This poses the threat, that the proxy can start his own delegation provider, based on the data from the original delegation provider. The proxy can become a new delegation provider very easily. The mandator, intermediary, and original delegation provider do not have any control over the newly created delegation provider. As the new delegation provider is completely under the control of the proxy, other proxies can use this particular delegation provider if they want to overcome some limitations that they have with the original delegation provider, like a revoked delegation. The new delegation provider knows nothing about revocation.

Furthermore, besides the threat of creating a new delegation provider, the amount of data transmitted can be huge. A delegation usually has a size of several kilobytes. Based on the data for new company registrations in Austria, a delegation provider for business transactions, that only lists the CEO of a company, will get about new 30,000

delegations per year. All this data needs to be sent to the proxy. The proxy then has to store each delegation locally and decrypt it. This search is then conducted in O(n) time and includes an expensive decryption operation, which makes the search very long. Another problem is space for the proxy. If the proxy does not have enough space left, then we cannot search the right delegation at all.

The naïve approach is not suited for the discovery of delegations.

7.3 Extending the Data of a Delegation Approach

For this approach, we take the problems from the previous sections, and try to solve them. We can observe that if the delegation provider sends all the stored delegations to the proxy, the proxy has a huge amount of data to search through. We extend the data on a delegation that is saved at the delegation provider.

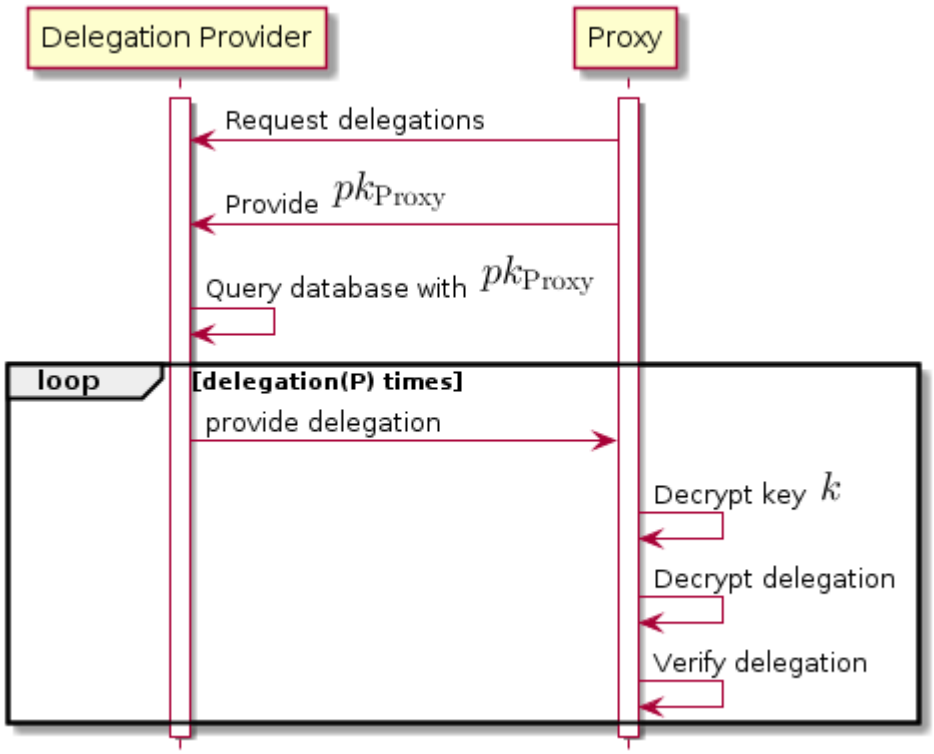

Figure 3: Extended approach

The proxy requests delegations at the delegation provider. The proxy further provides a hash of his own public key to the delegation provider as additional information. With the public key as additional information, the delegation provider can now query its database and search if delegations for this public key do exist. If delegations exist, the delegation provider will return them to the proxy. The proxy can then decrypt the key and the delegation data and finally select the correct delegation data.

In this approach, the delegation provider has minimum information about the proxy available, namely the public key of the proxy. As the public key of the proxy can be available to anyone, the delegation provider will send all the delegations for a particular public key to anyone who is in possession of this key. An attacker can easily test if the delegation provider has delegations for a certain public key.

Similar to the previous approach, the malicious proxy can generate a mirror delegation provider for a known proxy with the same implications as before. If the actual delegation provider revokes the delegation, the mirror delegation provider does not need to update this information, thus having a second delegation available, which would still verify as valid in case of verification.

7.4 A Challenge Response Authentication Protocol

The previous approach is still not ideal, as everyone who is in possession of the public key of a proxy can find out about the delegations for the proxy. This approach extends the previous approach only allowing data transfer between the delegation provider, the designated proxy, and an unauthorized third party.

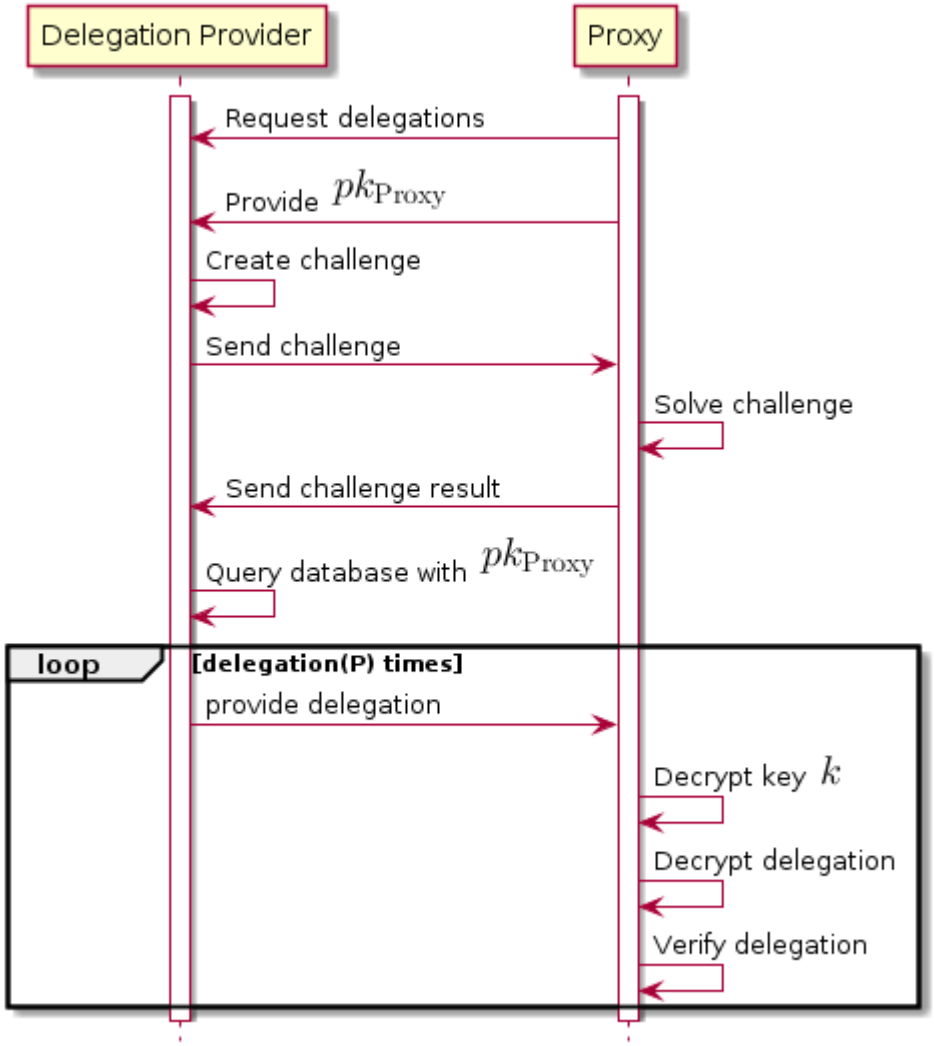

Figure 4: Challenge-Response Approach to Authenticate the Proxy based on its public/private key pair

In this approach, the delegation provider needs to create a challenge that the proxy then has to solve. The delegation provider already has the public key of the proxy stored in its database. The delegation provider chooses a random secret, and encrypts it with the proxy's public key and sends it to the proxy. Since the proxy is in possession, he should be able to decrypt the secret correctly. If an impostor tries to decrypt the secret, the impostor should fail, as he is not in possession of the proxy's private key and unable to find the right challenge word. In any case the proxy does not know the solution to the challenge. Only the delegation provider knows the solution to the challenge. The proxy signs the answer with his secret key and sends it back to the delegation provider. The delegation provider now verifies the signature and checks the answer provided by the proxy. In case of a correct answer, the delegation provider can now send the available delegation data for a particular public key to the proxy. The proxy then can decrypt the

key and the data and search for the correct delegation locally. If the answer to the challenge is incorrect, because the proxy is an impostor and not in possession of the required private key, the impostor sends a wrong answer to the challenge back to the delegation provider. The delegation provider then knows that the request is coming from an impostor and that it does not need to show and disclose any delegations.

8. Pointers to Relevant Delegation Data

This section describes pointers to relevant delegation data. The ATV uses these pointers during the verification of transactions. The pointers to relevant delegation data describe the type of the delegation, as well as the content of the delegation. They provide information about the delegation provider and information about the revocation list for the discovered delegation.

8.1 Pointers in an Associated Signature Container

Signature containers provide a means to link detached signatures with its associated data, i.e. the data over which the signature has been created. ASiC is currently the chosen container for use in Light[est]. ASiC provides a standardized container format for linking detached digital signatures to their associated data. It was standardized by ETSI in ETSI TS 102 918 [7]. However, some containers were proposed before ASiC and they were used in diverse domains. Some of the other containers are OEBPS Container Format (OCF), Open Document Format – Open Office (ODF) and Universal Container Format (UCF). ASiC is essentially a derivative of ODF, UCF and OCF.

A valid OCF container must contain a META-INF root folder that defines the properties of the container. An example taken from the format specification document [7] is given below.

```
META-INF/
   container.xml – Note: includes multiple <rootfile> elements
   [manifest.xml]
   [metadata.xml]
   [signatures.xml]
   [encryption.xml]
   [rights.xml]
OEBPS/
   Great Expectations.opf
   cover.html
   chapters/
      chapter01.html
      chapter02.html
      … other OPS files for the remaining chapters …
PDF/
   Great Expectations.pdf
```

Two physical container methodologies used to represent the OCF format is the Zip Container and the File System Container [3]. This format was standardized by the IDPF.

The OpenDocument Format (ODF) [4] format specifies the standard format for text, spreadsheet, database front-end, presentation, formula, chart and drawing documents[ref]. It was created as a successor to the vendor-specific formats such as .doc, .wpd, .xls and .rtf. A valid ODF package must contain at least a content.xml and styles.xml file. These describe the structure and style of the document. The ODF format

is used in the OpenOffice software package and is supported by Microsoft Office. The ODF format was standardized by OASIS and adopted by ISO/IEC JTC1 SC34.

Universal Container Format (UCF) [6] collects related set of files into a single container. The UCF container also has a META-INF folder contained at the root of the container. This folder contains information about the content, signature and encryption of the container. Adobe Systems standardized this format.

An ASiC container is a data container that incorporates a group of file objects and their associated digital signatures/and or time assertions using the ZIP format. The ZIP format is a good way of storing and transferring files and folders.

The internal structure of an ASiC container includes the following:

- A **root** folder for all the container content, including folders that reflect the structure of the content
- A "**META-INF**" folder inside the above mentioned root folder that holds files that contain metadata about the content, including the associated signature/and or time assertion files

The associated signature and time assertion files that are contained in the "META-INF" folder are placed in such a way that they can be verified against their associated file objects. Two kinds of ASiC containers are defined: ASiC-S (ASiC-Simple) and ASiC-E (ASiC-Extended). The major difference between the two containers is that ASiC-S contains only one data object while the ASiC-E contains multiple data objects. An example of ASiC-S taken from the specification document is shown in Figure 5.

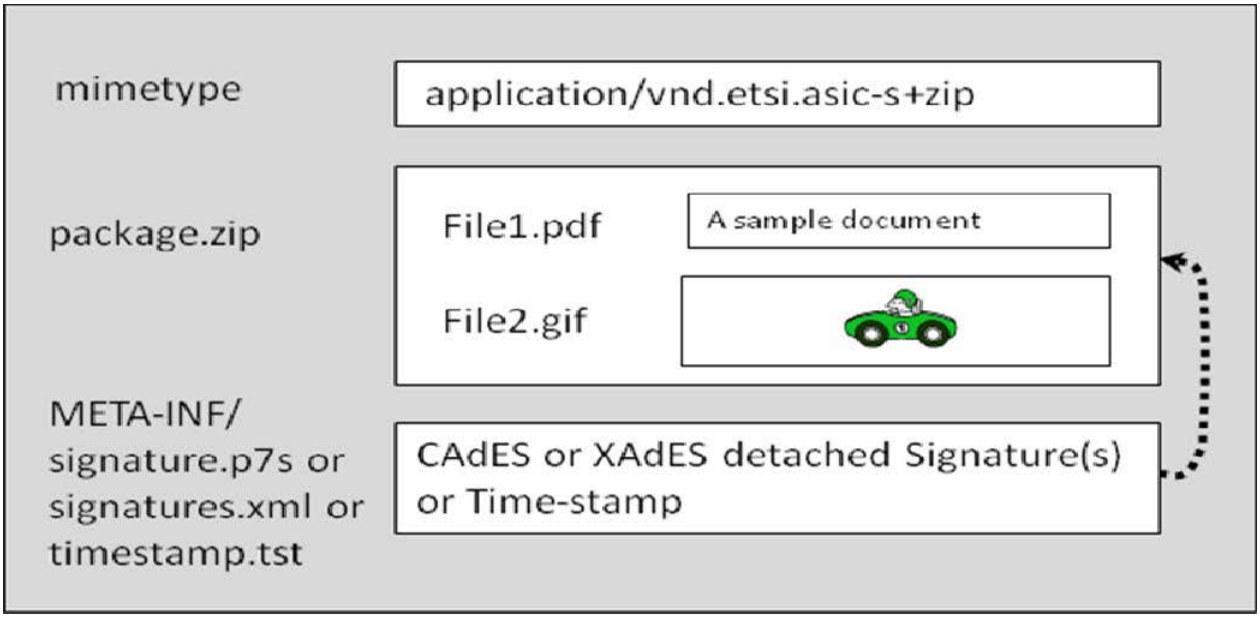

Figure 5: Example of an ASiC container

An extension to support LIGHT[est] will be to add a new XML data structure in the META-INF folder describing the delegation information (in XML) proposed by LIGHT[est]. The structure of a delegation in XML, taken from [8], is shown below:

```
<?xml version="1.0"?>
<delegation version="1.0">

    <!-- Mandatory Information -->
    <issuedDate> 2017-05-14T23:59:59 </issuedDate>
    <proxy> Bob </proxy>
    <issuer> Alice </issuer>
    <intermediary />
    <substitutionAllowed>false</substitutionAllowed>
    <delegationAllowed>false</delegationAllowed>
    <validity>
        <notBefore> 2017-06-15T00:00:00 </notBefore>
        <notAfter> 2017-10-06T23:59:59 </notAfter>
    </validity>

    <domain name="purchase" version="0">
       ...
    </domain>

    <ds:signature>
    </ds:signature>

</delegation>
```

This format allows the detection of the delegation, together with the required attributes, Mandator, Intermediary, and Proxy. It further provides information about the validity period of the delegation within the file. The file is also signed by the Mandator; or Intermediary; to provide prove that no information in the file has been changed.

9. Demonstration of the Discovery

In this section, we provide a demonstration of the discovery at the delegation provider. For the purpose of this demonstration, we create a simple delegation between a mandator and a proxy. We publish the delegation at a delegation provider and discover it at the delegation provider among others.

The delegation is between Alice and Bob. Alice is the mandator and gives a delegation for ordering goods to Bob. Alice further defines that Bob can order goods up to a value of 200,000 and can do this for the next two years. The XML below shows the full delegation in plain text.

Alice also chooses the delegation provider of her choice. For the purpose of this demonstration she chooses the delegation provider example.com which will be located at *https://delegation.example.com*.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<ns2:delegationType xmlns:ns2="http://www.w3.org/2000/09/xmldsig#">
<information/>
<issuedDate>2018-05-07T16:53:11.847+02:00</issuedDate>
<proxy>
MIIDQjCCAjUGByqGSM44BAEwggIoAoIBAQCPeTXZuarpv6vtiHrPSVG28y7FnjuvNxjo6sSWHz79
NgbnQ1GpxBgzObgJ58KuHFObp0dbhdARrbi0eYd1SYRpXKwOjxSzNggooi/6JxEKPWKpk0U0CaD+
aWxGWPhL3SCBnDcJoBBXsZWtzQAjPbpUhLYpH51kjviDRIZ3l5zsBLQ0pqwudemYXeI9sCkvwRGM
n/qdgYHnM423krcw17njSVkvaAmYchU5Feo9a4tGU8YzRY+AOzKkwuDycpAlbk4/ijsIOKHEUOTh
jBopo33fXqFD3ktm/wSQPtXPFiPhWNSHxgjpfyEc2B3KI8tuOAdl+CLjQr5ITAV2OTlgHNZnAh0A
uvaWpoV499/e5/pnyXfHhe8ysjO65YDAvNVpXQKCAQAWplxYIEhQcE51AqOXVwQNNNo6NHjBVNTk
pcAtJC7gT5bmHkvQkEq9rI837rHgnzGC0jyQQ8tkL4gAQWDt+coJsyB2p5wypifyRz6Rh5uixOdE
vSCBVEy1W4AsNo0fqD7UielOD6BojjJCilx4xHjGjQUntxyaOrsLC+EsRGiWOefTznTbEBplqiuH
9kxoJts+xy9LVZmDS7TtsC98kOmkltOlXVNb6/xF1PYZ9j897buHOSXC8iTgdzEpbaiH7B5HSPh+
+1/et1SEMWsiMt7lU92vAhErDR8C2jCXMiT+J67ai51LKSLZuovjntnhA6Y8UoELxoi34u1DFuHv
F9veA4IBBQACggEAA7vBj/jBnCX1hyhmoDTSfNZS3+yZbMVlOSXCxC656uE/KJMmhSCOWf+wD3Rq
KUtvhAbibnYNe+PhfWUcp3awA6pd0reqZo1hYW3sZoAt+W2z6J0bxd9I2z5wUfM1g/WS8rl0JdBy
vzbTfQYkMN0lNh7s5FrQk1qt5RKW/SyVwxdpCPuunM7lQxrZL0nxMMSeA5Yk3bGoJ739YaVNimE7
qr/pnFP6+rlAQCNntj9ySgFJShBDxUI5EyTysWrIzvkl9cOlRdsfiLBkaXEG6TiQ0QZoh2pahfr8
jPteksBfDKfU9RI/1OH24+ExUf+OQFS8cHQ/0h+MOoECUBAjypqsxw==
</proxy>
<issuer>
MIIDQjCCAjUGByqGSM44BAEwggIoAoIBAQCPeTXZuarpv6vtiHrPSVG28y7FnjuvNxjo6sSWHz79
NgbnQ1GpxBgzObgJ58KuHFObp0dbhdARrbi0eYd1SYRpXKwOjxSzNggooi/6JxEKPWKpk0U0CaD+
aWxGWPhL3SCBnDcJoBBXsZWtzQAjPbpUhLYpH51kjviDRIZ3l5zsBLQ0pqwudemYXeI9sCkvwRGM
n/qdgYHnM423krcw17njSVkvaAmYchU5Feo9a4tGU8YzRY+AOzKkwuDycpAlbk4/ijsIOKHEUOTh
jBopo33fXqFD3ktm/wSQPtXPFiPhWNSHxgjpfyEc2B3KI8tuOAdl+CLjQr5ITAV2OTlgHNZnAh0A
uvaWpoV499/e5/pnyXfHhe8ysjO65YDAvNVpXQKCAQAWplxYIEhQcE51AqOXVwQNNNo6NHjBVNTk
pcAtJC7gT5bmHkvQkEq9rI837rHgnzGC0jyQQ8tkL4gAQWDt+coJsyB2p5wypifyRz6Rh5uixOdE
vSCBVEy1W4AsNo0fqD7UielOD6BojjJCilx4xHjGjQUntxyaOrsLC+EsRGiWOefTznTbEBplqiuH
9kxoJts+xy9LVZmDS7TtsC98kOmkltOlXVNb6/xF1PYZ9j897buHOSXC8iTgdzEpbaiH7B5HSPh+
+1/et1SEMWsiMt7lU92vAhErDR8C2jCXMiT+J67ai51LKSLZuovjntnhA6Y8UoELxoi34u1DFuHv
F9veA4IBBQACggEARwI6up7aVo3gyBOIRD03P32m1jfeBJCkbQTWJWHXDg8hMhmH94kZqyvpZVIF
jWbTb0c/HAsEnvtbaHErcP4GwTR//wdZVvyaCXj8Ki8Ke19yvyuJIMdOB6cFY7Q5raeO0WZ6fwEn
oYZ0jYTZ9JRIOtCtEuOS9OKCqMxrcIwqOQjyINd6AbVgCV/+MksdSatHYPJGl2tOi1fx4HaSLo8c
ckSyu48K3kQG1NSNH9lLVE1TcSFxW8aTXK0R/D3q7FfNXI7zzII1BiBDGPgqiahj4qzauuboYBc/
```

```
UtM6tzQ5pZlxGdDX7cDtzAA/jDJ1ZnAZzHFJYaOnWFDeKAY+tzKEHQ==
</issuer>
<substitutionAllowed>false</substitutionAllowed>
<delegationAllowed>false</delegationAllowed>
<validity>
    <notBefore>2018-05-14T16:49:53.520+02:00</notBefore>
    <notAfter>2020-05-14T16:50:01.366+02:00</notAfter>
</validity>
<domain name="Ordering" version="1.0">
    <Ordering>
        <ammount>200000</ammount>
    </Ordering>
</domain>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#" Id="mandator">
    <SignedInfo>
        <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-
c14n-20010315#WithComments"/>
        <SignatureMethod Algorithm="http://www.w3.org/2009/xmldsig11#dsa-
sha256"/>
        <Reference URI="">
            <Transforms>
                <Transform
Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
            </Transforms>
            <DigestMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>

<DigestValue>saUfKs8x7LBLf8vcFZgwH3rCG8OHyiWFaS/f2luQoeg=</DigestValue>
        </Reference>
    </SignedInfo>

<SignatureValue>cn8JSTaEWWG4yBlRc8D3ZpcHku16IZrXgdy95H5xTNxFV/9AB4sY/iOOdgCLqG
05m4AIXTiIKHs=</SignatureValue>
    <KeyInfo>
        <KeyValue>
            <DSAKeyValue>
                <P>
j3k12bmq6b+r7Yh6z0lRtvMuxZ47rzcY6OrElh8+/TYG50NRqcQYMzm4CefCrhxTm6dHW4XQEa24
tHmHdUmEaVysDo8UszYIKKIv+icRCj1iqZNFNAmg/mlsRlj4S90ggZw3CaAQV7GVrc0AIz26VIS2
KR+dZI74g0SGd5ec7AS0NKasLnXpmF3iPbApL8ERjJ/6nYGB5zONt5K3MNe540lZL2gJmHIVORXq
PWuLRlPGM0WPgDsypMLg8nKQJW5OP4o7CDihxFDk4YwaKaN9316hQ95LZv8EkD7VzxYj4VjUh8YI
6X8hHNgdyiPLbjgHZfgi40K+SEwFdjk5YBzWZw==
                </P>
                <Q>uvaWpoV499/e5/pnyXfHhe8ysjO65YDAvNVpXQ==</Q>
                <G>
FqZcWCBIUHBOdQKjl1cEDTTaOjR4wVTU5KXALSQu4E+W5h5L0JBKvayPN+6x4J8xgtI8kEPLZC+I
AEFg7fnKCbMgdqecMqYn8kc+kYebosTnRL0ggVRMtVuALDaNH6g+1InpTg+gaI4yQopceMR4xo0F
J7ccmjq7CwvhLERoljnn08502xAaZaorh/ZMaCbbPscvS1WZg0u07bAvfJDppJbTpV1TW+v8RdT2
GfY/Pe27hzklwvIk4HcxKW2oh+weR0j4fvtf3rdUhDFrIjLe5VPdrwIRKw0fAtowlzIk/ieu2oud
Syki2bqL457Z4QOmPFKBC8aIt+LtQxbh7xfb3g==
                </G>
                <Y>
RwI6up7aVo3gyBOIRD03P32m1jfeBJCkbQTWJWHXDg8hMhmH94kZqyvpZVIFjWbTb0c/HAsEnvtb
aHErcP4GwTR//wdZVvyaCXj8Ki8Ke19yvyuJIMdOB6cFY7Q5raeO0WZ6fwEnoYZ0jYTZ9JRIOtCt
EuOS9OKCqMxrcIwqOQjyINd6AbVgCV/+MksdSatHYPJGl2tOi1fx4HaSLo8cckSyu48K3kQG1NSN
```

H9lLVE1TcSFxW8aTXK0R/D3q7FfNXI7zzII1BiBDGPgqiahj4qzauuboYBc/UtM6tzQ5pZlxGdDX
7cDtzAA/jDJ1ZnAZzHFJYaOnWFDeKAY+tzKEHQ==

```
                </Y>
            </DSAKeyValue>
        </KeyValue>
    </KeyInfo>
</Signature>
</ns2:delegationType>
```

In the next step, this delegation data is encrypted and uploaded to the delegation provider at *https://delegation.example.com*, together with the encrypted key and the public key of the proxy which allows the proxy to search for the key afterwards.

To achieve this, The delegation is encrypted with the secret key *CoffeBarCoffeBar* and send to the delegation provider. The secret key is given in ASCII encoding here. At the delegation provider, we can find the encrypted delegation. Further, the mandator automatically sends the encrypted key. In order for the key to be available by the proxy, the key is encrypted with the proxy's public key. The encryption of the key leads to the following values: VioLBrFUzDJ5liTrrrq0xqbBZ8rhriMx+/7UGv/1D0Y=.

After the publication of the delegation, the mandator is not needed for the discovery anymore. We have now successfully shown steps 1 to 2.1 from Figure 1. Now the proxy can start discovering the delegation. Therefore, the proxy only needs the information where the delegation provider is located. In this case, Bob knows that Alice uses https://delegation.example.com as her delegation provider.

Bob now queries the delegation provider and sends his public key. With this public key, the delegation provider generates a random challenge and encrypts it with Bob's public key and sends it back to Bob. Now that only Bob is in possession of the secret key, Bob can decrypt the challenge, sign it, and send it back to the delegation provider. The delegation provider detects the correctness of the answer and will send the all the delegations that are saved for the same public key and secret keys to Bob. Bob can now decrypt the keys with his secret keys, and the delegation with the decrypted key.

10. Conclusion

In this deliverable, we have shown the process for the discovery of delegations. The discovery of a delegation requires the delegation provider and the proxy as actors in this process. We have shown how to develop the delegation process and started with a naïve approach where all data from the delegation provider would be send to the proxy. We further defined our approach to a point where only the available delegation data for a proxy is transmitted. We also provide a challenge-response authentication to discover if the proxy holds the correct private key.

We show how we can discover a delegation in a transaction. For this discovery we use an ASiC container. This container holds all relevant data for the transaction. We provide the details of the container and the new extension for the delegation.

In the following demonstration, we show how two parties, Alice and Bob, can publish and discover a delegation at the delegation provider. In our demonstration Alice wants to delegate some of her tasks to Bob, so she create the delegation. When Bob signs a document he now has to embed the delegation data in the signature container and needs to get it from the server first.

11. References

[1] The LIGHTest Project, “D5.1 Conceptual Framework for Delegations,” 2017.

[2] The LIGHTest Project, D5.3 - DNS-based Publication of Delegations, 2018.

[3] The LIGHTest Project, D3.4 - Discovery of Trust Scheme Publication Authorities, 2018.

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

[5] O. K. P. Falstrom, The Uniform Resource Identifier (URI) DNS Resource, RFC 7553, Internet Engineering Task Force, June 2015.

[6] P. Mockapetris, Domain names – implementation and specification, RFC 1035, Internet Engineering Task Force, November 1987.

[7] International Digital Publishing Forum, “Open Container Format (OCF) 2.0.1 v1.0.1,” International Digital Publishing Forum, 2010.

[8] International Organization for Standardization, “Information technology — Open Document Format for Office Applications (OpenDocument) v1.2,” International Organization for Standardization, 2015.

[9] Adobe Systems, “XMP Specification Part 3: Storage in Files,” Adobe Systems, 2016.

[10] European Telecommunications Standards Institute, “Electronic Signatures and Infrastructures (ESI): Associated Signature Containers (ASiC),” European Telecommunications Standards Institute, 2011.

[11] G. Wagner, O. Omolola and S. More, “Harmonizing Delegation Data Formats,” in *Open Identity Summit 2017*, Karlstad, Sweden, 2017.

12. Project Description

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

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

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

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

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

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