D6.6

Open Source Tool for Automated Trust Verification

<table>
<thead>
<tr>
<th>Document Identification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
</tr>
<tr>
<td>Status</td>
</tr>
<tr>
<td>Version</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Related WP</th>
<th>WP 3, 4, 5, 6, 8, 9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Related Deliverable(s)</td>
<td>D6.5, D5.5, D6.1, D6.2, D2.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Lead Authors</th>
<th>TUG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dissemination Level</td>
<td>PU</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Lead Participants</th>
<th>TUG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contributors</td>
<td>TUG, FHG, DTU</td>
</tr>
</tbody>
</table>

| Reviewers | TUBITAK, CORREOS |

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

This document and its content are the property of the Lightest Consortium. All rights relevant to this document are determined by the applicable laws. Access to this document does not grant any right or license on the document or its contents. This document or its contents are not to be used or treated in any manner inconsistent with the rights or interests of the Lightest Consortium or the Partners detriment and are not to be disclosed externally without prior written consent from the Lightest Partners.

Each Lightest Partner may use this document in conformity with the Lightest Consortium Grant Agreement provisions.
1. Executive Summary

This document describes how to obtain, build, and maintain the source code for the automated trust verifier (ATV) component.
2. Document Information

2.1 Contributors

<table>
<thead>
<tr>
<th>Name</th>
<th>Partner</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stefan More</td>
<td>TUG</td>
</tr>
<tr>
<td>Georg Wagner</td>
<td>TUG</td>
</tr>
<tr>
<td>Olamide Omolola</td>
<td>TUG</td>
</tr>
<tr>
<td>Lukas Alber</td>
<td>TUG</td>
</tr>
<tr>
<td>Sven Wagner</td>
<td>FHG</td>
</tr>
<tr>
<td>Sebastian Mödersheim</td>
<td>DTU</td>
</tr>
</tbody>
</table>

2.2 History

| Version | Date       | Author                          | Changes                                                        |
|---------|------------|---------------------------------|                                                               |
| 0.1     | 11/02/2019 | Stefan More                     | Initial Version                                               |
| 1.0     | 12/02/2019 | Stefan More, Lukas Alber, Peter Lipp | Additional Information from internal feedback                  |
| 1.1     | 14/02/2019 | Stefan More                     | Extended document based on reviewer feedback                  |
| 1.2     | 14/02/2019 | Olamide Omolola                 | Added API specification and additional information.           |
| 1.3     | 15/02/2019 | Olamide Omolola                 | Implemented reviewer feedback                                 |
3. Table of Contents

1. Executive Summary 2
2. Document Information 3
   2.1 Contributors ................................................................................................................... 3
   2.2 History ........................................................................................................................... 3
3. Table of Contents 4
4. Table of Acronyms 5
5. Introduction 6
6. Components 7
   6.1 Technical Infrastructure ................................................................................................ 7
   6.2 Access to Source Code and other Artefacts ................................................................. 7
   6.2.1 Relevant Artefacts ..................................................................................................... 8
   6.3 Obtaining the Source Code for the Automated Trust Verifier .................................... 8
   6.4 Obtaining the Source Code for the Trust Policy Interpreter ....................................... 8
   6.5 Getting the right Revision ........................................................................................... 8
   6.6 Building the Source Code ........................................................................................... 8
   6.7 Running the ATV .......................................................................................................... 9
   6.8 Trust Policy Format ...................................................................................................... 9
   6.9 Electronic Transaction Format ..................................................................................... 9
   6.10 API Specification ....................................................................................................... 10
7. References 15
8. Project Description 16
4. Table of Acronyms

ATV – Automated Trust Verifier
ET – Electronic Transaction
TPL – Trust Policy Language
5. Introduction

The automated trust verifier (ATV) is the central component of LIGHTest. It takes a trust policy (TP) and an electronic transaction (ET) as input and uses other LIGHTest components to verify the ET. The LIGHTest Consortium develops a prototype implementation of the ATV component and plans to provide it to the public under an open source license.

Since the ATV is actively verifying trust policies in the TPL language, a component directly used by the ATV is the Trust Policy interpreter. This component is also mentioned in this document.

The ATV is tested systematically via unit tests. In addition, integration tests are performed in WP8 [1]. After completion, the ATV will be delivered as a library to LIGHTest pilot partners for integration into their pilots (WP9). In addition, other components access the LIGHTest ATV via a REST API. Furthermore, we provide a basic GUI for demonstration purposes.

The ATV prototype developed by the LIGHTest consortium is currently under active development. The scope of this document is therefore limited to the development version.
6. Components

6.1 Technical Infrastructure
The details of the technical infrastructure can be found in D8.1 [1]. This deliverable describes the technical infrastructure for source code hosting as well as automated deployment methods. We use Git as source code repository and Nexus to manage our software artefacts. The default operating system used for development is Ubuntu Linux.

Git [2] is a version control system for tracking changes in computer files and coordinating work on those files among multiple people. It is used for source code management in software development, but it can be used to keep track of changes in any set of files.

6.2 Access to Source Code and other Artefacts
The programming language for the ATV is JAVA 8 and Maven [3] is used to build the source code. Maven retrieves dependencies; you need to give Maven access to artefacts hosted on their respective Maven repository in the IAIK GitLab repository hosted at https://extgit.iaik.tugraz.at. This can be done by creating a personal access token in GitLab and adding it to your maven config.xml.

A access token can be created in GitLab in Profile → Settings → Access Tokens → Personal Access Tokens

Maven’s global settings.xml is located in ~/.m2/settings.xml and needs to contain the following:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0
        http://maven.apache.org/xsd/settings-1.0.0.xsd">
    <servers>
        <server>
            <id>gitlab-maven</id>
            <configuration>
                <httpHeaders>
                    <property>
                        <name>Private-Token</name>
                        <value>YOUR_TOKEN_HERE</value>
                    </property>
                </httpHeaders>
            </configuration>
        </server>
    </servers>
</settings>
```
6.2.1 Relevant Artefacts

To build the ATV, the following dependency is needed and therefore access to its repositories needs to be configured:

Component name: TrustPolicyInterpreter
GIT repository: https://extgit.iaik.tugraz.at/LIGHTest/trustpolicyinterpreter
Maven repository: https://extgit.iaik.tugraz.at/api/v4/projects/1050/packages/maven

6.3 Obtaining the Source Code for the Automated Trust Verifier

The source code of the ATV can be obtained via the following git command:

```
git clone git@extgit.iaik.tugraz.at:LIGHTest/AutomaticTrustVerifier.git
```

and receive updated versions via the following git command:

```
git pull
```

6.4 Obtaining the Source Code for the Trust Policy Interpreter

The source code of the Trust Policy Interpreter can be obtained via the following git command:

```
git clone git@extgit.iaik.tugraz.at:LIGHTest/trustpolicyinterpreter.git
```

6.5 Getting the right Revision

The development for this version takes place in the branch ‘master’. The branch can be changed via the command

```
git checkout master
```

to get the sources of that particular branch.

6.6 Building the Source Code

The project is written in version 1.8 of the Java programming language and uses several open source libraries (a detailed list is given in the project's `pom.xml`).
The project uses Maven as build system and can be built and tested using the command

```
mvn verify
```

To only build the project, use the command

```
mvn build
```

As an alternative to building the source code on the command line, you can also use your favourite IDE.

Since Maven is also taking care of dependencies, you need to setup access to several GitLab Maven repositories first to build the ATV development version with the Maven commands described above.

### 6.7 Running the ATV

Since the ATV is a library, there are multiple ways of accessing it.

For demonstration purposes, there is a simple GUI in:

```
/at/tugraz/iaik/lightest/verifier/Main.java
```

Provided the Java classpath is setup, the GUI can be started using the following command inside `target/classes`:

```
java at.tugraz.iaik.lightest.verifier.Main
```

As an easier alternative, you can also use your favourite IDE.

### 6.8 Trust Policy Format

A Trust Policy must be given to the ATV in TPL format, as specified in D2.5 [4]. The D2.5 contains numerous examples on creating Trust Policies.

### 6.9 Electronic Transaction Format

An Electronic Transaction Format describes the contents of an electronic transaction in machine-readable format. It resides within the Electronic Transaction. An example of the Electronic Transaction Format written in XML is given below:

```xml
<?xml version="1.0"?>
<Transaction>
  <Creator>
    <FirstName>John</FirstName>
    <LastName>Mark</LastName>
    <ContactNo>1234567890</ContactNo>
    <Email>johnmark@xyz.com</Email>
  </Creator>
</Transaction>
```
6.10 API Specification

The API document for the ATV is located at https://extgit.iaik.tugraz.at/LIGHTest/atv-mockup/blob/master/atv.yaml. This is based on the OpenAPI specification [5].

A live mock-up of the API document is available at http://virtserver.swaggerhub.com/EULIGHTest/atv/1.0.0/ and it was created using an open-source virtual server from Swaggerhub.

An example of a query to the live mock-up is

1. curl -X GET "http://virtserver.swaggerhub.com/EULIGHTest/atv/1.0.0/verifyInstance/" -H "accept: application/json"

The REST API specification for the ATV is given below:

1. swagger: '2.0'
2. info:
3. version: 1.0.0
4. title: Verification Instance
5. description: A sample API for verification
7. contact:
8. name: IAIK Lightest Team
9. email: lightest@iaik.tugraz.at
10. url: 'http://iaik.tugraz.at'
11. license:
12. name: MIT
13. url: 'http://opensource.org/licenses/MIT'
14. # host: iaik.tugraz.at
15. # basePath: /api/atv
16. # schemes:
17. #   - http
18. consumes:
19. - application/json
20. produces:
21. - application/json
22. paths:
23. /verifyInstance:
24. get:
25.   description: >-
26.   Returns all verification instances from the system that the user has
27.   access to
28. operationId: findInstances
29. produces:
30. - application/json
31. - application/xml
32. - text/xml
33. - text/html
34. parameters:
35. - name: tags
36.   in: query
37.   description: tags to filter by
38.   required: false
39.   type: array
40.   items:
41.     type: string
42.   collectionFormat: csv
43. - name: limit
44.   in: query
45.   description: maximum number of results to return
46.   required: false
47.   type: integer
48.   format: int32
49. responses:
50. 200:
51.   description: instance response
52.   schema:
53.     type: array
54.   items:
55.     $ref: '#/definitions/instance'
56. default:
57.   description: unexpected error
58.   schema:
59.     $ref: '#/definitions/errorModel'
60.
61. post:
62.   description: Creates a new verifyInstance in the store.
63. operationId: addInstance
64. produces:
65.  - application/json
66. parameters:
67.  - name: instance
68.      in: body
69.      description: Instance to create
70.      required: true
71.      schema:
72.          $ref: '#/definitions/newInstance'
73. responses:
74.    '200':
75.      description: instance response
76.      schema:
77.          $ref: '#/definitions/instanceVerification'
78.    default:
79.      description: unexpected error
80.      schema:
81.          $ref: '#/definitions/errorModel'
82. '/verifyInstance/{id}':
83. get:
84.      description: Returns a instance based on a single ID
85.      operationId: findInstanceById
86.      produces:
87.        - application/json
88.        - application/xml
89.        - text/xml
90.        - text/html
91. parameters:
92.        - name: id
93.        in: path
94.        description: ID of instance to fetch
95.        required: true
96.        type: integer
97.        format: int64
98. responses:
99.    '200':
100.       description: instance response
101.       schema:
102.          $ref: '#/definitions/instanceVerification'
103.    default:
104.       description: unexpected error
105.       schema:
106.          $ref: '#/definitions/errorModel'
107. delete:
108.       description: deletes a instance based on the ID supplied
109. operationId: deleteInstance
110. parameters:
111. - name: id
112.   in: path
113.   description: ID of instance to delete
114.   required: true
115.   type: integer
116.   format: int64
117. responses:
118.   '204':
119.     description: instance deleted
120.     default:
121.     description: unexpected error
122.     schema:
123.       $ref: '#/definitions/errorModel'
124. definitions:
125.   instance:
126.     type: object
127.     required:
128.       - id
129.     properties:
130.       id:
131.         type: integer
132.         format: int64
133.         example: 42
134.   newInstance:
135.     type: object
136.     required:
137.       - id
138.       - policy
139.       - transaction
140.     properties:
141.       id:
142.         type: integer
143.         format: int64
144.         example: "<id to store the result under here>
145.       policy:
146.         type: string
147.         format: binary
148.         example: "<policy in tpl here>
149.       transaction:
150.         type: string
151.         format: binary
152.         example: "<transaction zip here>
153. instanceVerification:
154. type: object
down to the point
155. required:
down to the point
156. - id
down to the point
157. - verificationResult
down to the point
158. - result
down to the point
159. properties:
down to the point
160. id:
down to the point
161. type: integer
down to the point
162. format: int64
down to the point
163. example: 42
down to the point
164. verificationResult:
down to the point
165. type: string
down to the point
166. example: "trusted, because of ..."
down to the point
167. result:
down to the point
168. type: integer
down to the point
169. format: int64
down to the point
170. example: 10
down to the point
171. errorModel:
down to the point
172. type: object
down to the point
173. required:
down to the point
174. - code
down to the point
175. - message
down to the point
176. properties:
down to the point
177. code:
down to the point
178. type: integer
down to the point
179. format: int32
down to the point
180. example: 500
down to the point
181. message:
down to the point
182. type: string
down to the point
183. example: "something went wrong"
down to the point
184. # Added by API Auto Mocking Plugin
down to the point
185. host: virtserver.swaggerhub.com
down to the point
186. basePath: /EULIGHTest/atv/1.0.0
down to the point
187. schemes:
down to the point
188. - https
down to the point
189. - http
7. References


8. 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), Ubisecure (FI) and University of Piraeus Research Center (GR). 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.