European Security in Health Data Exchange

Deliverable D5.4

Data hiding tools

Editor(s):	Micha Moffie, Muhammad Barham
Responsible Partner:	IBM
Status-Version:	Final
Date:	13/12/2017
Distribution level (CO, PU):	CO

Project Number:	GA 727301
Project Title:	SHIELD

Title of Deliverable:	Data hiding tools
Due Date of Delivery to the EC:	31/12/2017

Workpackage responsible for the Deliverable:	WP5
Editor(s):	Micha Moffie, Muhammad Barham
Contributor(s):	Micha Moffie, Muhammad Barham, Alberto Berreteaga, Xabier Larrucea
Reviewer(s):	Jason Xabier Mansell
Approved by:	All partners
Recommended/mandatory readers:	All WP's

Abstract:	We lay the ground work for the adaptation, development and integration of the data hiding tool within the SHiELD framework. We first review the most relevant state of the art and presents the main use case requirements for the hiding tool. Next, we provide a detailed summary of the existing hiding tool which will be used as baseline and on top of which we plan to continue the development. Lastly, we provide a high-level view of the expected integration of the hiding tool within the SHiELD framework.
Keyword List:	Data Hiding, Data Masking, Enforcement Point
Disclaimer	This document reflects only the author's views and neither Agency nor the Commission are responsible for any use that may be made of the information contained therein

Document Description

Document Revision History

Version	Date	Modifications Introduced	
		Modification Reason	Modified by
0.1	01/08/2017	First draft ToC version	Micha Moffie, Muhammad Barham (IBM)
0.2	02/11/2017	Initial requirements, updated ToC	Micha Moffie (IBM)
0.3	14/11/2017	Dicom SOTA	Muhammad Barham (IBM)
0.4	15/11/2017	Draft PDF SOTA	Micha Moffie (IBM)
0.5	16/11/2017	Draft integration	Micha Moffie (IBM)
0.6	19/11/2017	Data hiding tool summary, draft integration	Micha Moffie (IBM)
0.7	20/11/2017	Draft introduction, executive summary and next steps	Micha Moffie (IBM)
0.8	27/11/2017	Updated SOTA	Muhammad Barham (IBM)
0.9	12/12/2017	Updated Enforcement point description, included direct reference and Text to DICOM, additional minor issues, updated conclusion with Enforcement Point future work.	Jason Xabier Mansell (TECNALIA), Alberto Berreteaga (TECNALIA). Xabier Larrucea (TECNALIA).
1.0	13/12/2017	Final Version addressing all issues agreed with the internal reviewer.	Micha Moffie, Muhammad Barham (IBM)

Table of Contents

List of Figures

List of Tables

Terms and abbreviations

AB	Advisory Board
API	Application Programming Interface
CA	Consortium Agreement
CSS	Cascading Style Sheets
CT	Computed tomography
DICOM	Digital Imaging and Communication in Medicine
EU	European Union
EP	Enforcement Point
GDPR	General Data Protection Regulation
HTML	HyperText Markup Language
Json	JavaScript Object Notation
MRI	Magnetic resonance imaging
PDF	Portable Document Format
PII	Personally identifiable information
REST	Representational state transfer
SSN	Social Security Number
XML	eXtensible Markup Language
XPath	XML Path Language

Executive Summary

This report lays the ground work for the adaptation, development and integration of the data hiding tool within the SHiELD framework. The report reviews the most relevant state of the art and presents the main use case requirements for the hiding tool. The report provides a detailed summary of the existing hiding tool which will be used as baseline and on top of which we plan to continue the development. Lastly, we provide a high-level view of the expected integration of the hiding tool within the SHiELD framework.

1 Introduction

This deliverable is the first deliverable in Task 5.4: Data hiding tools. Its goal is to lay the ground work for the adaption, development, and integration of data hiding within Shield.

Data masking is the process by which sensitive data is replaced, possibly in a reversible manner, with data that is unintelligible to receiver. The masked data is usually sensitive data, such as personally identifiable information (PII), health information, names, addresses, and so on. The main purpose of data masking is to preserve the data owner privacy enforce the data owners consent and comply with legal regulation (such as GDPR).

Encrypting all the data, or replacing the data with tokens or fake data would be a trivial solution for designing a masking data tool. However, when the requirement is that the data must remain meaningful and real, designing data masking tool become more challenging. For example, in case of Shield, a transmitted CT image must remain meaningful, however, it must not reveal sensitive data.

2 Organization of the document

The deliverable is organized as follows: In the next section we provide a short summary of the most relevant data masking tools and provide more details about select masking methods for common health care formats. Section 4 provides a summary of the data hiding tool requirements based on the work done in both WP2 – requirements and architecture as well as WP6 – validation case studies. In section 5 we provide details about our existing data hiding tool which forms the basis of our work. Section 6 provides a short overview of the expected integration of the hiding tool (which will be defined in WP2) and section 7 provides a short summary and next steps.

3 State of the art

Data masking can be considered in the following use cases:

1. Static data masking – masking of data at rest, for example, when duplicating a data base for testing purposes.

Figure 1 - Static Data Masking

2. Dynamic data masking – masking data on the fly, when data is moved from one point to another. For example, when a web page is displayed in different locations information can be masked on a proxy based on the target location.

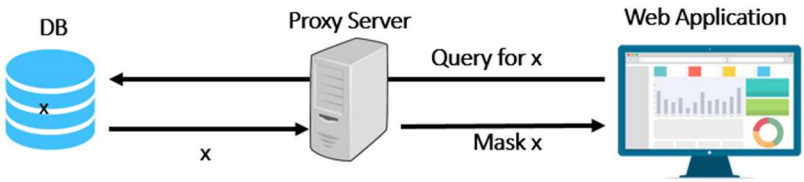


Figure 2 - Dynamic Data Masking

I this section, we review general purpose masking tools, then focus on DICOM and PDF masking tools.

3.1 General Purpose Masking tools

In this section, we review some masking tools.

1. Static Masking tools: Oracle, JumbleDB, Impreva, DataMasker, DataVeil, hushush, solix, InfoSphere [1] [2] [3] [4] [5] [6] [7] [8]: all of these tools reside on top of data bases to mask sensitive data (static data masking), and all of them are policy driven.
2. Dynamic Masking tools:
   - Microsoft Data masking [9]: a dynamic data masking tool, the tool resides on the top of data bases, and it masks the queries results based on configurations and policies.
   - Informatica Data Masking [10]: The same as Microsoft's Data Masking, Informatica data masking resides on top on data bases, and mask the query results based on configurations and policies.
   - HPE SecureData [11]: HPE SecureData is a component that provides end-to-end data protection and masking, the tool contains format preserving encryption, Stateless tokenization and Stateless key management.
   - Compuware Test Data Privacy [12]: from our understanding, the tool also resides on top of data bases. The tool provide encryption, FPE capabilities. Users can write they policies in Java-like Syntax for the tool configurations.
   - IRI Data Masking [13]: The tool can identify predefined formats, such as: PII's, PHI's (protected health information), PAN's (primary account numbers), and other sensitive information. The tool is a Data base firewall, meaning resides on top of data bases.
   - dataVantage masking tool [14]: the tool provides on-the-fly masking, the tool resides of top of data bases and file systems (DB2, VSAM, and IMS). The tool can be configured by using policies.
   - Delphix Data Masking [15]: Delphix dynamic data masking tool can scan and detect sensitive data in data sources (such as data bases) and replace the sensitive data with irreversible values. The tool scans the data and the meta data to identify the sensitive data. The tool is a policy driven.

To conclude, there are many data masking tools, most of them are static data masking tools.

3.2 DICOM & Images

Digital Imaging and Communications in Medicine (DICOM) is an international standard related to the storing and transmitting of medical images (i.e., CT, MRI, and more) and other related digital data. The DICOM standard specifies formats and the protocol to be used to storage of

digital images and other digital data. In order to comply with the DICOM Standard description bellow is the extract of section “1.1 Scope of DICOM” of the DICOM Standard [16]

1.1 Scope of DICOM

Digital Imaging and Communications in Medicine (DICOM) is the standard for the communication and management of medical imaging information and related data.

The DICOM Standard facilitates interoperability of medical imaging equipment by specifying:

• For network communications, a set of protocols to be followed by devices claiming conformance to the Standard.

• The syntax and semantics of Commands and associated information that can be exchanged using these protocols.

• For media communication, a set of media storage services to be followed by devices claiming conformance to the Standard, as well as a File Format and a medical directory structure to facilitate access to the images and related information stored on interchange media.

• Information that must be supplied with an implementation for which conformance to the Standard is claimed.

The DICOM Standard does not specify:

• The implementation details of any features of the Standard on a device claiming conformance.

• The overall set of features and functions to be expected from a system implemented by integrating a group of devices each claiming

DICOM conformance.

• A testing/validation procedure to assess an implementation's conformance to the Standard.

There are many DICOM anonymizers (or de-identifiers). The following list of tools provide support de-identification of DICOM:

- PixelMed DicomCleaner [17]: is a free open source tool with user interface for importing and de-identifying (cleaning) DICOM instances. It can remove headers (meta-data). It also, can blackout burned in annotations in pixel data.
- DICAT [18]: is a graphical tool that facilitates DICOM de-identification directly on local workstation. The tool can only de-identify headers (meta-data).
- AG Mednet DICOM De-Identification [19]: the tool enables to eliminate patient-identifying information (headers and meta-data). The tool also provides pixel de-identification
- Yakami, Tudor, GDCM, DVT DICOM Tools [20] [21] [22] [23]: these tools can only mask meta-data (headers).
- RSNA [24]: can anonymize meta-data, in addition to burn pixel data (mask images).

To conclude, all the DICOM masking tools can mask either meta-data and/or burn pixel data. And all of them, are static, meaning, none of them can mask on-the-fly.

3.3 PDF

Portable Document Format (PDF) is a file format used to present documents in a manner independent of application software, hardware and operating systems. PDF captures all the elements of a printed document as an electronic image that you can review, navigate, print or forwards to someone else.

Existing state of the art redaction tools are able to support redaction of pdf elements. The following list identifies the ones which are relevant for the analysis:

- Adobe pro – Redaction tool [25]: The tool supports the redaction of hidden information such as meta data. In addition, this tool allows to:
  - Highlight text to redact,
  - Redact an entire page,
  - Search and redact instances of words, phrases or patterns.
- Iskysoft [26]: The tool allows the user to redact pdfs by selecting items or searching for text.
- Similarly, pdfexperts [27]: The tool allows the user to redact pdfs by selecting items or searching for words.
- Similarly, pdfelement [28]: The tool allows the user to redact by selection or searching.
- Appligent [29]: The tool provides supports for advances pattern matching redaction. It supports built-in search for credit cards, email, dates, postal codes, ssn, telephone numbers and URL.

It is important to note that advanced tools do support masking of documents based on templates. In this case, the selection of the sensitive data is done by identifying the area in which they are displayed on the screen. There is not enough information of the internals of these tools available to know whether the information is removed from the file or simply hidden.

The following two tools have been identified as the most relevant for the purpose of our study, since they are able to mask document based on templates:

- Evermap [30]: This tool supports multiple way for redacting documents:
  - Manually selecting areas for redaction,
  - Redaction of text using dictionaries,
  - Redaction of text using patterns,
  - Redaction of text of predefined patters, (e.g. SSN, phone, employer id number, credit card, email etc.)
  - Redaction based on text fonts / styles,
  - Redaction of all text/images.

  The most relevant abilities are:

  - Offset redaction, which allows specifying a matching text (anchor) and the offset from it – which Is to be redacted.
  - Template redaction – use templates to mark-up information that Is always in the same location and can be redacted in identical forms.

- Rapid redact [31]: This tool has similarly support for redaction templates to support automatic redaction of predefined areas.

4 Use case requirements

In this section we summarize the main requirements related to data analysis. Much of this work is based on Deliverable D6.1 - use case specification [32], Deliverable D2.3 - SHiELD Architecture [33] and discussions with the use case providers.

The main responsibility of the data hiding tool is expected to be to remove personal and sensitive data on documents in transition. The SHiELD project will utilize the standard record format developed by the epSOS project. This format, xml based, includes basic and mandatory health data such as: patient identification, allergies, medical problems, medication summary and more.

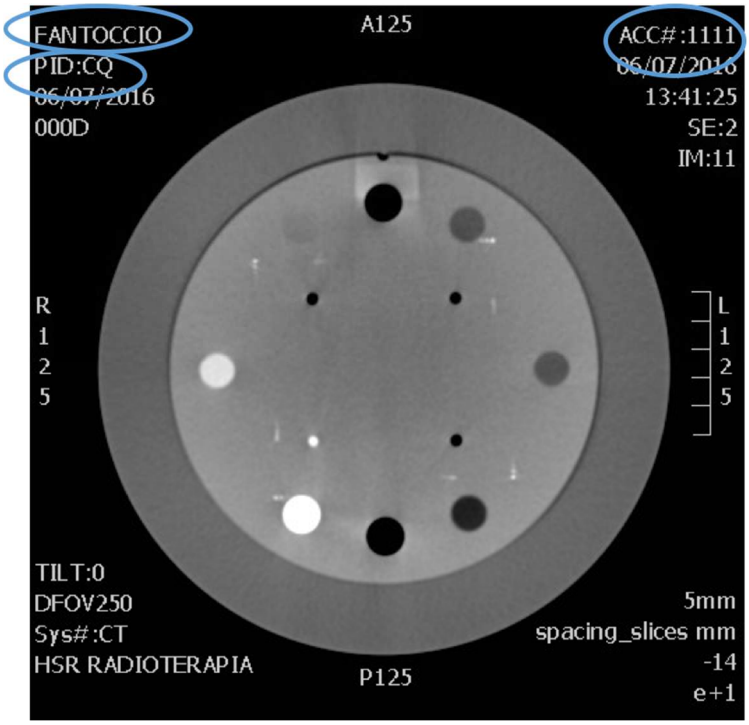
Following is an example of the epSOS standard xml containing a patient summary section and an example medication history section taken from [34]. (epsos schema can be found here [35]):

```
<ClinicalDocument xmlns='urn:hl7-org:v3'>
      <typeId extension="POCD_HD000040" root="2.16.840.1.113883.1.3"/>
      <templateId root=’1.3.6.1.4.1.12559.11.10.1.3.1.1.3'/>
      <id root=' ' extension=' '/>
      <code code='60591-5' displayName='Patient Summary'
                  codeSystem='2.16.840.1.113883.6.1'
codeSystemName='LOINC'/>
      <title>epSOS Patient Summary</title>
      <effectiveTime value='20081004012005'/>
      <confidentialityCode code='N' displayName='Normal'
                  codeSystem='2.16.840.1.113883.5.25'
codeSystemName='Confidentiality' />
      <languageCode code='en-GB'/>
      :
      <component>
            <section>
                  <templateId root='2.16.840.1.113883.10.20.1.8'/>
                  <templateId root='1.3.6.1.4.1.12559.11.10.1.3.1.2.3'/>
                  <!-- The section ID is the Prescription ID -->
                  <id root=' ' extension=' '/>
                  <code code='10160-0' displayName='History of medication
use'
                              codeSystem='2.16.840.1.113883.6.1'
codeSystemName='LOINC'/>
                  <title>Medication Summary</title>
                  <text>
                        Text as described above
                  </text>
                  <!-- Each entry is a Medication -->
                  <!-- Medication 1 -->
                  <entry>
                        :
                        <!-- Required element indicating the medication
item entry content module -->
                        <templateId
root='1.3.6.1.4.1.12559.11.10.1.3.1.3.4'/>
                        :
                  </entry>
```

```
                    <!-- Medication 2 -->
                    <entry>
                          :
                          <!-- Required element indicating the medication
item entry content module -->
                          <templateId
root='1.3.6.1.4.1.12559.11.10.1.3.1.3.4'/>
                          :
                    </entry>
              </section>
       </component>
       :
       <component>
              <structuredBody>
              :
              </structuredBody>
       </component>
</ClinicalDocument>
```

In addition, the record can be easily extended to include other types of information that may be necessary for treatment in particular circumstances. These extended types of information may be of different formats and include other types of data such as medical images (e.g. DICOM, electrocardiogram images etc.). Some of these data types may also contain structured information. (e.g. pdf containing a patient visit summary)

Following is an example of extended data, a DICOM image, taken from D6.1. The circles in the image show where sensitive information resides.



The main target requirements are summarized in the next table:

Table 1 – Data Hiding Tool Requirements

#	Requirement	
1	Hide data elements in epsos xml	Support hiding elements in different granularity (e.g. leaf element, branch)
2	Restore hidden elements (unhide)	e.g. decrypt
3	Allow the user means to specify which elements in xml to hide	Support specification of xml elements in a user-friendly way
4	Based on use case requirement, Hide data elements within some of the extended data types	e.g. DiCOM
5	Allow the user means to specify which elements in the extended data type to hide	Support a coherent, unified specification of xml elements containing extended data well as the specification of extended elements
6	Support different methods of data hiding	e.g. encryption, redaction.
7	Tool should be configuration driven	Tool should be configured to support multiple types of payloads
8	Performance	The tool should be optimized to support high rate of documents
9	The tool should be extendible to support other types of formats	Tool design should support extension
10	The tool should provide Rest API	Provide a REST API service

5 Data Hiding tool

Task 5.4 was defined to support hiding "which may be based on masking or anonymization" – depending on the requirements. It is clear, based on the requirements above, that a masking tool, able to mask specific data items in records, is needed. To this aim we will build upon a tool and service developed in the EU SUNFISH project [36], extending it as necessary based on Shield Requirements as well as the implementation required for the integration within the Shield Architecture (Open-NCP). Specifically, existing commercial tools do not support composite formats (e.g. Dicom embedded within an xml) and, in the case of pdf, are limited to rigid templates which do not allow for changes in the relative position of elements.

In this section we provide a short overview of the existing tool and service and summarize the work done in SUNFISH ID3v1 [37]. It is provided here for completeness and establishment of the basis for the implementation and integration within Task 5.4.

5.1 Overview

The process of masking seems straight forward; replace an item with '***'. This can be achieved with a simple regular expression. However, building a generic, flexible and powerful masking

engine is more complex. Consider an example where we would like to mask a name and the id, both highlighted in yellow in Figure 3.

```
<?xml version="1.0" encoding="utf-8"?>
<partial-response>
... <update ...>
        <![CDATA[
             <table ..> ...
                 <td ...">Name</td>
                 <td ...">john doe</td>
                 ..
                 <script .. type="text/javascript"> ..
                   Object.func("..Button",..,
                       {id:"12345678",widget:".."});
              .. </script>
            ..</table> ]]>
    </update> ...
</partial-response>
```

Figure 3 – Composite payload masking example

This example shows some of the difficulties: how to find the name in the text? Once found - how should we mask/unmask it (e.g. should this process be reversible)? And lastly, how do we allow the user to specify this information for different payloads?

The Data Masking library was developed to address these difficulties in a generic and flexible manner. Specifically, the library supports the following:

1. Support a wide array of mechanisms to identify and select data elements within:
    a. Structured documents.
    b. Unstructured documents.
    c. Composite documents.
2. Provide a wide array of masking/unmasking operations (e.g. redact, tokenize, encrypt).
3. Enable the modification (rewrite) structured, unstructured and composite documents while keeping their structure.
4. Provide the user with a policy allowing her to specify which data elements to select and what operation to perform on each of those elements.
5. Support conditional processing for greater flexibility.

5.2 Data Masking library design concepts

One of the main challenges addressed by the design is the ability to support multiple formats (structured, unstructured and composite) in a flexible and reusable way. This has guided the design and resulted in the following core concepts:

- Processor – an element responsible for processing a specific type or kind of payload with a predefined functionality. A Processor can be responsible to parse a Json payload, or encrypt text.
- Selector – an element that provides the means to specify how to select a certain data element. A selector can contain a XPath identifying an element in XML or can contain a regular expression.
- Predicate – a function returning a Boolean value which indicates whether a processing should be performed.

Based on these concepts, the masking/unmasking 'engines' were created. These engines handle (composite) payloads by processing the payloads in discrete steps. Essentially creating a flow of data where each step handles one aspect of the payload and postpones the rest of the processing for the next steps. The data flow in the design is implemented as a directed acyclic graph (a graph with directed edges and no cycles). The graph elements are mapped to design entities as follows:

- Node – a node in the graph corresponding to a single processor.
- Edge – a directed edge in the graph between two nodes that corresponds to a Selector and Predicate.
- Graph – a graph corresponds to an 'engine' - a collection of nodes and edges which collectively are able to perform the masking/unmasking operation on a payload.

To explain the design, we present an example where the user would like to mask the name in the payload depicted in Figure 3. The steps to be taken are the following:

1. Parse XML, select XPath (e.g. "/partial-response/changes/update")
2. Parse Html, select CSS (e.g. "table > tbody > tr > td:nth-child(2)")
3. Mask content
4. Rebuild HTML with updated text
5. Rebuilt XML with updated element

These steps are realized using the following processors and selectors

1. An XML processor that parses XML payloads, support XPath Syntax expressions to select XML nodes and support updating of selected nodes.
2. A HMTL processor that parses HTML payload, support CSS Syntax and selectors as well as support updating of selected elements.
3. A Masking Processor which is able to mask text, e.g. replace text with '*',
4. An XPath selector that specifies the relevant XPath, and
5. A CSS selector that specifies the relevant CSS path.

A graph connecting the described processors and selector is shown in Figure 4

Figure 4 – Masking engine instance example

The design of the masking library contains two main components. The first component is a masking engine which is responsible to process the payload and execute the data flow. The second component – Policy Engine – is responsible for instantiating the graph discussed above, creating a masking engine, based on a masking policy provided by the user.

5.3 Masking Policy

A masking policy is the logical encapsulation of the *answers* to these questions: (1) **where** is the data (that is considered to be sensitive or confidential?) and (2) **what** to do with the data? (should it be removed, modified or transformed?).

Where?

Sensitive data items may be detected by their structure and/or location in the document:

1. **Content based** detection is done by means of a construct called a data classifier. The data classifier can contain either (1) a regular expression or group of regular expressions, (2) a text or group of text elements to match against by simple identity, or (3) a piece of code which can describe the exact constraints that a data item needs to contain[1].
2. **Context based** detection is performed when it is possible to identify a data item based on its' relative location in the payload/document. For example, this might mean an XPath in an XML document, or more visually, a column in a table as seen it an application screen. As discussed above, payloads may be complex and contain composite formats. Detecting items in composite formats can be achieved by using multiple layers of context or content based detection.

What?

Once the data items have been identified and selected some action needs to be done. In particular, there are three basic types of masking transformations:

1. **Redaction** is a method where the data item is removed or transformed into a constant expression, e.g., '*****'. Redaction is non-reversible masking method where the original value cannot be derived from the masked value.
2. **Encryption** is a reversible masking method that uses a key as a basis for applying a transformation on the text (or plaintext) such that the result (or ciphertext) does not reveal information on the source text. The reverse process, i.e., decryption, is a computationally hard task if you do not have the key and an easy task for those who do.
3. **Tokenization** is a process that uses a unidirectional function to transform from original text to it matching token (e.g. hash function). Similar to encryption, it provides a one-to-one mapping between the original value and masked value. However, tokenization is not reversible by itself (though a mapping between original and tokenized data may be stored if unmaksing is needed)

As explained above, a masking policy is answering the **where** and **what** questions. The policy builds upon the basic design concepts and enables the user to answer the where and what questions by specifying the data flow (using a graph of nodes and edges), the processors, selectors and predicates.

The format of the masking policy is JSON. All elements (processors, selectors and predicates) have 4 attributes: id, type, configuration and loggable. (Note: configuration is type specific and is JSON itself).

Consider for example the following payload:

[1] Code based data classifiers are only partially implemented

```
{
  "phones" : {
    "home" : "(855) 003-8973",
    "work" : "(456) 194-3754",
    "mobile" : "(395) 383-3875"
  },
  "email" : "my.example@fake.com"
}
```

Assume the user would like to mask the home phone using standard encryption. In this case, the masked payload should look like this:

```
{
  "phones" : {
    "home" : "{#eraUTMB9x3xLJqP4+yMk9Q==#}",
    "work" : "(456) 194-3754",
    "mobile" : "(395) 383-3875"
  },
  "email" : "my.example@fake.com"
}
```

Now, let us assume that the user would like to be able to mask and unmask the payload. The user is able to use predicates to construct a unified policy to support both masking and unmasking. The graph representation of such a policy is show in Figure 5. Note that the json processor and jpath selectors are used to identify the home phone. Then, based on the predicates the value is encrypted or decrypted.

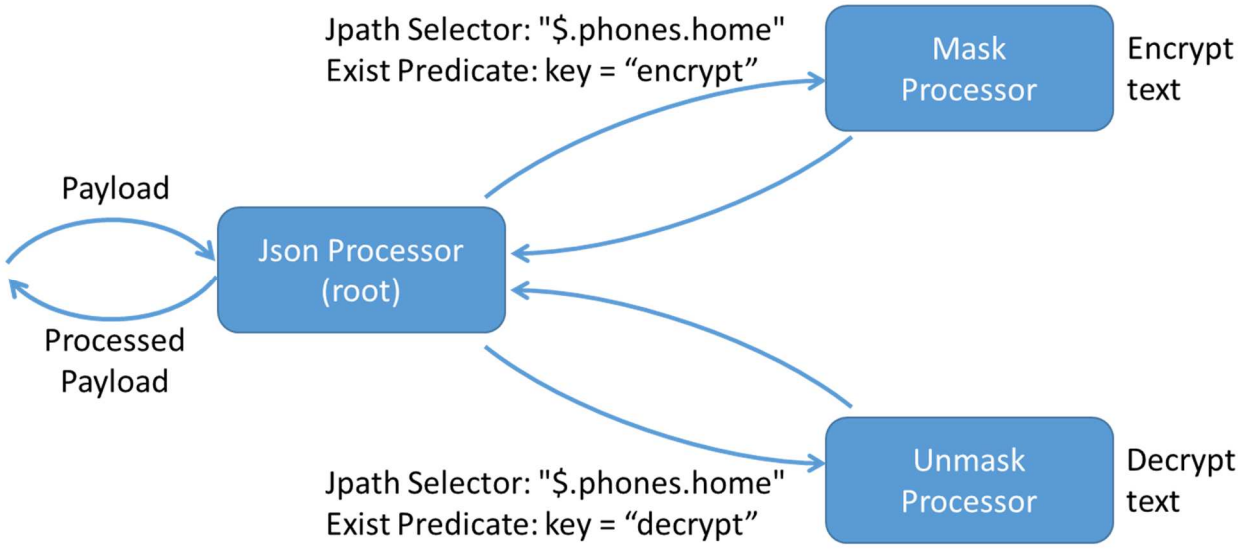


Figure 5 – Graph representation of the policy example

Figure 6 shows the complete policy for masking and unmasking the example payload. The policy is simple to follow: the data flow is described using a graph. Each node or edge refers to an element id (processor, selector, predicate) which is elaborated in the relevant sections.

```
{
	"graph": {
		"nodes": [
			{
				"id": "json-processor",
				"root": true
			},
			{
				"id": "mask-processor"
			},
			{
				"id": "unmask-processor"
			}
		],
		"edges" : [
			{
				"source": "json-processor",
				"target": "mask-processor",
				"selector": "homephone-selector",
				"predicate": "encryptionPredicate"
			},
			{
				"source": "json-processor",
				"target": "unmask-processor",
				"selector": "homephone-selector",
				"predicate": "decryptionPredicate"
			}
		]
	},
	"processors": [
		{
			"id": "json-processor",
			"type": "JsonProcessor"
		},
		{
```

```
                    "id": "mask-processor",
                    "type": "EncryptionProcessor",
                    "configuration" : { "mode" : "encrypt", "keysize" : 16 }
              },
              {
                    "id": "unmask-processor",
                    "type": "EncryptionProcessor",
                    "configuration" : { "mode" : "decrypt", "keysize" : 16 }
              }
       ],
       "selectors": [
              {
                    "id": "homephone-selector",
              "type": "JsonPathSelector",
              "configuration": { "jpath" : "$.phones.home"},
              "loggable": true
              }
       ],
       "predicates": [
              {
                    "id": "encryptionPredicate",
                    "type": "ExistPredicate",
                    "configuration": {"key" : "encrypt" },
                    "loggable": true
              },
              {
                    "id": "decryptionPredicate",
                    "type": "ExistPredicate",
                    "configuration": {"key" : "decrypt" },
                    "loggable": true
              }
       ]
}
```

Figure 6 – Policy definition example

5.4 Data Masking service

The Data Masking service provides a REST API on top of the Data Masking library. Although currently not all the functionality in the library can be accessed from the service, the service

does enable to mask and unmask payloads and supports the storage of encryption keys in a persistent storage – the sunfish service ledger (blockchain).

6 Integration

As described above, masking is used to prevent exposure of data and may be necessary as a result of access control and/or consent. It can be used when data is transferred from a data provider - and must be protected prior to it being introduced - to a data consumer. The masking policy defines what data is exposed to the data consumer. The reason to use such a method, as opposed to modification of the underlying data, is that decoupling the data security and privacy restrictions from the functional requirements allows for greater flexibility in adjusting to new requirements and less risk. Changing the functional code base, if at all possible, for non-functional requirements may introduce new problems and complications.

The point of intervention should be chosen such that the impact on the overall data flow is minimal as well as the overhead in both performance and management. In dynamic data masking the point where a masking policy is being enforced is the Enforcement Point. In SHiELD the Enforcement point will be implement as part of the openNCP following the architecture defined in WP2.

The Enforcement Point is still under definition and analysis within WP2 and will be in charge of managing the interchange of the Patient records as well as the specific country rules and regulations defined in order to provide the data hiding tools the patient record to be masked as well as the rules that should be executed by the masking rules.

The EP will interface with the consent management which will provide the access rights granted by the patient to the different information in the patient records. This access grant will be validated by the EP with the regulation (relevant for each country) and will account for the rules that will be collected in the "*knowledge base*" that is being defined in "D4.1.1 Privacy by design models and tools: proof of concept". In future versions of the document additional technical implementation details will be provided for the Enforcement Point."

Figure 7 shows the interactions between the Enforcement point (EP), the consent manager and the data hiding (masking) tool – all within the OpenNCP. Note that the data is transferred between the source and destination EP's and Hiding tool. It may not be necessary to transfer the data to the consent manager.

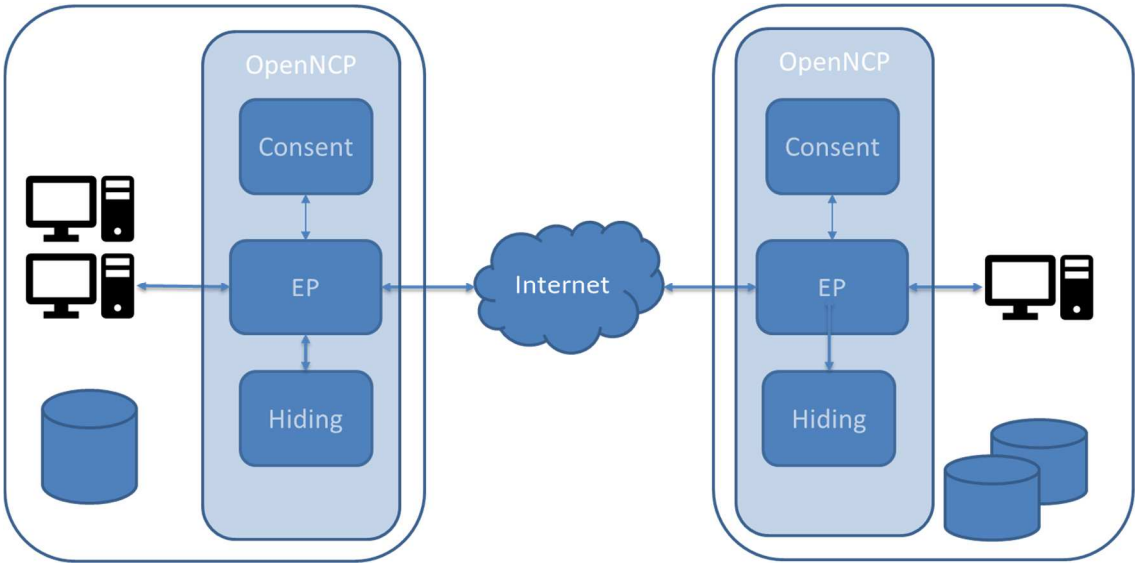


Figure 7 – Enforcement Point in OpenNCP

7 Summary and next steps

This deliverable lays the basis on which further development can proceed. The use case requirements have been clarified and the existing relevant state of the art reviewed. We have described the Data Hidding Tool proposed and provided initial details about the expected integration.

With respect to the Data Hiding Tool the future work will focus on:

1. Update the data masking service. We intend to simplify the service by removing the support for storing the context (storing encryption keys in the blockchain), enabling the user to specify the necessary keys in the API and instead, provide support for storing policies.
2. Integrate the data hiding tool (or service) within the openNCP Enforcement Point. The Enforcement Point implementation details will be defined as well as the integration with additional modules of the architecture, as for example the interaction with the rules knowledge base developed in WP4 as well as the consent Manager.
3. Prepare a masking policy able to address the main use case requirement, namely hiding select data items in the epSOS standard xml.
4. Extend our masking library and add specialized processors and selectors that can address the requirements. For example, adding a specialized DiCom processor and DiCom selector which will allow the tool to mask DiCom headers and/or images.
5. Update the policy to reflect the changes made to the library and support the new functionality. Adding a UI to support the creation of the policies will also be analysed.
6. Validate, and if necessary improve, the performance of the masking tool and library. As these operations may be on the critical path it is necessary that the library will perform masking and unmasking operations within 10s or 100s millisecond – depending on the payload size.

8 References

[1] Oracle, “Oracle Data masking,” [Online]. Available: https://www.oracle.com/database/data-masking-subsetting/index.html.

[2] orbiumsoftware, “JumbleDB,” [Online]. Available: http://www.orbiumsoftware.com/.

[3] imperva, imperva data masking, [Online]. Available: https://www.imperva.com/data-security/data-security-101/data-masking/.

[4] DataMasker. [Online]. Available: http://www.datamasker.com/.

[5] dataveil. [Online]. Available: https://www.dataveil.com/.

[6] HushHush. [Online]. Available: http://mask-me.net/.

[7] Solix, “Solix Data Masking,” [Online]. Available: https://www.solix.com/products/solix-enterprise-data-management-suite/data-masking/.

[8] I. InfoSphere, “IBM InfoSphere Data Masking,” [Online]. Available: https://www.ibm.com/il-en/marketplace/infosphere-optim-data-privacy.

[9] Microsoft, “Microsoft Data Masking,” Microsoft, [Online]. Available: https://docs.microsoft.com/en-us/sql/relational-databases/security/dynamic-data-masking.

[10] Informatica, “Informatica Data Masking,” Informatica, [Online]. Available: https://www.informatica.com/products/data-security/data-masking.html#fbid=tQXKRVGaTd7.

[11] HPE, “HPE SecureData,” HPE, [Online]. Available: https://www.voltage.com/products/data-security/hpe-securedata-enterprise/.

[12] compuware, “compuware test data privacy,” compuware, [Online]. Available: https://compuware.com/test-data-privacy/.

[13] IRI, “IRI Data Masking,” IRI, [Online]. Available: http://www.iri.com/solutions/data-masking.

[14] DataVantage, “DataVantage data masking,” DataVantage, [Online]. Available: https://datavantage.com/.

[15] Delphix, “Delphix Data Masking,” Delphix, [Online]. Available: https://www.delphix.com/platform/masking.

[16] NEMA, “dicom.nema.org,” 2017. [Online]. Available: http://dicom.nema.org/medical/dicom/current/output/pdf/part01.pdf. [Accessed 11 November 2017].

[17] PixelMid, “PixelMid DicomCleaner,” PixelMid, [Online]. Available: http://www.pixelmed.com/.

[18] S. e. al., “DICAT,” DICAT, [Online]. Available: https://github.com/aces/DICAT.

[19] A. Mednet, “AG Mednet DICOM De-Identification,” AG Mednet, [Online]. Available: http://www.agmednet.com/clinical-trials-solutions/dicom-de-identification/AG .

[20] Yakami, “Yakami DICOM Tools,” Yakami, [Online]. Available: https://idoimaging.com/programs/67.

[21] Tudor. [Online]. Available: http://santec.tudor.lu/project/dicom.

[22] GDCM. [Online]. Available: https://sourceforge.net/projects/gdcm/.

[23] DVT. [Online]. Available: http://www.dvtk.org/.

[24] RSNA. [Online]. Available: https://www.rsna.org/ctp.aspx.

[25] Adobe, “adone reduction tool,” Adobe, [Online]. Available: https://www.adobe.com/content/dam/Adobe/en/products/acrobat/pdfs/adobe-acrobat-xi-pdf-redaction-remove-visible-data-from-pdf-files-tutorial-ue.pdf.

[26] Iskysoft. [Online]. Available: https://pdf.iskysoft.com/redact-pdf/the-best-pdf-redaction-tool-for-free-download.html.

[27] pdfexperts. [Online]. Available: https://pdfexpert.com/how-to/how-to-redact-pdf.

[28] p. element. [Online]. Available: https://pdf.wondershare.net/pdfelement/ .

[29] appligent. [Online]. Available: https://appligent.com/redax-5-advanced-redaction-for-pdf-documents/.

[30] evermap. [Online]. Available: https://www.evermap.com/autoredact.asp#Batch.

[31] r. redact. [Online]. Available: http://www.rapidredact.com/rapidredact-features.html.

[32] SHiELD, “Deliverable D6.1 Use case specification and validation methodology,” 2017.

[33] SHiELD, “Deliverable D2.3 SHiELD Architecture,” 2017.

[34] EPSOS, “D3.9.1 Appendix B1/B2 epSOS Semantic Implementation Guidelines,” 2011.

[35] ART-DECOR, “an open-source tool suite that supports the creation and maintenance of HL7 template,” [Online]. Available: http://art-decor.org/decor/services/TemplateIndex?prefix=epsos-&language=de-DE. [Accessed 2017].

[36] SUNFISH, “http://www.sunfishproject.eu/,” [Online]. [Accessed 2017].

[37] Sunfish deliverable, “ID3v1 - Access control and data sharing techniques,” 2017.