

D:D-5.3 User-Centric Transparency Tools V2

Deliverable Number: D:45.3 (D:D-5.3)

Work Package: WP 45

Version: Final

Deliverable Lead Organisation: KAU

Dissemination Level: PU

Contractual Date of Delivery (release): 30/09/2015

Date of Delivery: 30/09/2015

Editor

Tobias Pulls (KAU)

Contributors

Julio Angulo (KAU), Stefan Berthold (KAU), Kaoutar Elkhiyaoui (EURC), M. Carmen Fernández Gago (UMA), Simone Fischer-Hübner (KAU), David Núñez (UMA), Tobias Pulls (KAU), Jenni Reuben (KAU), Cédric Van Rompay (EURC), Anderson Santana de Oliveira (SAP), Melek Önen (EURC)

Reviewers

Mehdi Haddad (Armines), Thomas Rübsamen (HFU)

Executive Summary

This deliverable presents the development of privacy-preserving transparency-enhancing tools in A4Cloud, with a focus on the Data Track tool. We present how the Data Track, working in conjunction with other A4Cloud tools, enable end-users (in this case also cloud/data subjects) of online services, such as cloud services, to:

- Get notified in case of policy violations at the services side and seek redress and/or remediation if applicable (Section 2). This is made possible by the integration of six different A4Cloud tools working in conjunction to empower end-users.
- Get an overview of what personal data the end-user has disclosed to online services (Sections 4 and 5). We present two different visualizations, the trace view and the timeline view, developed in several iterations with usability tests to guide the design process [AFPW15, ABFH⁺15]. The Data Track tool is generic and not only for use with A4Cloud-enabled online services.
- Request access to, correction of, or deletion of their personal data stored remotely at online services (Sections 5 and 6). This is made possible by the use of machine readable privacy-policies (in the A-PPL¹ language) and the A-PPL engine that provides granular online access to end-users.

In addition to the above contributions, we also present:

- An overview of the Transparency Log tool, advances since the previous deliverable, and its uses in A4Cloud (Section 3). The cryptographic schemes that make up the Transparency Log and a use-case have been published [PP15a, PP15b, PP15c, Pul15, RPR15], and the source code of two proof-of-concept implementations have been made available under open-source licenses.
- A summary of a literature study on how to use log analysis to detect different types of privacy violations (Section 7). The study presents a classification of policy validation techniques, investigating the differences between privacy anomaly detection and privacy misuse detection approaches.
- A plug-in for assessing policy violations and how it relates to the Data Track (Section 8). The plug-in uses an action-driven analysis to assess the relevance of policy violations to end-users. A proof-of-concept implementation is ready but final integration is not complete due to delays on other parts of the A4Cloud tool-set around policy violations.
- Research on privacy-preserving word search and how it could be used to provide strong privacy protections for an auditing tool in A4Cloud (Section 9) [EOM14].

Moving forward, the goal is to open-source the Data Track and provide functionality for importing personal data from at least one non-A4Cloud online service. Currently, we are investigating parsing Google Takeout² data.

¹The Accountable-PrimeLife Privacy Policy Language (A-PPL) [dOSoTP⁺15].

²<https://encrypted.google.com/takeout/>, accessed 2015-08-27.

Contents

1. Introduction	6
2. A4Cloud Tools and Kardio-Mon	7
3. Transparency Log	9
3.1. Advances After Deliverable D:D-5.2	10
3.2. Uses in A4Cloud	10
3.2.1. AAS and TL	10
3.2.2. A-PPLE, DT, and TL	11
3.2.3. DTMT and TL	11
4. Data Track User-Interfaces	12
4.1. Trace View	13
4.2. Timeline View	14
4.3. Data Import	17
4.4. Filtering and Searching Data	18
5. Data Track Design	20
5.1. Data Disclosure Model	20
5.2. Sources of Data Disclosures	22
5.2.1. Data Track Plug-ins	22
5.2.2. Self-Tracking	22
5.2.3. Transparency Log Messages	23
5.3. Remote Access	23
6. Data Track Implementation	25
6.1. Architecture	25
6.1.1. Database Layout	26
6.1.2. Web Server	28
6.1.3. Remote Plug-in Extensions	30
6.1.4. TL Recipient	30
6.2. API	31
6.3. Remote Access	33
6.3.1. A-PPLE API	33
6.3.2. Retrieving, Correcting, and Deleting Data	34
6.4. Parsing A-PPLE Messages	35
6.5. Importing Data Disclosures from Remote Data	37
7. Log Audit for Validating Policy Adherence	39
7.1. Privacy Misuse Detection	39
7.2. Privacy Anomaly Detection	41
7.3. Potential Use in A4Cloud	41

8. Plug-In for Assessing Policy Violations	42
8.1. Architecture	42
8.2. Relevance Assessment	42
8.2.1. Data-Driven Assessment	42
8.2.2. Action-Driven Analysis	44
8.3. Implementation of the PAPV	45
8.4. Current State of Integration	45
9. Privacy-Preserving Word Search	46
9.1. Overview	46
9.2. Performance Evaluation	48
9.3. Potential Use in A4Cloud	49
10. Summary	51
Appendix A. Mapping Graphical Icons to Personal Attributes	57

List of Figures

1.	The relevant A4Cloud tool-set for the Data Track and the data flow when an incident is sent to data subjects [Pul15].	7
2.	The prototype of the <i>trace view</i> interface of the Data Track tool.	13
3.	When hovering over an item, more information is shown inside an enhanced tooltip.	14
4.	The modal dialog showing the explicitly sent and derived data stored at the service's side.	15
5.	Sketch showing the responsive properties of the timeline view.	15
6.	The timeline view of the Data Track showing disclosure events in chronological order.	16
7.	A disclosure event with a time stamp, showing four personal data attributes.	18
8.	Some filtering and searching controls.	19
9.	A conceptual overview of the design of Data Track.	20
10.	A simple data model of data disclosures in the Data Track.	21
11.	Technical classification of solutions that validates policy adherence using log audits.	40
12.	Overview of PAPV architecture.	43
13.	Cost of the setup phase.	48
14.	Cost of the OPRF phase.	49
15.	Cost of the search phase.	50
16.	Extraction of the search response.	50

List of Tables

1.	Example of mapping between actions and relevance levels.	45
2.	The listed personal attributes are represented by corresponding icons from the Font-Awesome library and assigned a category.	57

1. Introduction

The notion of accountability is at the heart of the A4Cloud project. We want to enable organizations to be accountable for how they manage personal, sensitive and confidential information in (cloud) services. By “accountable” we mean that an organization *defines* what it does with data, that it *monitors* how it acts, *remedies* any discrepancies between what should occur and what is actually occurring, and *explains* and justifies any action it takes [DFG⁺14]. To achieve this, the first objective of the A4Cloud project is to:

develop tools that enable cloud service providers to give their users appropriate control and transparency over how their data is used, confidence that their data is handled according to their expectations and is protected in the cloud, delivering increased levels of accountability to their customers.

The goal of this deliverable is to present the second iteration of the development of privacy-preserving transparency-enhancing tools in A4Cloud, with a focus on the Data Track. We present how the Data Track, working in conjunction with other A4Cloud tools, enable end-users (in this case also data subjects) of online services, such as cloud services, to:

- Get notified in case of policy violations at the services side and seek redress and/or remediation if applicable (Section 2). This enables an organization to remediate any discrepancies.
- Get an overview of what personal data the end-user has disclosed to online services (Sections 4 and 5). This increases the *transparency* towards end-users.
- Request access to, correction of, or deletion of their personal data stored remotely at online services (Sections 5 and 6). This enables users to exercise some *control* over their personal data, in essence managing their consent to data processing.

In addition to the above points, we also present:

- An overview of the Transparency Log tool, advances since the previous deliverable, and its uses in A4Cloud (Section 3). The Transparency Log tool is used to notify end-users about, e.g., policy violations.
- A summary of a literature study on how to use log analysis to detect different types of privacy violations (Section 7). Log analysis is one conceptual way for end-users to detect policy violations on his- or her-own if a service provider is transparent.
- A plug-in for assessing policy violations and how it relates to the Data Track (Section 8). Assessments of severity can assist users in selecting which notifications to pay extra attention to.
- Research on privacy-preserving word search and how it could be used to provide strong privacy protections for an auditing tool in A4Cloud (Section 9). Cryptographic schemes like this can prevent information leaks at the service provider.

Finally, Section 10 concludes this deliverable with a summary and outlook. Next, we give a high-level overview of relevant tools and their interactions in A4Cloud.

2. A4Cloud Tools and Kardio-Mon

The so-called Wearable application is the main demonstrator in A4Cloud. The cloud provider at the center of the demonstrator is Kardio-Mon, a company that provides a branded web-service for the Wearable company. While the Wearable company is the legal entity that users of the Wearable application are interacting with, in reality the actual interactions are with Kardio-Mon. Therefore, several of the A4Cloud tools, and in particular those that are relevant for the Data Track, interact with Kardio-Mon. Figure 1 shows an overview of the relevant tools for Data Track, running both at the user's device (left side) and at Kardio-Mon in the cloud (to the right). The arrows represent the data flow when an incident, like a potential privacy-policy violation, travels from Kardio-Mon to the data subject through different tool interactions.

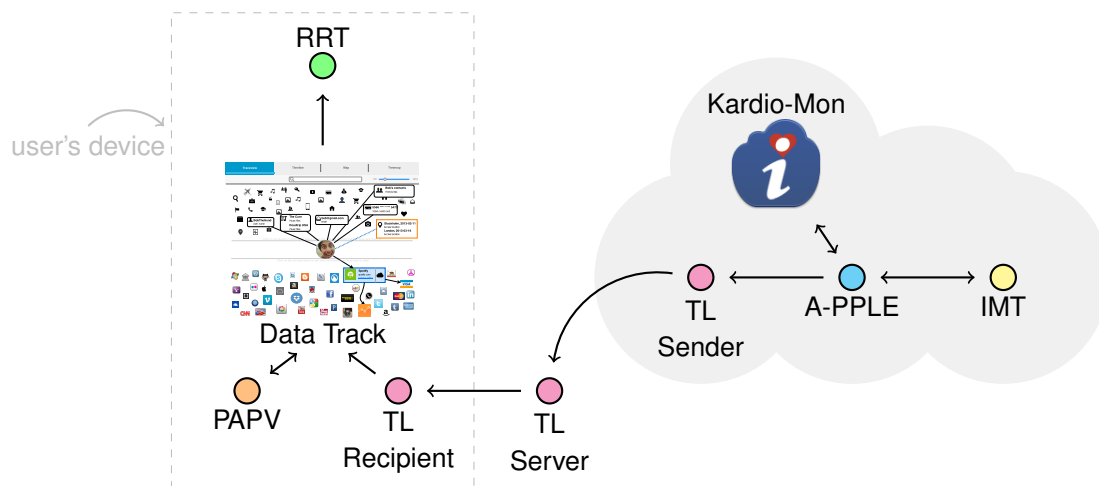


Figure 1: The relevant A4Cloud tool-set for the Data Track and the data flow when an incident is sent to data subjects [Pul15].

From right to left in Figure 1, the first tool involved in reporting an incident to a data subject is the Incident Management Tool (IMT). IMT is used by a service provider, like Kardio-Mon, to manage incidents. The inputs to the tool are either manually detected incidents (e.g., by Kardio-Mon staff) or automatically detected incidents by other tools, like the Audit Agent System developed in A4Cloud [WRR⁺15, RRR⁺15, RPR15]. The output of the tool is a human-readable incident description intended for one or more data subjects. The output of IMT is sent to the accountable PrimeLife privacy-policy language engine (A-PPLE) [dOSoTP⁺15]. A-PPLE is a tool that acts like a middleware between a database storing personal data at a service provider and the main application provided as a service, on our case the Wearable application provided by Kardio-Mon. A-PPLE will attempt to enforce the privacy policy associated with personal data stored in the database, with policy rules such as purpose-binding and obligations like retention period. Since A-PPLE knows of all data subjects in the Wearable application, it is ideally suited to forward all human-readable incident descriptions from IMT to the relevant data subjects. To do this, A-PPLE uses the Transparency Log (TL) tool. TL is described in the following section

in more detail. On the data subject's device, he or she uses the Data Track (DT) to receive incidents from the service provider. DT uses its TL Recipient (the part of TL that receives messages) to do so. Once an incident is received, DT uses the Plug-in for Assessing Policy Violations (PAPV) to assess the severity of the incident, in case it is a policy violation. Based on the severity, DT displays the notification of an incident more or less prominently in the interface for the user. Once the user wishes to address the incident, the user uses the Remediation and Redress Tool (RRT). RRT provides more information about the incident and offers, as the name suggests, remediation and redress options to the user.

3. Transparency Log

The Transparency Log (TL) is an A4Cloud tool based around the cryptographic scheme In-synd [PP15c]. TL is a tool for secure and privacy-preserving asynchronous one-way messaging, where a *sender* sends messages intended for *recipients*. TL is “secure” in the sense that it provides:

Forward-secrecy of messages Any messages sent through TL are secure (secret) from future compromise of the sender. The recipient can also be forward-secure if it discards key-material (not implemented or thoroughly evaluated).

Deletion-detection forward-integrity No events³ sent prior to sender compromise can be deleted or modified without detection. In other words, TL is “tamper evident” for all events sent prior to some attacker compromises the sender.

Publicly verifiable consistency Anyone can verify that *snapshots*⁴, that fix all data sent through TL, are consistent. This can be seen as a form of publicly verifiable deletion-detection forward-integrity.

TL is “privacy-preserving” in the sense that it provides:

Forward-unlinkability of events Any two events generated before sender compromise are unlinkable, meaning that an adversary cannot tell if they are related or not. For example, this means that an adversary cannot tell that two events are sent to the same recipient.

Forward-unlinkability of recipient identifiers Any two identifiers used to identify recipients prior to sender compromise are unlinkable. For example, this means that an adversary cannot deduce the structure of senders (in a distributed setting) by inspecting the identifiers.

TL is “asynchronous” in the sense that for a sender to send, or a recipient to receive, the other party (recipient or sender, respectively) does not have to be online. TL is also “one-way”, meaning that a recipient *cannot reply* to the sender. Finally, TL also enables a sender and recipient to produce publicly verifiable proofs of:

Sender Who was the sender that sent a particular event.

Recipient Who was the recipient of a particular event.

Message What is the message inside of an event.

Time When did a particular event exist, relative to time provided by a time-stamping authority.

Note that while the recipient of a message can always produce the above proofs, the sender is limited to deciding at the time of event generation if it wishes to keep key material to be able to produce the proofs of recipient and message. Each proof is an isolated violation of another property of TL, such as unlinkability of events or message secrecy, and will not lead to the inadvertent disclosure of, say, a long-term private key.

³An event is a container for an encrypted message and an event identifier.

⁴A snapshot fixes all data sent through TL at the time of snapshot creation. The snapshot consists of a cryptographic signature over the root of an authenticated data structure that authenticates all data sent through TL.

3.1. Advances After Deliverable D:D-5.2

Since deliverable D:D-5.2 [PM14], we have focused our efforts on the authenticated data structure at the core of TL: Balloon⁵. Balloon is a *forward-secure append-only persistent authenticated data structure*. Balloon is customized for the TL setting of a sender and recipient, where the sender is initially trusted. The sender generates events to be stored in a data structure (the Balloon) kept by an untrusted server, and recipients query this server for events intended for them based on keys and snapshots. The support for an untrusted server enables the sender to safely outsource storage. Balloon is “authenticated” in the sense that the server can prove all operations with respect to snapshots created by the sender as events are added. Balloon is also “persistent” by enabling recipients to efficiently query for events by keys for the current and past versions of the Balloon. Finally, Balloon is “append-only” and “forward-secure” in the sense that events can only be added to the Balloon, and no event added prior to the compromise of the sender can be modified or deleted without detection. Balloon has been published [PP15a, PP15b, Pul15] and its source-code made available under an open-source license on GitHub⁶.

Regarding the core Insynd scheme that makes up TL, we have improved the design and security analysis, and published the results on the Cryptology ePrint Archive [PP15c] and as part of a PhD thesis [Pul15]. A conference submission is pending at the time of writing. The source code of a proof-of-concept implementation has also been made available as open-source⁷. The implementation has a number of uses in A4Cloud, which we look at next.

3.2. Uses in A4Cloud

TL is used by four other tools in A4Cloud for different purposes, ranging from evidence storage to securely delivering notifications to data subjects.

3.2.1. AAS and TL

The Audit Agent System (AAS) is an agent-based system for auditors to automate auditing tasks in complex cloud applications and infrastructures [RR13, RPR15]. Auditors, using the AAS controller interface at a cloud provider, can task *agents* with *auditing tasks*. Agents are small pieces of software⁸ that are automatically deployed in the cloud environment, where they collect data for auditing purposes. This data is considered evidence and is stored by agents in a dedicated *evidence store*. Other agents, tasked with *processing evidence*, query the evidence store for evidence and produces a *report*. The report is returned to the auditor through the AAS controller interface.

AAS uses TL as an evidence store to protect the evidence collected by AAS agents. Auditing agents *send* evidence to the evidence store, the evidence store stores evidence, and processing

⁵“Balloon” was called an “Opaque Pile” in D:D-5.2 [PM14]. Later work used Balloon as a working name and it stuck, so now the name of the authenticated data structure is Balloon.

⁶At <https://github.com/pylls/balloon>, accessed 2015-09-14.

⁷At <http://www.cs.kau.se/pulls/insynd/>, accessed 2015-09-14.

⁸AAS is built on top of the Java Agent DEvelopment Framework (JADE), <http://jade.tilab.com>, accessed 2015-07-29.

agents *receive* evidence by querying the evidence store. This is analogous to how the TL Sender sends data, the TL Server stores data, and the TL Recipient receives data (see Figure 1). Using TL as an evidence store provides a number of benefits for AAS:

Evidence authenticity All evidence in TL are tamper-proof, i.e., any modifications are detectable thanks to the publicly verifiable consistency property of TL.

Increased reliability Since evidence cannot (undetectably) be tampered with, this increases reliability.

Completeness Once evidence is stored in TL, a recipient agent can show that it provided all evidence recorded for a particular auditing task into a report. This prevents a malicious agent from cherry-picking evidence, increasing the completeness of the evidence.

Confidentiality All evidence stored in TL is encrypted.

Improved data minimization The privacy-preserving properties of TL prevents information leakage as a consequence of using AAS, improving data minimization.

Enforcing retention time TL supports forward-secure recipients, which could be used to enforce the retention time of evidence in the evidence store by simply discarding decryption keys.

For further details, see the paper [RPR15].

3.2.2. A-PPLE, DT, and TL

The A-PPL Engine [dOSoTP⁺15] uses TL to handle the communication with cloud subjects and cloud auditors. The engine logs all relevant personal data events using the appropriate recipient identifier to an instance of the TL sender. Cloud subjects register at the instance of TL sender used by the engine with their respective recipient identifiers. This registration is automated with the Data Track. The engine uses a special recipient to log all actions to a trusted third party (the cloud auditor) who will be able to conduct investigations (for instance using the AAS tool).

3.2.3. DTMT and TL

The Data Transfer Monitoring Tool (DTMT) observes the cloud infrastructures operations to determine if the virtual resources are handled as defined in policies [dOSGJ13, dOoTPV⁺15]. For example, when the underlying infrastructure performs automated load-balancing, or when it creates backups, and then stores them in different hosts. DTMT accumulates the facts concerning the whereabouts of cloud resources, and performs correlations and inferences on them using a rule-based engine. By combining rules with pre-defined agreements between cloud customers and the cloud provider(s), DTMT can identify whether all transfers were compliant with the agreements and policies in place. In this way, potential policy violations can be detected and notified. DTMT uses TL to protect the collected evidence, given the security properties it provides. In this way, DTMT can create logs for the different tenants in a given infrastructure, and service providers cannot repudiate the logs about the alerts about transfers that are likely violating the agreements.

4. Data Track User-Interfaces

The Data Track transparency-enhancing tool allows end-users to visualize and manage their disclosures of personal data that they have made over time to various service providers. The Data Track has undergone several iterative cycles of development and UI design. In its final iteration done for the A4Cloud project the prototype for the front-end of the Data Track has been created using web-based technologies, making it possible to launch the prototype from any common browser and to make it responsive to different screen sizes. Various open source libraries and templates have been used to put together the different components of the interface, including:

Bootstrap Bootstrap provides a UI framework for front-end development, allowing for the creation of web pages in a faster, more robust and consistent manner. The Data Track utilizes Bootstrap components for navigations menus, modal (pop-up) dialogs, as well as its typography and layout of the different components.

D3.js Data Driven Documents [BOH11] is a JavaScript library that allows for easy and flexible creation of data visualizations. D3 creates an SVG element with associated properties that can be embedded within a web page, and which can be made interactive and dynamic by adding handlers to common JavaScript events. The Data Track uses D3.js to provide an interactive network-like visualizations of data disclosures consisting of tracing lines which connect disclosed personal attributes to the service providers to which these attributes have been disclosed to. This visualization is referred to as the *trace view* and it is described in Section 4.1.

qTip qTip is a jQuery plugin for displaying enhanced tooltips. It provides multiple options to customize the content of the tooltip and prettify its looks. The Data Track makes use of qTip to display additional information about elements representing service providers or personal attributes as soon as the user interacts (i.e., clicks or hovers) with these elements.

Font Awesome is an open source collection of font-based icons. Since they are vector icons they can be manipulated (i.e., resized, colored, etc.) dynamically through their styling and be treated as other common fonts. The Data Track uses the icons provided by the Font Awesome library to represent common disclosed personal attributes. In our work, we mapped some of icons provided by the Font Awesome library to the possible personal attributes that are likely to be disclosed to service providers. To do this, we suggested a preliminary ontology (see Appendix A) concerning the type of icons that may suit peoples' perceptions of personal attributes imagery. A preliminary evaluation of their perception was made and it is found in [Lin15].

CodyHouse is a library of pre-made and easily adaptable web templates composed in HTML, CSS and JavaScript. For the Data Track, we adapted two main examples: a template of a "Vertical Timeline"⁹ to present personal data disclosures in a timeline fashion (Section 4.2)

⁹<http://codyhouse.co/gem/vertical-timeline/>, accessed 2015-08-17.

and a template called “Content Filter”¹⁰ to include controls for filtering and searching through data (Section 4.4).

The design of the Data Track’s UI considers different methods for visualizing a user’s data disclosures. Based on the ideas from previous studies suggesting ways to display data disclosures [KZH12, KNP10] and the creation of meaningful visualizations for large data sets [BNG11, Fre00, KMSH12], we have designed and prototyped two main visualizations for the Data Track in A4Cloud. We refer to the two visualizations as the *trace view* and the *timeline view*. The following sections describe the design of these visualizations and other aspects of the Data Track’s UI.

4.1. Trace View

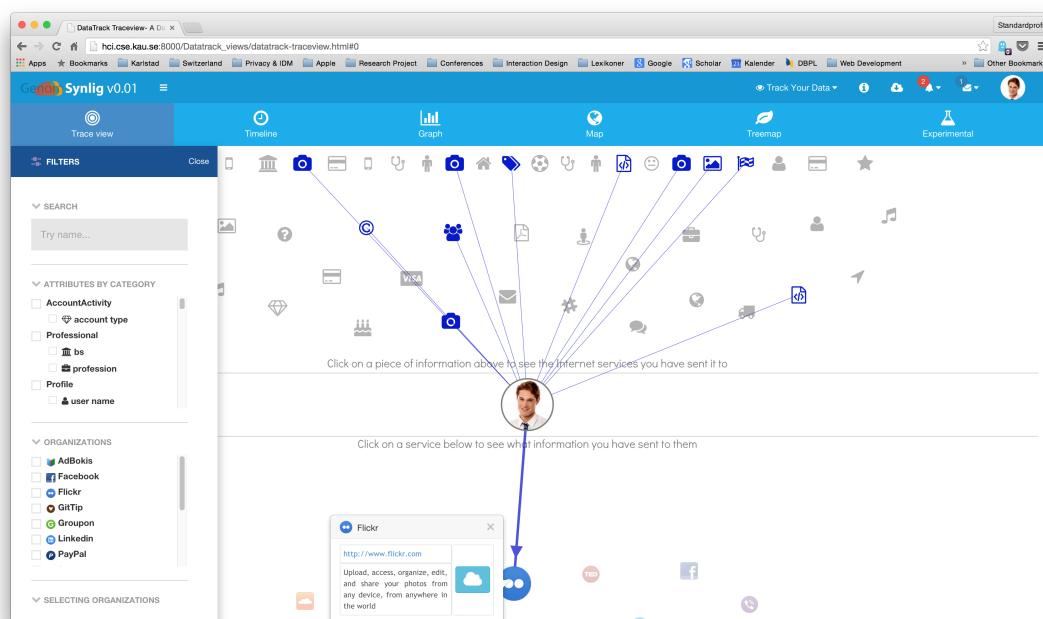


Figure 2: The prototype of the *trace view* interface of the Data Track tool.

The *trace view* visualization, shown in Figure 2, is an SVG element created using the D3.js library which is divided into three main sections or panels. The services to which the user has released personal information appear in the bottom panel and the information attributes that have been released by the user to these different services appear in the top panel. The user is represented by the panel in the middle. When the user clicks on one (or many) service(s) from the bottom panel, a *trace* is shown to the personal attributes that have been disclosed to the selected service(s). Similarly, if the user selects a personal attribute from the top panel, a trace

¹⁰<http://codyhouse.co/gem/content-filter/>, accessed 2015-08-17.

is shown to the service(s) to which the selected attribute has been disclosed at some point in time. By its design, the trace view lets users answer the question of “*what information about me have I sent to which online services?*”.

The traces and the floating elements being shown within the SVG follow the patterns of D3’s so called *forced-directed layout*¹¹, which uses different parameters to define the position and mobility of the nodes being displayed on the screen.

When the user hovers over one of the items been displayed by the trace view, a box is displayed with more detailed information about the targeted item, as seen in Figures 3a and 3b. Hovering over attributes show their type as well as the value disclosed. Hovering over a service provider logo shows the service’s contact information and a short description of the service, and a button in form of a cloud is also displayed. Clicking on this button, a dialog appears that displays information about the user which is stored at the service provider’s side, and outside the user’s control. This dialog, shown in Figure 4, does not only show the data that the user has explicitly disclosed to the service, but can also show the data stored at the service provider that has to do with inferred attributes of the user. For instance, a service provider can derive with certain probability the *religion* of the user based on a combination of other disclosed attributes.

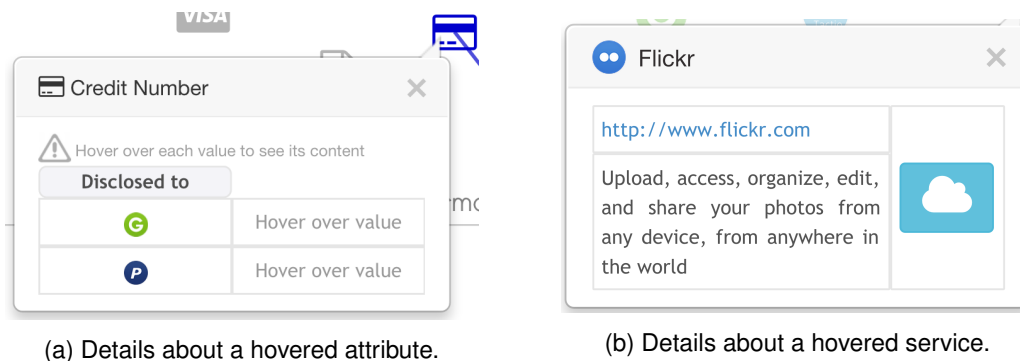


Figure 3: When hovering over an item, more information is shown inside an enhanced tooltip.

4.2. Timeline View

It was obvious for us that the design of the above mentioned trace view would not scale well for devices with smaller screens. Therefore, we created sketches and mock-ups to get an idea of the look-and-feel of the timeline elements on devices with various resolutions. An example is shown in Figure 5. In the implementation of the timeline, we adapted a freely available JavaScript library provided by CodyHouse¹² that included responsive properties of a vertically displayed timeline in its styling.

¹¹An example of force layout is given in <https://github.com/mbostock/d3/wiki/Force-Layout>, accessed 2015-08-28.

¹²<http://codyhouse.co/gem/vertical-timeline/>, accessed 2015-08-28.

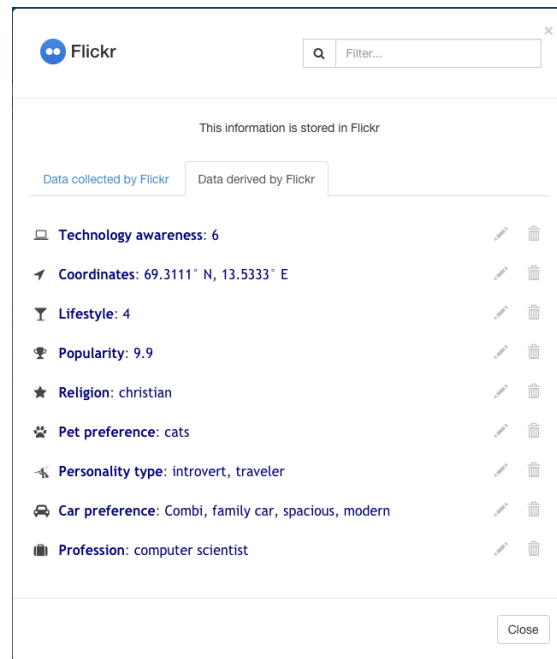


Figure 4: The modal dialog showing the explicitly sent and derived data stored at the service's side.

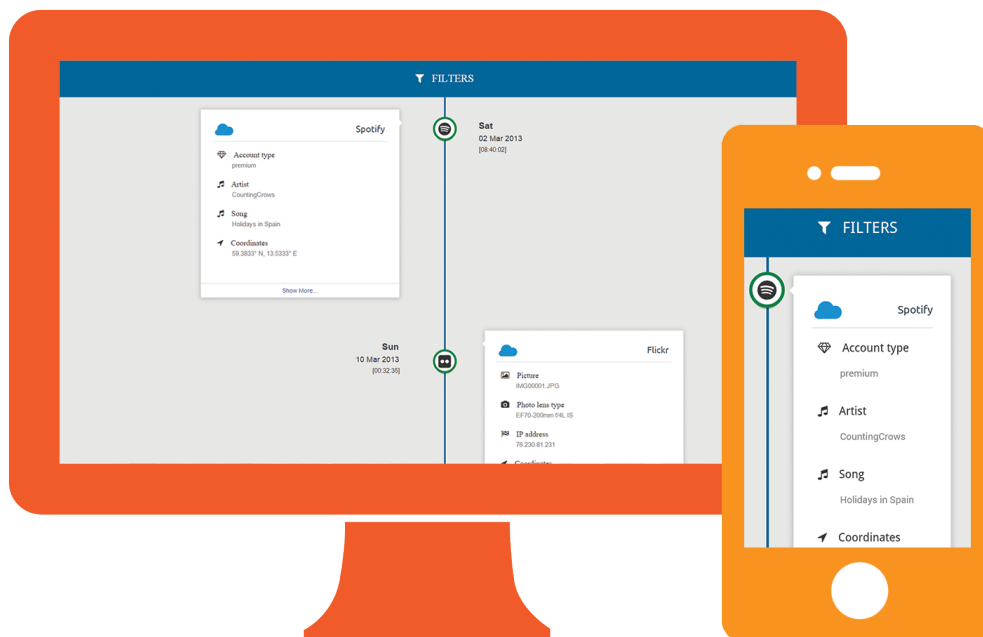


Figure 5: Sketch showing the responsive properties of the timeline view.

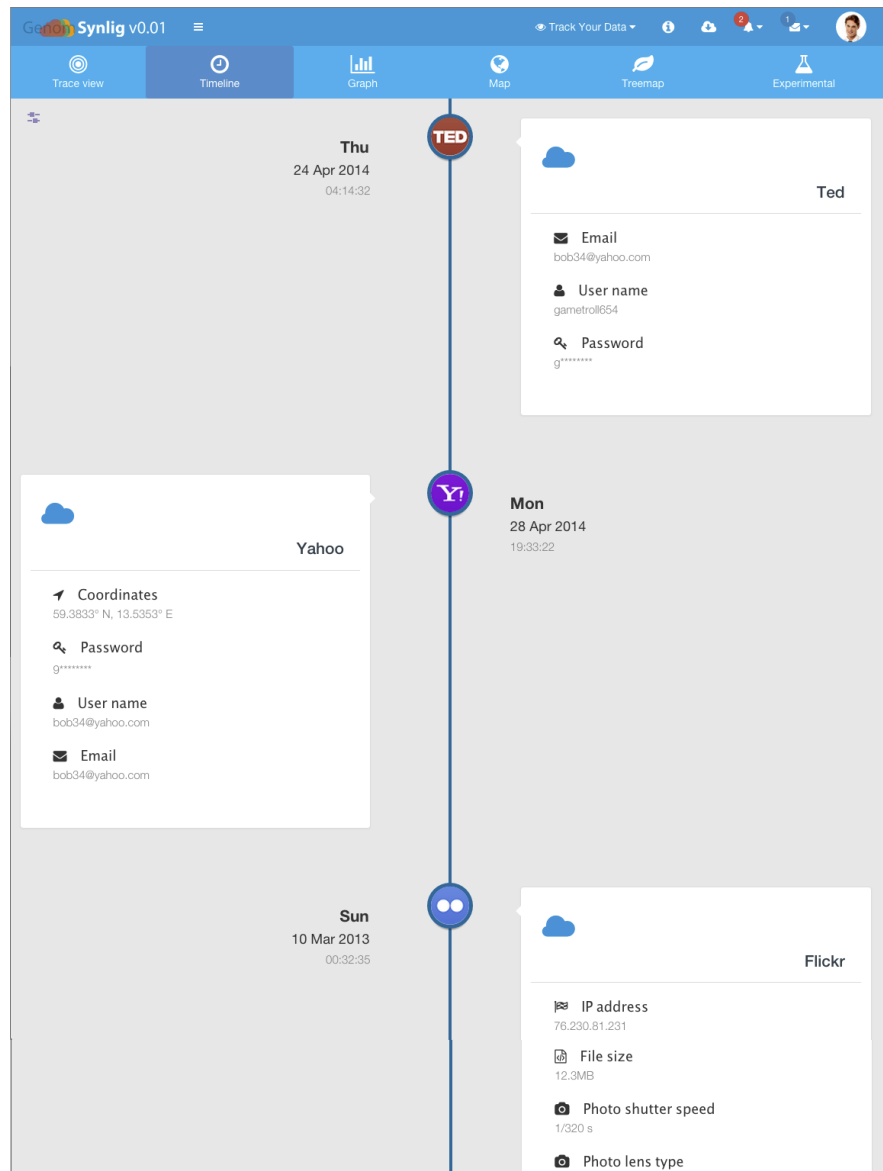


Figure 6: The timeline view of the Data Track showing disclosure events in chronological order.

The *timeline view* of the Data Track presents each disclosure event in chronological order along a vertical timeline, as shown in Figure 6. The purpose of this visualization, as opposed to the trace view, is to let users answer queries related to the time or time range in which disclosures were made, such as “*what information about me have I sent on this date?*” or “*how much information about me has been disclosed in this time period?*”, or even “*what day of the week do I tend to disclose more personal information?*”.

For the purposes of the UI of the Data Track, we refer to a *disclosure event* to an instance in which personal information is disclosed to a service provider. How the Data Track “tracks” data

disclosures is explained in Sections 5 and 6. Each disclosure event is done at a specific time, comes from a specific location, and is made using a particular device. This is information that can be conveyed to the user via a graphical interface, and employed to search for particular events according to some of these criteria.

The timeline front-end retrieves ranges of disclosure events and displays them to the user in a vertical timeline, sorted initially from the newest disclosure to the oldest. In the timeline, the logotypes of the service providers to which a disclosure was made are shown on the vertical line, the date and time in which the disclosure was made is displayed on one side of the logotype, and a so called *disclosure box* appears to the opposite side of the line [ABFH⁺ 15].

A disclosure box displays the personal attributes that were released on that disclosure event. For the sake of simplicity and cleanliness in the UI, each disclosure box only shows four of the attributes contained within the disclosure event at the beginning. By clicking on the button that reads “Show more” (see bottom of Figure 7), the box expands to show a list of all the personal attributes that were disclosed at that instance.

Inside a disclosure box, each listed personal attribute disclosed has three properties:

Type The *type* of personal attribute (e.g., creditcard number, user name, email, heart rate, etc.).

Icon A graphical representation of the personal attribute, mapped to the Font Awesome library, as explained at the beginning of this section.

Value The actual value of the disclosed attribute (e.g., bob, bob@gmail.com, 89 bpm).

The Data Track’s timeline also considers the concept of infinite scrolling, meaning that it keeps retrieving disclosures events from the Data Track’s local database as the user scroll downs the page.

To access the data on the services’ side, a cloud icon is displayed on the top corner of a disclosure box. As with the trace view, the timeline is also equipped with filtering and searching controls to allow users to look through the data and manipulate the disclosures that are being shown on the screen. We show the filtering and searching controls in Section 4.4.

4.3. Data Import

For users to perceive the value in the Data Track’s accountability and transparency properties, they need to be able to populate the Data Track with data that they have already disclosed to online services during their earlier online activities.

For this reason, we envision a mechanism in which users would be able to *fetch* or import their data from different services. In order to do this with today’s online ecosystem, the Data Track would have to implement various plug-ins which connect to some kind of API made available by service providers. The plug-ins would act as translators of the data available to be downloaded from the service providers to a format specified by the Data Track. More explanations on this approach are found in Section 5.2.1.

Through the Data Track’s user interface, users would be able to select the services from which they wish to import their data. The UI would then ask users to authenticate to this service using

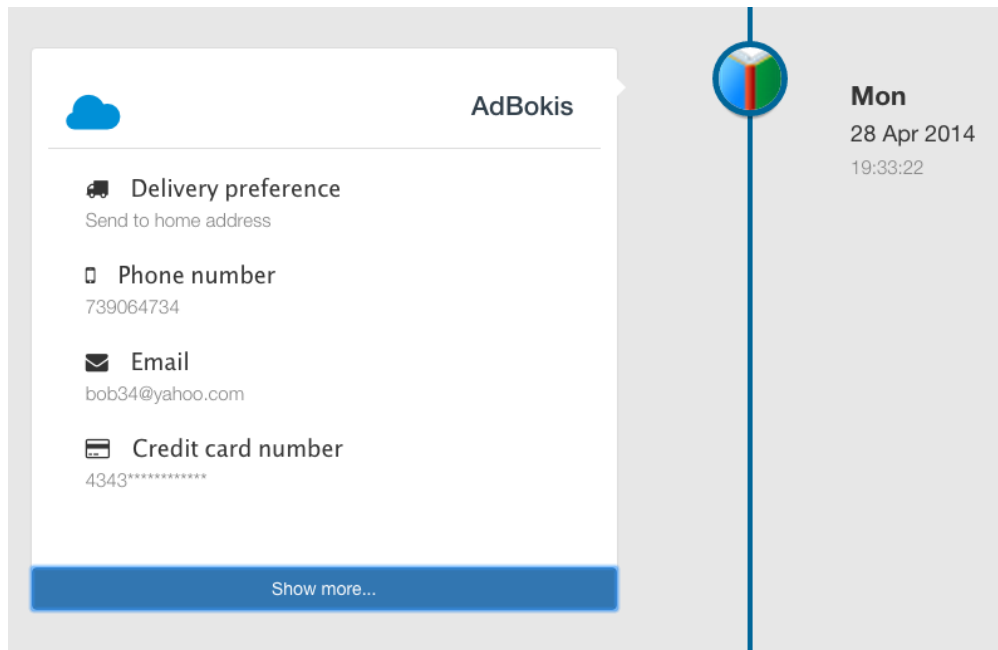



Figure 7: A disclosure event with a time stamp, showing four personal data attributes.

their credentials, and a confirmation that the import succeeded or not would be given to the user at the end. In this way, users would be connecting their Data Track tool to different online services, and they could be given options to *refresh* the data from the connected services in different ways, either triggered manually, periodically or by certain event.

4.4. Filtering and Searching Data

Presenting big amounts of information entities on a screen can be overwhelming for users. In order to cater for users perceptual capabilities and considering the screen real state, filtering mechanisms are put in place that would allow users to filter for information that is relevant to what they want to find out and make meaningful sense of their data.

Figure 8 shows an example of the trace view's filtering pane which slides open when requested by the user. In the trace view, users can search using free-text (i.e., by typing the name of a company, like Flickr or Kardio-Mon, or the name of a personal attribute, like 'credit card' or 'heart rate'), they can also select categories of data or individual pieces of data, as well as the number of entities to be displayed on the screen. Similarly, users can select only those service providers to be displayed on the screen, which might be useful when it might not be easy to find a service only by their logotype.

 **FILTERS**

Close

▼ SEARCH

Try name...

▼ ATTRIBUTES BY CATEGORY

☐ SocialMedia

☐ wallpost

☐ shared with

☒ Transportation

☒ delivery preference

☐ Media

☐ artist

▼ ORGANIZATIONS

☒ AdBokis

☐ Facebook

☒ Flickr

☒ GitTip

☐ Groupon

☐ LinkedIn

☐ PayPal

▼ SELECTING ORGANIZATIONS

Choose the way to select organizations

☒ One organization at a time

Show attributes in common

Many organization at a time

Figure 8: Some filtering and searching controls.

5. Data Track Design

The conceptual design of the Data Track has evolved over the years. For security and privacy considerations, see the following WP C-7 deliverable [FHNOP14b]. Figure 9 shows a conceptual overview of the design of Data Track, where the entire tool is running locally on a user's device primarily due to privacy and security concerns. On the top, you see the user interface (UI), as presented in Section 4. The UI is hosted by the Data Track back-end from a local web-server, and communicates with the back-end through the Data Track API. The API provides a consistent interface over all data disclosures stored in the Data Track database and remote access to data at service providers. The Data Track database receives data disclosures from three different categories of sources: plug-ins, user self-tracking, and the Transparency Log.

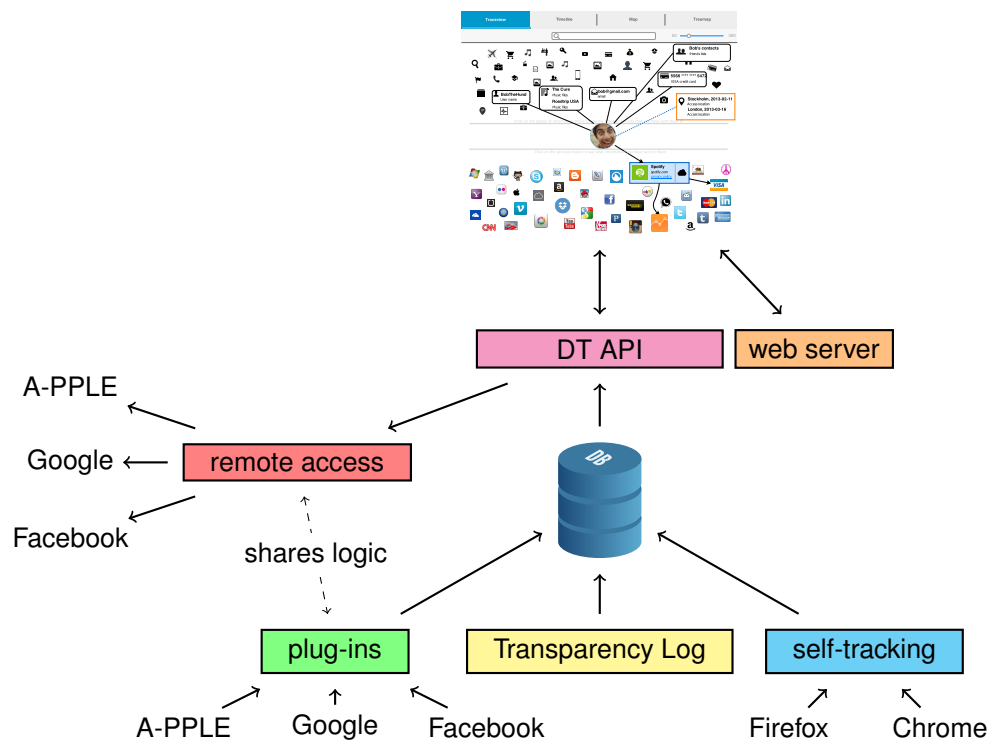


Figure 9: A conceptual overview of the design of Data Track.

We describe the Data Track API, the internal web server, and its implementation in Section 6. Next, we look at the Data Track data disclosure model, the three different sources of data disclosures, and remote access.

5.1. Data Disclosure Model

Figure 10 illustrates the data disclosure model in the Data Track. In a nutshell, it is as follows: A *disclosure* describes a data disclosure that *disclosed* a number of *attributes* to an *organization*.

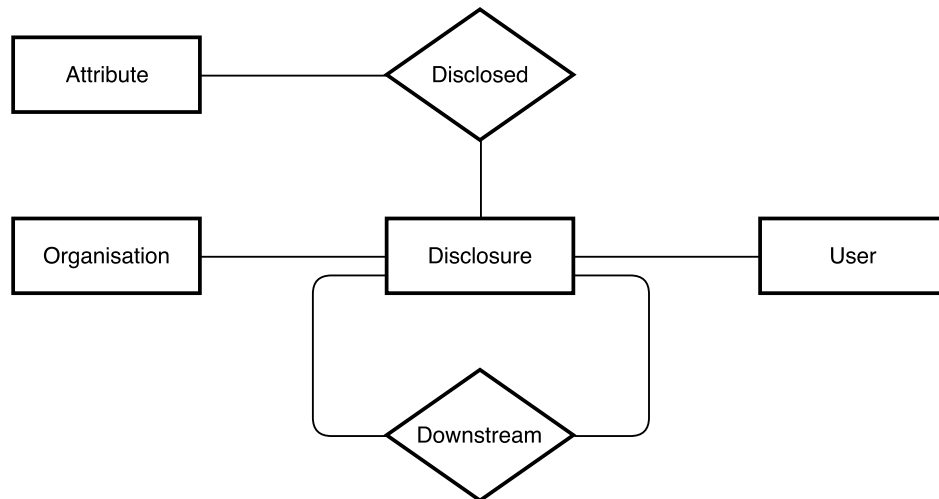


Figure 10: A simple data model of data disclosures in the Data Track.

The entity that made the disclosure is either the *user* or it is a *downstream* disclosure. By supporting downstream disclosures, we can model events like “the user disclosed the attributes address and name to the organization Amazon, and Amazon in turn disclosed the attribute address to the organization DHL”. This also models the fact that, e.g., Amazon learned of your address from you, and not from another party. We can model any directed acyclic graph of disclosures. The creation of disclosures is also the means of how we introduce new information (attributes) about users. For example, in Sweden, it is common that organizations request your credit history from a third party as part of, e.g., buying something on credit or renting an apartment. This can be modeled in the Data Track as a data disclosure from the third party credit rating agency to the organization that made the request.

In the model, we store the following information for each entity (rectangle) in Figure 10:

Attribute An identifier, the name, the type, and the value. For example, the name could be “Artist”, the type “music”, and the value “Rammstein”. The type is a categorization that could benefit from some structure, like using schema.org¹³, but currently is only used to derive the icon in the UI.

Disclosure An identifier, the sender, the recipient, a timestamp, a URL to a human-readable privacy policy, a URL to a machine-readable privacy policy, the location of the data in the disclosure, and metadata for plug-ins (described later). The sender is who sent the data in the data disclosure, normally the user, but could also be an organization. The recipient is the organization that received the disclosure.

Organization An identifier, its name, the street, zip, city, state, country, telephone, email, URL, and a description.

User A name and an optional picture.

¹³<http://schema.org/>, accessed 2015-07-31.

5.2. Sources of Data Disclosures

To populate the local database of the Data Track with data disclosures we have three different sources: plug-ins, self-tracking, and parsing TL messages.

5.2.1. Data Track Plug-ins

DT plug-ins are pieces of software tied into DT. Each plug-in provides the functionality to import all (or rather as much as is provided) personal data about the DT user stored by one specific service provider. Service providers can be, for instance, Kardio-Mon, Google, or Facebook, but currently only Kardio-Mon is fully supported by a DT plug-in. Support for other service providers is under development. A plug-in works in three steps:

1. Work with the user to obtain access to the data at the service provider. This can involve asking for the user's credentials, like a username and password, or guiding the user to perform some steps at the service provider to retrieve a file to provide to the plug-in.
2. Interpret the data from the service provider. This can involve parsing standardized formats for, e.g., email or dealing with a proprietary format only used by the service provider.
3. Based on the interpretation of the data, create one or more data disclosures in the DT database. This will most likely also involve adding the organization and several attributes.

In A4Cloud, we have developed a generic plug-in for services that run A-PPLE, described later in Section 6. We have also started investigating using the Google Takeout¹⁴ service as part of a Google plug-in for DT. At the time of writing, our conclusions are that the design of the Takeout service makes automatically retrieving the data hard. Therefore, the DT plug-in will guide the user through the steps necessary to perform a "takeout" and then enable the user to upload the resulting zip-file to DT. We are currently in the process of parsing parts of the content of the zip-file into data disclosures.

One problem with the data import approach is that it completely relies upon service providers actually sharing all data it has about a user. There are very few, if any, services that provide such transparency towards users today, even if the user exercises his or her rights until the EU Data Protection Directive 95/46/EC of data access by sending a written letter requesting all his or her personal data¹⁵.

5.2.2. Self-Tracking

Earlier versions of the Data Track [FHHW11, Pul12] collected data disclosures solely by having the user self-track him- or herself as the user disclosed data to service providers. This tracking took place either by using a middleware running locally, called the PRIME Core [SMP08], or by browser plug-ins. The main advantage of self-tracking over the plug-in based approach is that users can be assured that self-tracking will provide all data it is able to track to the user. The main

¹⁴<https://encrypted.google.com/takeout/>, accessed 2015-07-31.

¹⁵<http://www.forbes.com/sites/kashmirhill/2012/02/07/the-austrian-thorn-in-facebooks-side/>, accessed 2015-07-31.

disadvantage is that no insights can be given about what data service providers infer or learn from other sources about the user. For example, all activity that took place prior to a user self-tracking is presumably not trackable by self-tracking, but would be made available by a honest service provider through a DT plug-in. Another aspect is that of scale: self-tracking requires software capable of tracking data disclosures from several different applications, potentially on different devices, and a norm (at least for web applications) of widely different means for users to disclose data. In A4Cloud, we did not further investigate self-tracking as a way of tracking data disclosures. We note that it would fit into the Data Track design similar to plug-ins: first collect the data, then interpret the data, and finally create one or more data disclosures in the Data Track database format. To complement the DT plug-in based approach, we looked at using the Transparency Log.

5.2.3. Transparency Log Messages

One of the use-cases for the Transparency Log in A4Cloud is, as mentioned in Section 3.2.2, to transport messages from A-PPLE running at a service provider to the Data Track of the service's users. These messages may include notifying the user that the service has shared the user's personal data with another service provider, or retrieved personal data about the user from another service provider. Parsing these types of messages is the third source of reconstructing data disclosures in the Data Track. One key disadvantage of this approach are similar to those for the plug-in approach, in that it relies on the service provider to provide an accurate account of what it does and what it knows about the user. One advantage of using the transparency log is that it enables users to receive messages in *distributed* and *dynamic* settings, once a user has registered at one sender (service provider). This is a distinct advantage over using the A-PPLE plug-in, since the plug-in only retrieves data from the service provider. This does not solve how to deal with distributed and dynamic settings that potentially involve many different service providers that change over time.

In A4Cloud, at the time of writing, it is not clear how the demonstrator in A4Cloud will deal with any potential downstream service provider after Kardio-Mon, if at all. Because of this, at the time of writing, the Data Track does not parse any meaningful downstream service provider information from the Transparency Log.

5.3. Remote Access

With the Data Track UI a user can explore different visualizations of data disclosures. From the UI, the user can also attempt to exercise some control over his or her data by attempting to remotely access the data at the service provider. The DT API exports the following five operations:

RetrieveAll Retrieve all personal data stored for the user at the service provider.

Correct Change one attribute at the service provider to the provided value.

Delete Delete one attribute at the service provider.

DeleteAll Delete all personal data for the user at the service provider.

GetPolicy Get the human- and machine-readable privacy policy associated to the user at the service provider.

The remote access functionality is then, just like the DT plug-ins, custom for each service provider with varying levels of support. For example, a remote access plug-in could support correcting and retrieving data, but not deleting, since the corresponding service provider does not support it. DT plug-ins and remote-access plug-ins share a lot of logic, but act at different points in time: the DT plug-ins import data, creating one or more data disclosures, while the remote access plug-ins are used *after import* by the user from the Data Track UI. In A4Cloud, we designed a generic plug-in for remote access that works for any A-PPLE enabled service.

6. Data Track Implementation

This section describes the implementation details of DT.

6.1. Architecture

The Data Track is a (local) web service that provides the web-based user interface, a database that stores all ‘tracked’ personal data of the user, and a number of plug-in extensions for importing the user’s personal data from external sources, such as third-party web services or the TL. All the functionality is compiled and linked in one executable file so that additional software dependencies, such as specific database management systems or web frameworks, can be avoided. The Data Track is programmed and tested in Go¹⁶ and the following libraries have been used:

github.com/kelseyhightower/envconfig provides parsers for the operating system environment. The Data Track uses variables in the operating system for finding the database file and switching features on or off.

insynd provides the TL functionality. It is also developed as part of the A4Cloud project. For the Data Track, TL is a data source.

github.com/mxk/go-sqlite provides the database management functionality. Each database is stored in a file. For the Data Track, the database is the only place where persistent data is stored. Persistent data is the user’s personal data as well as setup and configuration data for the Data Track.

github.com/codahale/blake2 provides cryptographic hash functions. The Data Track uses these hash functions to create unique primary keys in the database tables.

github.com/zenazn/goji provides the web server framework. The Goji framework makes it easy to connect a web service (REST) API with program logic. The Data Track defines all API calls via Goji.

github.com/rs/cors provides Goji middleware that manages policies for cross-origin resource sharing (CORS). The Data Track uses CORS to enable a website, like that provided by Kardio-Mon, to trigger an import of a data disclosure through the Data Track API running locally on a user’s device.

github.com/parnurzeal/gorequest extends the Go HTTP client framework with multiple HTTP verbs, such as PUT, DELETE, etc. The Data Track uses the HTTP client when requesting, modifying, and deleting web resources, for instance, the user’s personal data on a third-party web services.

¹⁶<http://golang.org/>, accessed 2015-08-26.

6.1.1. Database Layout

The database is implemented using the SQLite library¹⁷. It uses a database file, by default `datatrack.db`, to store the data. Name and path of the database file can be changed by setting the operating system's environment variable `DATATRACK_DATABASEPATH`. The file and the database structure are created from scratch, if the file does not exist when the Data Track is started. If the file exists, it is assumed to be a Data Track database, in particular no effort is done to detect or deal with corrupted database files. The database contains nine tables:

user contains the DT user's name and picture. This table should not contain more than one record. The last record is used, if there are many. The following attributes exist:

name (primary key) The username.

picture The path to the user picture.

organization contains identification and contact data for organizations that store and/or process data of the user. The following attributes exist:

id (primary key) A unique ID or the short name of the organization.

name The organization's legal name.

street The organization's street.

zip The organization's zip code.

city The organization's city.

state The organization's state.

country The organization's country.

telephone The organization's customer service phone number.

email The organization's customer service mail address.

url The organization's web site.

description A description of the organization.

attribute contains the personal data of the user. The following attributes exist:

id (primary key) A unique ID.

name The attribute name.

type The attribute type, used to determine the icon in the user interface.

value The attribute value.

disclosedattribute assigns attributes to disclosures. The following attributes exist:

disclosure (primary key) The ID of the disclosure.

attribute (primary key) The ID of the attribute.

¹⁷<https://sqlite.org/>, accessed 2015-08-26.

disclosure contains the meta data of events in which the DT user disclosed personal data to an organization. The table also contains the meta data of events in which organizations disclosed the user's personal data to subcontractors (*downstream* disclosures). The following attributes exist:

id (primary key) A unique ID.

sender Either the username (in case of a upstream disclosure) or the ID of an organization (in case of a downstream disclosure or a derived disclosure)¹⁸.

recipient The ID of an organization.

timestamp The time of the disclosure.

policyhuman The data handling policy in human-readable form.

polycymachine The data handling policy in machine-readable form.

datalocation The (geographic) location of the user's personal data.

api Credential information for remote access (and import).

downtream assigns disclosures and downstream disclosures. The following attributes exist:

upper (primary key) The ID of the upstream disclosure.

lower (primary key) The ID of the downstream disclosure.

incident stores incident data and a severity level. The following attributes exist:

id (primary key) A unique ID.

notification The notification.

proof Proof material.

severity The severity level.

disclosureid The ID of the disclosure which is affected by the incident.

insynd stores the crypto data for TL recipient part of the Data Track. The following attributes exist:

name (primary key) The ID of the key.

key Key material.

insyndevent stores events that are received via TL and that are not handled in a particular way. The following attributes exist:

key (primary key) The ID of a key.

eventcount The number of already read events.

¹⁸Upstream disclosures are disclosed by the user to an organization. A derived disclosure is created whenever the organization uses the data to derive new conclusions from the disclosed data. A downstream disclosure is created whenever the organization forwards that user's data to a subcontractor.

All low-level interactions with the database are encapsulated in one separate light-weight thread of the Data Track program, since SQLite is not thread-safe. Moreover, DT's database module is logically separated from the rest of the program so that SQLite can be replaced by any other database library which may suit better in future releases. The logical separation is provided by abstractions of the SQL `INSERT` and `SELECT` statements. Outside the database module, `INSERT` or `SELECT` statements are created by means of two domain-specific languages (DSLs). The words of these DSLs, i. e., the statements are rendered to strings in the database module right before SQLite processes them. As a result, only the DSL rendering would have to be adapted to an alternative database system (other than SQLite) without the need to touch any other module of the Data Track.

6.1.2. Web Server

The web server of the Data Track is implemented using the Goji library. It consists of different handlers that fall in six groups, (1) handlers for static content; (2) handlers for dynamic content that depends on data in the database; (3) handlers that retrieve data from external data sources without modifying the database; (4) handlers that retrieve data from external data sources and add records to the database; (5) handlers that query the TL client; and (6) handlers that exist to support debugging and the development of the Data Track.

The first group of handlers manages the static files such as HTML, CSS, JavaScript, and font files for the main user interfaces as described in Section 4 and the yet unused incident management. These handlers are registered to Goji in the file

- `server/main.go`.

The other handlers are registered together with API information in a common framework. This framework is defined in the file

- `src/datatrack/handler/core.go`.

The second group of handlers translates web requests to database queries. For each of the database tables *user*, *disclosure*, *attribute*, *organization*, *incident*, and *category* there are two handlers, one for enumerating IDs of the database records, the other one for extracting the data stored in one record for a given ID. The representation of the ID enumerations can be modified in different ways: they can be (1) printed in chronological and reverse chronological order where applicable, (2) filtered, for instance up- or downstream disclosures or attributes belonging to up- or downstream disclosures, and (3) limited to a number of n IDs at a time. Moreover, the number of IDs can be counted and just the count is printed. Combinations are possible, for instance the count of all downstream disclosures. The modifiers and their combinations are taken care of in the ID enumeration handler in combination with the database DSL for SQL `INSERT` statements. More precisely, most ID enumeration handlers are implemented as *closures*, i. e., functions that return (anonymous) functions. The closures are used to parametrize the actual handlers that are returned, thus giving complex handlers a compact shape in the code. The handlers are defined in the directories

- `src/datatrack/handler/user/`,

- `src/datatrack/handler/local/disclosure/`,
- `src/datatrack/handler/local/attribute/`,
- `src/datatrack/handler/local/organization/`,
- `src/datatrack/handler/incident/`, and
- `src/datatrack/handler/category/`.

The third group of handlers is used to retrieve and show the data stored of an external service in the user interface. Apart from showing data, there also handlers for correcting and deleting data from the external service. All these handlers make use of the plug-in infrastructure for remote data and the data is passed on to the user interface, no matter what format is provided by the external service. The handlers are defined in the directory

- `src/datatrack/handler/remotedata/`.

The fourth group also uses the plugin-infrastructure for remote data and adds the data as new disclosures to the database. These are import handlers. Some services require an initialization and registration procedure for each client, for instance the TL server. For these cases, specific registration are implemented. In the current implementation, there is a registration handler for all A-PPLE/TL services that is specifically instantiated for the Wearable application scenario. The handlers are defined in the directories

- `src/datatrack/handler/importremote/` and
- `src/datatrack/handler/wearable/`.

The fifth group is a number of handlers that exposes the functionality of the TL recipient prototype. With these handlers in place, the Data Track becomes a full-featured TL recipient. Code from the original TL recipient implementation has been reused where possible. The implementation differ in the way they store persistent data, for instance cryptographic keys. In the Data Track, all persistent data, including the data used by TL, is stored in the database. The handlers are defined in the directory

- `src/datatrack/handler/insynd/`.

The sixth group comprises a number of handlers for different purposes. One handler is populating the database with static test data which is used to test the user interface. Another one can be used to send e-mails from the user interface. The mail server is configured at the program start via the environment variables `DATATRACK_SMTPSERVER`, `DATATRACK_SMTPUSER`, and `DATATRACK_SMTPPASSWORD`. This feature has been considered, but is not used at the moment. There is also a temporary handler for adding *incidents* to the database. Incidents would otherwise be added through TL messages. The handlers of this group would be removed before a productive release of the Data Track can be rolled out. The handlers are defined in the directories

- `src/datatrack/handler/testdata/`,

- `src/datatrack/handler/testadbokis/`,
- `src/datatrack/handler/bouncemail/`, and
- `src/datatrack/handler/incidentinject/`.

6.1.3. Remote Plug-in Extensions

The remote plug-in framework assigns remote capabilities to each organization ID. Capabilities are, for instance, showing and reviewing personal data, correcting the data, and deleting data. Each capability is addressed with a string which makes it easy to extend or reduce the number of capabilities for any organization. The framework is defined in the file

- `src/datatrack/remote/core.go`.

At the time of writing this report, six capabilities are specified and used:

import Imports the personal data in the Data Track.

get all Retrieves the personal data and passes it to the user interface in the format of the remote service.

delete all Deletes the personal data, i. e., all attributes, from the remote service's storage.

correct Corrects an attribute in the remote service's storage.

delete Deletes an attribute from the remote service's storage.

get policy Retrieves the data handling policy and passes it to the user interface in the format of the remote service.

Any remote service that uses A4Cloud's A-PPLE implements all six capabilities and can easily be connected to the Data Track. The Data Track comes with a template plug-in for A-PPLE which only needs to be completed with the URI of the remote service's A-PPLE web service and with access credentials. The template plug-in is defined in the directory

- `src/datatrack/remote/appl/`.

The Wearable application is a service that uses A-PPLE. The Data Track is therefore configured to connect to it and give the user access to and control over his data. The plug-in is derived from the A-PPLE template and is defined in the directory

- `src/datatrack/remote/kardiomon/`.

6.1.4. TL Recipient

The Data Track is in the words of the TL prototype a TL recipient. The code of the Data Track's TL implementation is mostly borrowed from the TL prototype. The main difference is the way persistent data is stored. In addition, the Data Track implements one extra feature, a concurrent light-weight thread that loops through all TL keys and queries all new messages. These messages are parsed and, if necessary, imported in the Data Track as disclosure or incident.

6.2. API

The API is part of the Data Track web service and allows other programs to connect to and interact with the Data Track. The client that connects to the API is the DT user interface which itself is provided by the Data Track web service. The web service connects to port 8000 and listens for incoming HTTP requests. The API follows a REST design, can handle all standard HTTP 1.1 verbs, and uses currently GET, POST, PUT, and DELETE.

At the time of writing, there are 122 different API calls implemented in the Data Track. Only one of them is particularly made for humans and not for machines: it is the call that prints the index of all available API calls. The call is

- `http://localhost:8000/v1`.

The following list summarizes the Data Track's API call families. Each family may have several members, for instance for sorting or filtering IDs. If the HTTP verb is not explicitly mentioned, GET is used. The prefix `http://localhost:8000` is omitted in the list.

`/v1/user`

Provides access to the username and picture. PUT sets the username and picture and GET queries for them.

`/v1/disclosure`

Enumerates all disclosure IDs.

`/v1/disclosure/:disclosureId`

Retrieves the disclosure with ID `'disclosureId'`.

`/v1/disclosure/:disclosureId/attribute`

Enumerates the attribute IDs of attributes that have been disclosed in the disclosure with ID `'disclosureId'`.

`/v1/disclosure/:disclosureId/implicit`

Enumerates the disclosure IDs of disclosures that the organization derived from the disclosure with ID `'disclosureId'`.

`/v1/disclosure/:disclosureId/downstream`

Enumerates the disclosure IDs of downstream disclosures with attributes from the disclosure with ID `'disclosureId'`.

`/v1/disclosure/toOrganization/:organizationId`

Enumerates all disclosure IDs of disclosures to the organization with ID `'organizationId'`.

`/v1/disclosure/:disclosureId/remotedata`

Retrieves (GET) or deletes (DELETE) the remote data linked with the disclosure with ID `'disclosureId'`. Note that the A-PPLE plug-in currently retrieves and deletes all remote data and not only the data linked with the disclosure.

`/v1/disclosure/:disclosureId/remotedata/attribute/:attributeId`
Updates (PUT) or deletes (DELETE) the remote attribute with ID `'attributeId'` that is linked with the disclosure with ID `'disclosureId'`. Note that the A-PPLE plug-in currently retrieves or deletes the remote data attribute no matter whether it is part of the disclosure or not.

`/v1/disclosure/:disclosureId/remotedata/policy`
Retrieves the data handling policy linked with the disclosure with ID `'disclosureId'`. Note that the A-PPLE plug-in fetches the policy of the first data attribute, assuming that all other attributes of the same disclosure are disclosed using the same policy.

`/v1/attribute`
Enumerates all attribute IDs.

`/v1/attribute/explicit`
Enumerates all attribute IDs of disclosed attributes.

`/v1/attribute/implicit`
Enumerates all attribute IDs of derived attributes.

`/v1/attribute/toOrganization/:organizationId`
Enumerates the attribute IDs of attributes that have been disclosed to the organization with ID `'organizationId'`.

`/v1/attribute/type`
Enumerates all attribute types.

`/v1/attribute/type/:typeId/value`
Enumerates all attribute values of attributes with type `'typeId'`.

`/v1/attribute/:attributeId`
Retrieves the attribute with ID `'attributeId'`.

`/v1/organization`
Enumerates all organization IDs.

`/v1/organization/receivedAttribute/:attributeId`
Enumerates all IDs of organizations the have been receiving the attribute with ID `'attributeId'`.

`/v1/organization/:organizationId`
Retrieves the organization data of the organization with ID `'organizationId'`.

`/v1/import`
Enumerates (GET) the organization IDs of organizations that have a import plug-in registered in the Data Track. Imports (PUT) the user's personal data from the organization.

`/v1/wearable/:username`
Imports the user's personal data from the Kardio-Mon server (Wearable application) and registers the Data Track as a TL recipient with the provided username.

`/v1/insynd`

Lists all cryptographic keys known to the TL recipient.

`/v1/incident`

Enumerates all incident IDs.

`/v1/testdata`

Resets the DT database with a predefined set of test data.

6.3. Remote Access

This section provides a consolidated view on messages and message formats between A-PPLE, TL, and the Data Track.

6.3.1. A-PPLE API

A-PPLE provides an API through which Data Track users can perform data subject access requests. Assuming the user is properly authenticated, the following API calls can be made:

`/apple-rest/pii/all`

GET Retrieves all PII of a given data subject. The PII Policy access control is not enforced since only the data subject about whom the data is being retrieved should use this functionality. The output is given in JSON format.

`/apple-rest/pii/all`

DELETE Deletes all PII of an owner from the repository. The corresponding policy and obligations are also deleted. The PII Policy access control is not enforced since only an authorized data subject should use this functionality.

`/apple-rest/pii`

PUT Updates / corrects a specific PII attribute with a given value. The purpose of usage as well as the action on the requested PII is needed in order to enforce the related PII Policy. The input is given in JSON format and must contain subject, attributeName, owner, purpose, action, newValue and authorization.

`/apple-rest/pii`

DELETE Deletes specific PII for a data subject from the repository. The corresponding policy and obligations are also deleted. Since PII must be accessed before deleted, the related PII policy must be enforced. Hence, purpose of usage as well as the action that will be performed is needed. The input is the same for the update API.

`apple-rest/policy`

GET Retrieves the policy used by the data controller to handle the accountability obligations concerning a given data subject. Requires that the engine has been set with a previous store policy call - which is not mandatory, since all personal data storage in the engine's PII store is coupled with a policy. Its parameters are `policyId` and `owner`. The output consists in the policy in XML format.

Note that the engine's API contains other methods that are not necessarily used by DT. See [dOSoTP⁺15] for full details.

6.3.2. Retrieving, Correcting, and Deleting Data

The Data Track uses the A-PPLE API to retrieve, correct, and delete the user's personal data in A-PPLE. The following variables are used by A-PPLE remote data plug-ins,

- the base URL of the A-PPLE API,
- the data owner,
- the data subject,
- the purpose,
- the attribute ID, and
- the new attribute value.

The *base URL* is defined in the remote data plug-in. The variables *data owner*, *data subject*, and *purpose* are defined in the credential attribute of the disclosure table in the DT database. The variables *attribute ID* and *new attribute value* are retrieved from the API call where *attribute ID* is the addressed resource and *new attribute value* is the request body of the API call.

Retrieve all remote data. The Data Track sends a HTTP GET request to the API call

- `/apple-rest/pii/all`

of A-PPLE including the query arguments `subject` and `owner`. Both query arguments are set to the value that is stored in the variable *data owner*. The response of A-PPLE is passed on as the response of the Data Track without any checking or processing. The DT API call for retrieving all remote data is

- `/v1/disclosure/:disclosureId/remotedata.`

Delete all remote data. The same procedure as for retrieving all remote data is followed with one difference: the GET request is replaced by a DELETE request. The query variables and all other details are the same as for retrieving all remote data. The DT API call for deleting all remote data is

- `/v1/disclosure/:disclosureId/remotedata.`

Note that A-PPLE sends or deletes all data that is stored for the user, since A-PPLE, in contrast to the Data Track, does not distinguish between different disclosures.

Update a single remote data attribute. The Data Track sends a HTTP PUT request to the API call

- /apple-rest/pii

of A-PPLE including the query arguments `subject`, `owner`, `purpose`, `action`, `resourceName`, and `newValue`. The first three query arguments are the values of the variables *data subject*, *data owner*, and *purpose*, respectively. The query argument `action` is hard-wired to the value 'update'. The query arguments `resourceName` and `newValue` are set to the values of the variables *attribute ID* and *new attribute value*, respectively. The response of A-PPLE is passed on as the response of the Data Track without any checking or processing. The DT API call for retrieving all remote data is

- /v1/disclosure/:disclosureId/remotedata/attribute/:attributeID.

Delete a single remote data attribute. The Data Track sends a HTTP DELETE request to the API call

- /apple-rest/pii

of A-PPLE including the query arguments `subject` and `owner`. Both query arguments are set to the values of the variables *data owner*. The response of A-PPLE is passed on as the response of the Data Track without any checking or processing. The DT API call for retrieving all remote data is

- /v1/disclosure/:disclosureId/remotedata/attribute/:attributeID.

6.4. Parsing A-PPLE Messages

DT is the interface of choice to communicate incident information to the data subjects. For that purpose, the obligations to notify in A-PPLE (implemented by `NotifyAction`) generate TL entries in a specific format, such that the incident information will be displayed to the data subject in an appropriate manner.

For example, the result of an incident pushed by the IMT tool to A-PPLE through the API call

```
1 PUT http://localhost:8080/apple-rest/notification/all HTTP/1
  .1
2 User-Agent: Fiddler
3 Host: localhost:8080
4 Content-Length: 74
5
6 {"resource": "country", "message": "This is a policy violation
  notification"}
```

Will generate the following output

```

1 HTTP/1.1 200 OK
2 Server: Apache-Coyote/1.1
3 Content-Type: application/json
4 Transfer-Encoding: chunked
5 Date: Tue, 25 Aug 2015 15:19:04 GMT
6
7 52
8 {"type":"Policy violation","value":"This is a policy
   violation notification"}
9 0

```

The JSON message format is the data sent to TL for each recipient registered in that specific engine instance.

The field value contains the message to be displayed to the user. The field type qualifies the notification according to what is determined by the accountability policy. The excerpts below illustrate the specification of notification types in two distinct obligations (a-ppl_rule_7 and a-ppl_rule_8).

```

1 <ob:Obligation elementId="a-ppl_rule_7">
2   <ob:TriggersSet>
3     <ob:TriggerPersonalDataAccessDenied/>
4   </ob:TriggersSet>
5   <ob:ActionNotify>
6     <ob:Media>e-mail</ob:Media>
7     <ob:Address>kardio.mon@a4cloud.com</
      ob:Address>
8     <ob:Recipients>Kardio-Mon</ob:Recipients>
9     <ob:Type>Unauthorized Personal Data Access
      Attempt</ob:Type>
10    </ob:ActionNotify>
11 </ob:Obligation>

```

And

```

1 <ob:Obligation elementId="a-ppl_rule_8">
2   <ob:TriggersSet>
3     <ob:TriggerOnViolation >
4       <ob:MaxDelay>
5         <ob:Duration>P0Y0M0DT0H2M0S<
          /ob:Duration>
6       </ob:MaxDelay>
7     </ob:TriggerOnViolation>
8   </ob:TriggersSet>
9   <ob:ActionNotify>
10    <ob:Media>e-mail</ob:Media>

```

```

11         <ob:Address>data.subject@a4cloud.com</
           ob:Address>
12         <ob:Recipients>alice</ob:Recipients>
13         <ob:Type>Policy violation</ob:Type>
14     </ob:ActionNotify>
15 </ob:Obligation>

```

In the Wearable-Co policy example of the A4Cloud demonstrator from the D7 (Instantiation for use case) work package, the following notification types are present:

- Data Lost, associated to `TriggerDataLost`.
- Policy violation associated to `TriggerOnViolation`.
- Unauthorized Personal Data Access Attempt associated to `TriggerPersonalDataAccessDenied`.
- Personal Data Deleted associated to `TriggerPersonalDataDeleted`.
- Data Collection associated to `TriggerOnUserRegistration`.
- Personal Data Sent to Data Processor associated to `TriggerPersonalDataSent`.

We suggest that notification types adhere to some best practice convention. Pilots of A-PPLE and Data track may help to identify which events are important to notify to the data subject, given their importance and frequency and importance. This is ongoing work in the context of the D7 work package.

The Data Track queries periodically the messages sent via TL and parses them. All received messages are stored in the database. One type of message is of particular interest, the *Personal Data Sent to Data Processor*. This message denotes a *downstream* disclosure which should be registered in the database. This can be done, provided that there is enough data conveyed in the TL message. The Data Track does not implement the parsing of the messages and registration of downstream disclosures yet, since the data format of the TL messages is not specified to the end. The Wearable application, as presented briefly in Section 2, specifies a downstream data processor, called Map-on-Web, that receives aggregate information about the location and overall health of Kardio-Mon's service users. However, the demonstrator does not at the time of writing *implement* the Map-on-Web service.

6.5. Importing Data Disclosures from Remote Data

This section discusses importing remote data from an instance of A-PPLE. Importing remote data requires that remote data can be retrieved. The algorithm basically works as follows:

1. Retrieve the remote data.
2. Parse the data.
3. Continue only if there is one or more data attributes.

4. Add the organization to the database, if not there already.
5. Determine the timestamp for the new disclosure.
6. Determine the data handling policy for the new disclosure.
7. Add the disclosure (with timestamp and policy, but without attributes) to the database.
8. For each remote data attribute,
 - a) add the attribute to the database and
 - b) assign the attribute to the disclosure (through the table *disclosed*).

The timestamp is determined by the modification time of the attribute that has been modified (or disclosed) as the last one within the retrieved remote data. Also, the import plug-in would retrieve all remote data attributes and add them as new disclosure, no matter whether some or all data attributes have been imported before. These shortcomings can be solved by making the import plug-in to group the attributes according to their modification time and their data handling policy and add a separate disclosure for each group. The DT database would then reject duplicate disclosures. This is future work at the time of writing this deliverable.

7. Log Audit for Validating Policy Adherence

Log audits are the technical means to retrospectively reconstruct and analyze system activities for determining if the actions executed by the system are in accordance with the policies. This section presents a review of existing mechanisms that analyze the logs for validating privacy policy adherence.

Privacy regulations and legislation such as the Health Insurance Portability and Accountability Act (HIPAA)¹⁹, the EU Data Protection Directive²⁰, etc., specify, among others, constraints such as; in order to execute an action or as a result of an action, some other action need to be executed. Few examples are, breach notifications, retention period, etc. In the machine readable privacy policy languages that attempt to enforce privacy regulations, these requirements are specified as obligations, which can be pre-obligations, post-obligations, conditional obligations, and recurring obligations (temporal constraint). However the obligations that occur in future cannot be determined during enforcement but can be validated through log auditing, thus log audits complement the enforcement mechanisms and also log audits make the execution of the processes verifiable. The later, in particular is suitable for the break-the-glass events.

The underlying idea of verifying logs against the rules has been long employed, for example, in Intrusion Detection Systems (IDS). Several mechanisms are purposed in the scientific literature to examine the logs for policy conformance. In particular, we are interested in the mechanisms that analyze the logs to verify if system activities recorded in the logs are consistent with the enterprises' data handling practices. A literature study was conducted to review the relevant literature that proposes technical solutions for performing log audits in the context of privacy. A total of 14 relevant literature was identified. The identified techniques are similar to IDS techniques as noted above, and thus follows two common IDS approaches, namely, misuse-based and anomaly-based IDS. We categorize the identified techniques into privacy misuse detection (6 out of 14) and privacy anomaly detection (the remaining 8) from their constructions that follows the IDS approach. Figure 11 presents our classification. Each category is briefly explained in the following subsections.

7.1. Privacy Misuse Detection

The common characteristic of the technical architecture in this category of solutions is that the privacy misuse patterns used for detection are explicit. In general, these patterns are known in advance, and are looked up in the logs for a mismatch in case of compliant patterns or a match in case of non-compliant patterns to detect violations. Three different look up approaches are observed in the technical design of the six identified solutions within this category, namely, pattern matching, query-based reasoning, and model checking:

Pattern matching Accorsi [Acc08] and Accorsi et al. [AS08] propose pattern matching auditing algorithms for verifying from the logs if the obligations specified in the policies are satisfied or not. The patterns are derived from user defined preferences, which are laid down by

¹⁹<http://www.hhs.gov/ocr/privacy/>, accessed on 2015-08-03.

²⁰<http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31995L0046:en:HTML>, accessed 2015-08-03.

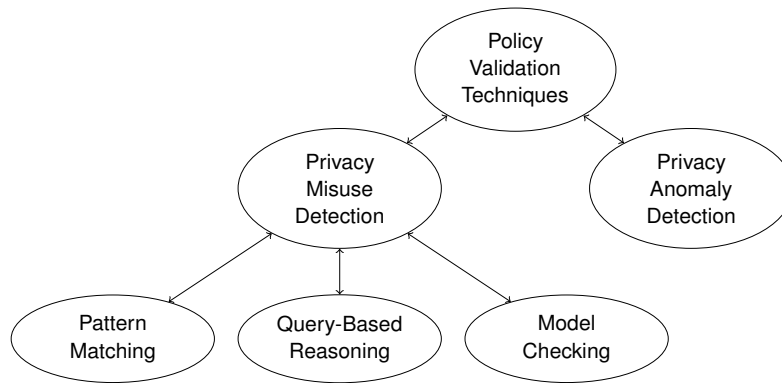


Figure 11: Technical classification of solutions that validates policy adherence using log audits.

the end-users in their privacy settings - a feature available in many websites. Accorsi derives violations to the user defined preferences as patterns. During the auditing step the logs that corresponds to an end-user is sequentially looked up for a match. A violation is detected if a match is identified. Accorsi et al. propose a novel approach for pattern matching to improve the efficiency of the log audits. Here the logs that corresponds to an end-user are pre-processed to form an action tree and are pruned against the actions specified in the user defined preferences. At the end of the look up the nodes that remains in the tree comprise the total violations.

Query-based reasoning Similar to Accorsi and Accorsi et al., Samavi et al. [AS08] derive the patterns from obligations specified by the end-users for their personal data management. Samavi et al. propose a query-based reasoning approach to verify the fulfillment of these obligations. For this purpose, they defined two ontologies in their framework; *i*) L2TAP (Linked Data Log to Transparency, Accountability and Privacy) ontology for logging of events and to provide provenance assertions, *ii*) SCIP (Simple Contextual Integrity Privacy) ontology for providing semantics to the L2TAP logs and uses SPARL (Simple Protocol and RDF Query Language) queries with RDFS (Resource Description Framework Semantics) reasoning support for auditing. SPARL queries first identify the obligations that correspond to the data items linked with an access request. Next, the SPARL queries evaluate the status (fulfilled, pending, violated) of these obligations to determine the compliance of the access request.

Model checking In this approach, data protection legislations, or business contracts are expressed in formal languages. These logical forms are then used as patterns, which are used by a model checker to deduce from the logs if a given formula is satisfied or not. Garg et al. [GJD11] use first-order logic to express the HIPAA Privacy Rules and describe an audit algorithm that uses the formally expressed HIPAA policies and the audit trails to validate the policy adherence. Similarly, Butin et al. [BLM13] formalize parts of the EU Data Protection Directive obligations and Banescu et al. [BPZ12] formalize business processes, using first-order logic to verify whether the system activities recorded in the logs satisfy the formal requirements.

7.2. Privacy Anomaly Detection

This category of solutions focus on detection of unauthorized actions performed over personal data, especially data access and transfer operations. However the solutions in this category operates in a different context, the patterns here are not random as in the misuse category but looking for signals of unexpected combination or bursts of activities. The anomaly-based detection systems identify events that deviate, from a “normal” profile in the case of supervised method, or from the other items in a dataset in the case of unsupervised method.

Supervised learning In the case of supervised learning, a classifier needs to be trained to learn normal and abnormal behavior in order to detect anomalies. Venter et al. and Boxwala et al. [VOE04, BKGOM11] manually build the training set whereas Bhattacharya et al. [BDK⁺05] use a data mining approach. Further, Heatley et al. and Gupta [HO98, Gup13] use statistical methods to build the training set. Given the training set, the classifier uses statistical techniques such as *k*-nearest neighbor to predict the probability of an access request to be an outlier.

Unsupervised learning In this method, data from the access logs is aggregated into micro clusters for the detection algorithm to determine an anomaly. Chen et al. [CM11] uses statistical methods such as graph-based modeling and dimensionality detection to aggregate the features in the audit trails to form community structures. Next the deviation probability of the items with respect to the rest of the items in the cluster is determined using the *k*-nearest neighbor algorithm. Fabbri et al. [FL11] follow the approach of Chen et al. but use the community structures to define social graphs for explaining the context of the events recorded in the access logs, which helps privacy officers to detect unauthorized access.

7.3. Potential Use in A4Cloud

According to the Swedish Patient Data Act (“Patientdatalagen”) and HIPAA, patients have the right to request for the access logs pertaining to their electronic-health record. A-PPLE logs the enforcement decisions and the logs that are specific to the data subjects can be made available to the data subjects using the TL component, which provides the security and privacy assurance for the transmission of such information. From the logs²¹ meaningful information can be deduced in terms of the data handling practices of the service provider. As described above, mechanisms such as association rules are used to form social graphs, which describe the relationship between the entities that accessed the personal data for detecting anomalous activities. Similarly, events related to the communication between the service provider and downstream data processors can be aggregated using data mining algorithms to detect anomalous transfer of personal data and also to some extent deduce information about the transfer. If a copy of the machine readable policy is available to the data subjects, matching algorithms can be used to validate the actions that are recorded in the logs against the policy.

²¹ If the information registered in the logs are *complete*.

8. Plug-In for Assessing Policy Violations

The Plug-in for Assessment of Policy Violations (PAPV) evaluates the relevance of previously detected policy violations. By using it, data subjects (or their representatives) can check which policy violations are the most relevant.

The input to the PAPV is a collection of instances of policy violations, which are received from the Data Track tool, where an instance of policy violation is any piece of evidence that describes an occurrence of a policy violation event. Then, for each violation it produces an assessment of its relevance, in the form of an ordered measure that can be quantitative or qualitative. Finally, it prepares an ordered list of policy violations, sorted by their level of importance.

In this section we describe the main architecture of this plug-in and the design decisions associated to its implementation.

8.1. Architecture

Figure 12 gives an overview of the architecture of the PAPV. Policy violations can be detected by several tools of the A4Cloud toolkit; however, all of them are communicated to the affected data subjects by means of the Transparency Log. For this reason, we do not take the tool that issues the policy violation into consideration, so we will assume that policy violations arrive from the Transparency Log. The Data Track consumes the data that is communicated by the Transparency Log, including policy violations. According to A4Cloud's Framework of Evidence [WP14], policy violations are XML records that contain information about the actions associated to the violation. The Data Track simply relays these XML records to the PAPV and in response obtains a new version of these records that includes a field "Relevance" with the result of the evaluation.

8.2. Relevance Assessment

The main functionality of this plug-in is to assess the relevance of policy violations. To this end, first it is necessary to define what will be the criteria for performing such an assessment. There are several possible approaches for evaluating the relevance of a policy violation, and we classify them according to what is the perspective that drives the assessment. We can assume that in most of cases, policy violations occur as actions that have some effect over some data. Therefore, this leads to two main approaches for the assessment: driven by the data, or driven by the action.

In the next subsections we analyze both approaches, and we conclude that, in our context, the action-driven assessment is more adequate.

8.2.1. Data-Driven Assessment

This type of assessment evaluates the relevance of a violation depending on the nature of the affected data. Although, in principle, this could be an ideal approach, we find that there are a lot of problems when it comes to analyzing the nature of the affected data.

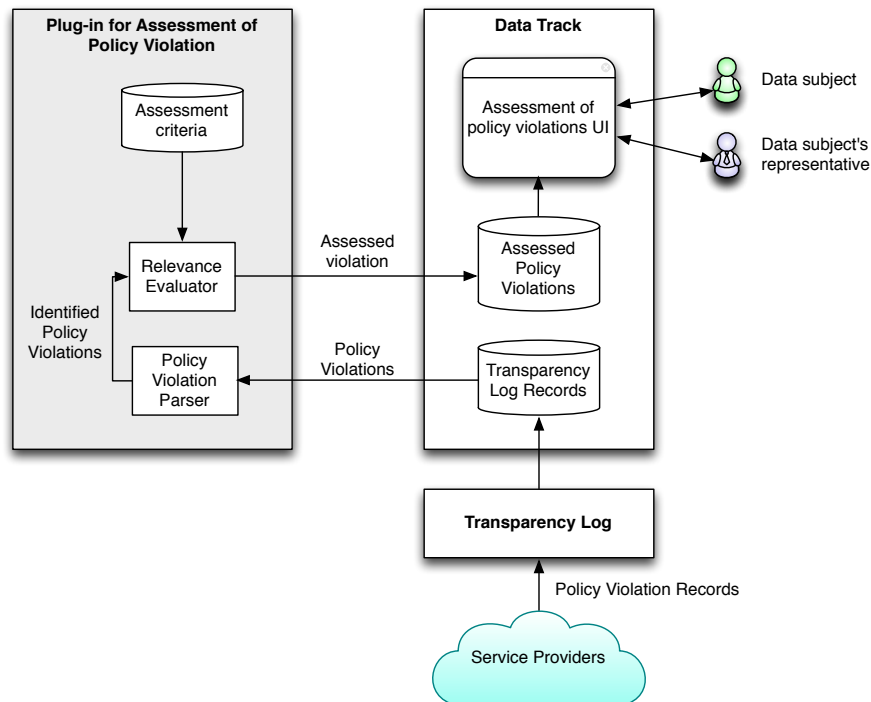


Figure 12: Overview of PAPV architecture.

In general, it is a difficult task to dynamically analyze the data in order to infer their level of sensitivity. There are multiple technical problems associated to it, such as unknown data formats, structures, languages, etc. In addition, dynamically analyzing the affected data implies that this data must be somehow read, which is troublesome since the fact that data must be available for reading raises privacy concerns.

A less ambitious approach is to assume that data has been previously labeled or categorized with respect to some scale of sensitivity or confidentiality, according to its semantic nature. For example, as discussed in NIST SP 800-122 [MGS10]:

“All Personally Identifiable Information (PII) is not created equal. PII should be evaluated to determine its PII confidentiality impact level, (...) so that appropriate safeguards can be applied to the PII. The PII confidentiality impact level – low, moderate, or high – indicates the potential harm that could result to the subject individuals and/or the organization if PII were inappropriately accessed, used, or disclosed.”

However, as noted by the Article 29 Data Protection Working Party, such task is not easy, since there are numerous factors that influence the classification of data as sensitive. For example, current European Data Protection Directive classifies as sensitive any *“data revealing racial or ethnic origin, political opinions, religious or philosophical beliefs, trade-union membership”* [art11]. Although these categories may seem clear, the fact is that there is a lot of room for interpretation and actual implementation is far from being harmonized between Member States [art11].

8.2.2. Action-Driven Analysis

A second, and much more feasible, option is to assume that all data belongs to the same category of sensitivity (i.e., we could say all data is “sensitive”), and what drives the impact on the relevance of the violation is the type of operation performed over the data or the action that has occurred. This can be considered as a static analysis, since there is no dynamic evaluation of the violation.

This approach can be linked to the work in A4Cloud’s Framework of Evidence [WP14], where evidence records represent actions performed over resources. The format for policy violations defined by this framework includes a field `ActionID` for specifying the action occurred. Therefore, we will assume that all instances of policy violations should have associated a value for this field, from a predefined set of actions.

Once we have opted for this approach, we have to decide what would be the possible relevance levels and how to perform the assessment of the occurred actions. To this end, we will make use of the guidelines from the Privacy Risk Management Methodology from the French Data Protection Authority (CNIL) [cni12]. According to these guidelines, the possible levels of relevance based on their prejudicial effect are the following:

Negligible Data subjects either will not be affected or may encounter a few inconveniences, which they will overcome without any problem (time spent re-entering information, annoyances, irritations, etc.).

Limited Data subjects may encounter significant inconveniences, which they will be able to overcome despite a few difficulties (extra costs, denial of access to business services, fear, lack of understanding, stress, minor physical ailments, etc.).

Significant Data subjects may encounter significant consequences, which they should be able to overcome albeit with serious difficulties (misappropriation of funds, blacklisting by banks, property damage, loss of employment, subpoena, worsening of state of health, etc.).

Maximum Data subjects may encounter significant, or even irreversible, consequences, which they may not overcome (financial distress such as substantial debt or inability to work, long-term psychological or physical ailments, death, etc.).

Now that the possible levels of relevance are defined, we have to establish the criteria to base the relevance assessment. In order to do this, it is necessary to analyze the possible actions associated to policy violations from the perspective of the potential consequences to data subjects. Table 1 shows an example of the result of such analysis.

Note that this are only examples for illustration purposes, and a more extensive analysis should be done for real deployment purposes. This would require also that all possible actions that origin policy violations are clearly defined beforehand.

ActionID	Description	Relevance
DENY_ACCESS	Data read access denied/not allowed (e.g. conflicting policies or incorrect access/authorizations configurations at another point of the Cloud Service)	Limited
DENY_MODIFICATION	Data update access denied/not allowed (e.g. conflicting policies or incorrect access/authorizations configurations at another point of the Cloud Service)	Limited
DATA_INEXISTENT	Data inexistent (e.g. data loss or incorrect stored)	Significant
REQUESTED_DELETION_NOT_EXECUTED	Deletion not executed (data still available)	Limited

Table 1: Example of mapping between actions and relevance levels.

8.3. Implementation of the PAPV

Since we have opted for a static analysis approach, the actual implementation of the PAPV in Go²² is simple and is mostly comprised of the following files:

relevall.go This code contains the declaration of the constant values that represent the possible relevance levels, described in the previous section. These levels are “Negligible”, “Limited”, “Significant”, and “Maximum”. In addition, a constant value “Not Applicable” is defined for the situations where the assessment is not possible.

criteria.go This code contains the assessment criteria, in the form of a mapping between possible actions and their associated relevance level. Given the design decisions above mentioned, the criteria in this case can be seen as a lookup table.

papv.go It contains the function `AssessPolicyViolationRelevance`, which wraps the main functionality of the PAPV. Additionally, it also implements the parsing functionality needed for reading the XML records that convey policy violations.

proto.go This code implements a simple command-line utility that permits to execute the PAPV as a standalone tool. It takes as argument a XML file that contains the policy violated record to analyze.

8.4. Current State of Integration

Due to delays in other parts of the A4Cloud project related to policy violations, PAPV has not been fully integrated into the Data Track at the time of writing. We stress that our implementation is ready, as well as the necessary work to ease the integration with the Data Track.

²²We opted to use the same language as the Data Track to ease integration.

9. Privacy-Preserving Word Search

As already analyzed in deliverable D37.2 [FHnOP14a], the cloud computing technology raises severe security and privacy issues: In particular, cloud customers consider the cloud provider itself as a potential adversary and do not wish to reveal the content of their outsourced data. While data encryption seems a viable solution for privacy protection, traditional encryption mechanisms fall short when it comes to mining/processing and more specifically searching over the data. Several solutions have been proposed for privacy preserving word search (see D37.2 [FHnOP14a]). In [EOM14], we address the problem of delegated word search whereby in addition to the data owner itself (for example the data controller), some authorized third-parties can perform search operations over private data.

We consider a scenario where a data owner outsources some privacy sensitive data to a cloud server and wishes to delegate part of the search operations to authorized third parties. An illustrative example in A4Cloud of such a requirement can be a scenario wherein due to regulatory matters, outsourced (confidential) logs or evidences need to be searchable by third parties such as data protection commissioners or auditors.

In our proposed solution, the data owner constructs a searchable index with all words listed in its files and applies a private information retrieval [TP10] to guarantee that an unauthorized party including the cloud itself does not discover any information about the search query and its result. The newly proposed solution outperforms existing ones thanks to a combination of Cuckoo hashing [PR04] with private information retrieval for the search operation. The delegation operation is assured thanks to the use of attribute based encryption (ABE) which only allows parties holding certain “attributes” to search over the data. Since the data owner may also wish to revoke the authorized party at any point of time, the solution also defines an efficient revocation protocol which combines the use of ABE [BSW07] and oblivious pseudo random functions (OPRF) [JL09] which allow two parties to jointly compute the output of some pseudo-random function without discovering each others’ input. With the use of OPRFs, the proposed protocol allows the cloud server to generate a one-time token for the authorized user without discovering the content of the query.

9.1. Overview

A privacy preserving delegated word search involves the following three parties:

- **Data owner \mathcal{O} :** It possesses a large file F that it outsources to the cloud server \mathcal{S} . Without loss of generality, we assume that the number of distinct words in F is n and the corresponding set is defined as $\mathcal{L}_\omega = \{\omega_1, \omega_2, \dots, \omega_n\}$.
- **Cloud server \mathcal{S} :** It stores an *encrypted* version of the outsourced file F and a searchable index \mathcal{I} of the set \mathcal{L}_ω of “distinct” words present in F .
- **Authorized user \mathcal{U} :** It has access to a set of credentials that enable it to perform search queries on F . This authorized user could be an auditor which as part of its auditing task has to search the activity logs of \mathcal{O} .

The proposed privacy preserving delegated word-search mainly comprises the following phases:

- Setup phase:** during this phase, data owner \mathcal{O} encrypts the file F using AES encryption and builds a searchable index using the algorithm BuildIndex_O which outputs a list of MACs $\mathcal{L}_H = \{h_1, h_2, \dots, h_n\}$, such that $h_i = \mathcal{H}_{\text{mac}}(K_{\text{mac}}, \omega_i || \text{fid})$. Next, \mathcal{O} defines the access policy AP that will be associated with file F and finally forwards (via a secure channel) the file identifier fid, the encryption C of file F , the list of MACs $\mathcal{L}_H = \{h_1, h_2, \dots, h_n\}$, the access policy AP and a secret key δ to cloud server \mathcal{S} . This secret key is used to compute some tokens for the dedicated user. The computation of this token is performed using an oblivious pseudo-random function (OPRF) [JL09] which allows \mathcal{U} and \mathcal{S} to jointly compute the output of some pseudo-random function without discovering each others' input. Since OPRF is deemed to be demanding, \mathcal{O} requests the cloud server to perform the more computationally intensive operations (i.e. OPRF and Cuckoo Hashing). Henceforth, the processing at the cloud comprises two operations: the first one is to compute OPRF over the MACs in $\mathcal{L}_H = \{h_1, h_2, \dots, h_n\}$ using the secret key δ . The second operation is to build an index with the resulting values using Cuckoo hashing. Additionally, to delegate the word search capabilities on the encrypted file F to some third parties, data owner \mathcal{O} encrypts its MAC key K_{mac} under its access policy AP using attribute-based encryption (ABE) and provides cloud server \mathcal{S} with the resulting ciphertext. An authorized user \mathcal{U} who in principle possesses the credentials that satisfy the access policy AP, will be able to decrypt the MAC key K_{mac} . This MAC key will then be used by \mathcal{U} to perform word search on \mathcal{O} 's file during the next phase.
- OPRF phase:** To search the encrypted file C for some word ω , the authorized user \mathcal{U} first needs to generate the proper token to construct the proper word search query. The token generation consists of executing an OPRF protocol between the authorized user \mathcal{U} and the cloud server \mathcal{S} . On inputs of the word ω , the file identifier fid and the MAC key K_{mac} , \mathcal{U} first computes an OPRF query $\mathcal{Q}_{\text{oprf}}$ to evaluate $f_\delta(h) = g^{1/(\delta+h)}$ and forwards it to cloud server \mathcal{S} . Upon receipt of $\mathcal{Q}_{\text{oprf}}$, \mathcal{S} calls the OPRF algorithm $\text{Response}_{\text{oprf}}$. This algorithm uses the secret OPRF key δ and the OPRF query $\mathcal{Q}_{\text{oprf}}$ to output an OPRF response $\mathcal{R}_{\text{oprf}}$. OPRF allows authorized user \mathcal{U} to receive a word search token without disclosing anything to cloud server \mathcal{S} about the word ω that \mathcal{U} is interested in. This response is obfuscated (through simple encryption) in such a way that only \mathcal{U} can have access to it and derive the token. \mathcal{U} further computes the OPRF response $\mathcal{R}_{\text{oprf}}$ by decrypting the received ciphertext and deriving the word search token using the OPRF algorithm.
- Search phase:** After obtaining the token τ corresponding to the word ω , \mathcal{U} runs the algorithm Query which first computes $H(\tau) = (x, y)$ and $H'(\tau) = (x', y')$. Then, it computes two PIR queries $(\vec{\alpha}, \vec{\alpha}')$ to retrieve the x^{th} and the x'^{th} row of a (k, l) binary matrix and sends the word search query $\mathcal{Q} = (\vec{\alpha}, \vec{\alpha}')$ to cloud server \mathcal{S} . On receiving \mathcal{U} 's search query, cloud server \mathcal{S} runs algorithm Response which computes the two sets of t PIR responses $\mathbb{R} = \{\vec{\beta}_1, \vec{\beta}_2, \dots, \vec{\beta}_t\}$ and $\mathbb{R}' = \{\vec{\beta}'_1, \vec{\beta}'_2, \dots, \vec{\beta}'_t\}$. \mathcal{S} sends then its word search response $\mathcal{R} = \{\mathbb{R}, \mathbb{R}'\}$ to \mathcal{U} .
 To verify whether ω is in the encrypted file C , the authorized user \mathcal{U} runs the algorithm Extract using the PIR building block and checks accordingly whether $\vec{b} = \mathcal{H}(\tau)$ or

$\vec{b'} = \mathcal{H}(\tau)$. If it is the case, then Extract outputs 1 meaning that $\omega \in F$; otherwise, Verify outputs 0.

- **Revocation phase:** For sake of simplicity, we assume that the revocation is attribute-based. To revoke an attribute att_i , \mathcal{O} runs the algorithm Revoke simply outputs a new access policy AP' that will be given to the cloud server \mathcal{S} . For instance, if we assume that the initial access policy AP of \mathcal{O} states that auditors from EU and the US can perform word search on \mathcal{O} 's files, then a revocation of attribute US will lead to a new access policy AP' that says that only auditors from the EU can perform word search. In this manner, auditors from the US will no longer have access to \mathcal{O} 's file.

9.2. Performance Evaluation

In order to evaluate the performance of the proposed solution, we have implemented all the underlying algorithms and analyzed the cost of each phase both at the client side and the cloud side. All algorithms were coded in python and ran on an Intel Core 2 Duo 2.80GHz with a RAM of 3.8GB under Ubuntu 14.04 LTS. We consider different cases with different index sizes: the number of words per index varies from 1 to 800000. Each algorithm was executed 50 times for each scenario and average values are used.

During the setup phase, the data owner is only required to encrypt the file to be outsourced using a symmetric encryption and to compute a MAC h_i for each word $\omega_i \in \mathcal{L}_\omega$. On the other hand, the cloud server computes the OPRFs (i.e. tokens) $\tau_i = f_\delta(h_i)$ and builds the corresponding index \mathcal{I} by following the algorithm of Cuckoo hashing. Figure 13 shows that the cost of this phase both at the user and at the cloud side is naturally linear with respect to the number of words and remains affordable for a lightweight user(uploading 200000 words approximately takes 3 seconds).

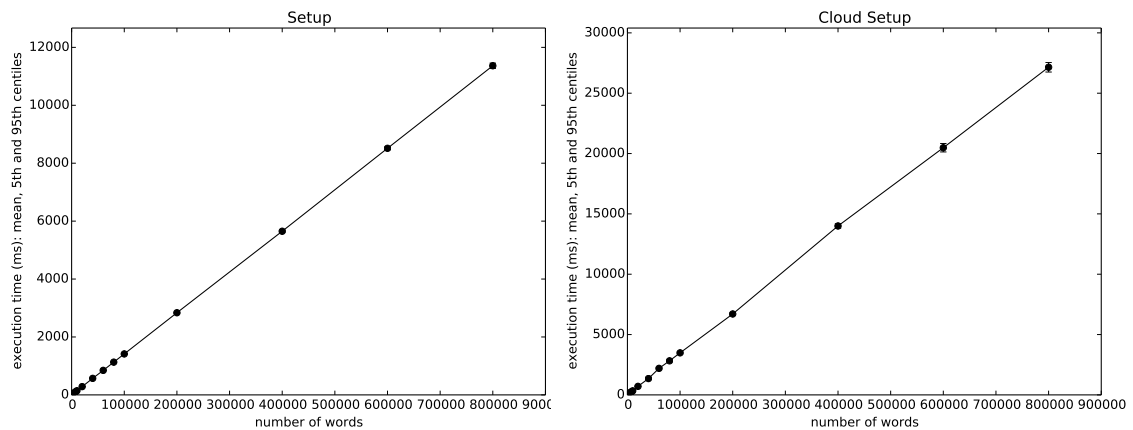


Figure 13: Cost of the setup phase.

During the OPRF phase (token query), although the computation of the OPRF may be deemed

computationally demanding as it calls for exponentiations, it can be efficiently parallelized at the cloud server. Figure 14 shows the cost of the generation of the query at the user side and of the response at the cloud side. While the execution time at the user side is very efficient, results at the cloud side seem acceptable and can be even more efficient in a real life deployment.

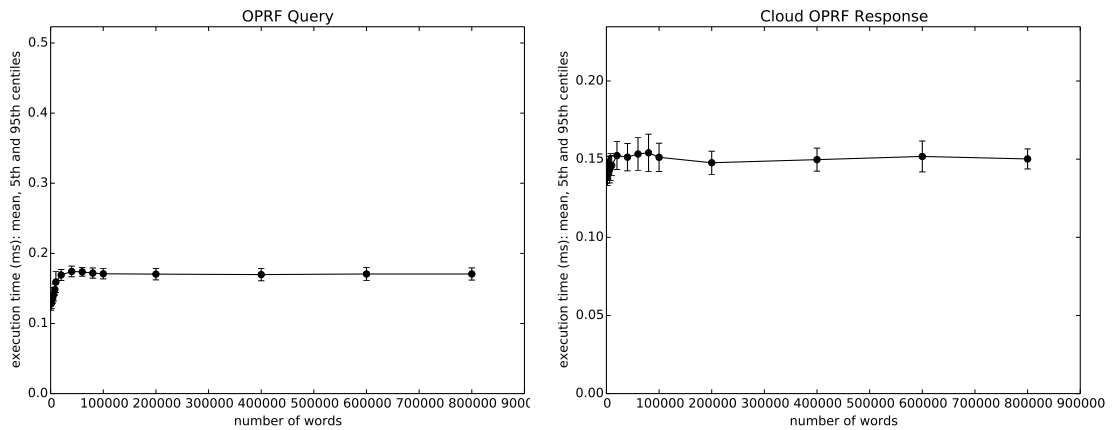


Figure 14: Cost of the OPRF phase.

During the search phase, similarly to the OPRF phase, some would argue that using PIR to compute the responses of the cloud server to word search queries is computationally intensive; however, we note that this computation consists of *matrix multiplications* which can easily be parallelized. The performance results shown in Figures 15 and 16 consider the cost of the computation of the PIR query, the PIR response, and the extraction of the PIR response at the user side. These figures show that the computation of the PIR query and responses at the user side depends on the size of the PIR matrix at the cloud side.

Finally, the revocation phase is very efficient since it does not require the re-encryption of the outsourced files and only calls for an update of the access policy of the data owner at the cloud server.

9.3. Potential Use in A4Cloud

In this section we have presented a new privacy preserving word search solution that allows an authorized party to search for a word without revealing any information to unauthorized parties including the cloud. Within the A4Cloud project, this solution can easily be used as a plug-in for the combination of the A-PPL engine and AAS. AAS usually searches for policy violations within the A-PPL Engine's logs stored in the TL (or another storage service). Therefore, the A-PPL Engine may encrypt these logs following the setup phase of our proposed protocol and provide the additional parameters to TL. AAS may further request the search token from the TL and further search for a specific word without revealing any information (neither the content of the query, nor the corresponding response) to TL.

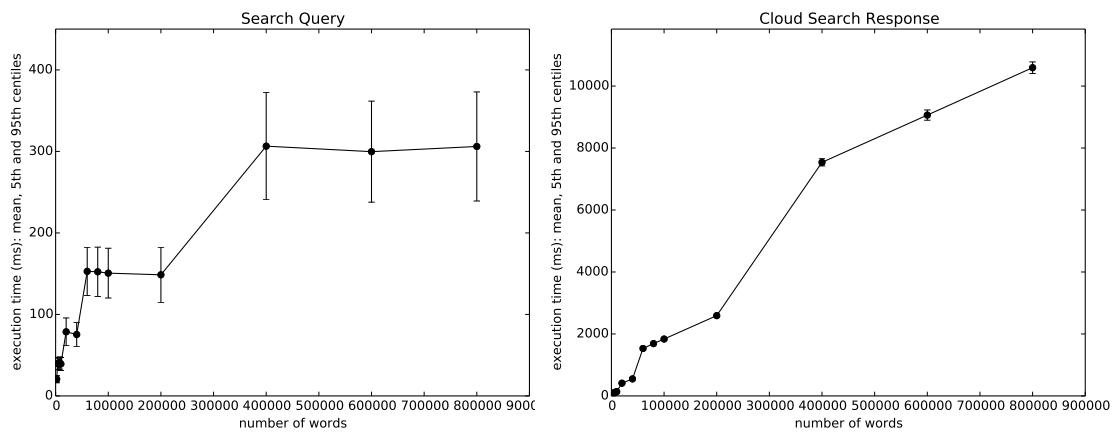


Figure 15: Cost of the search phase.

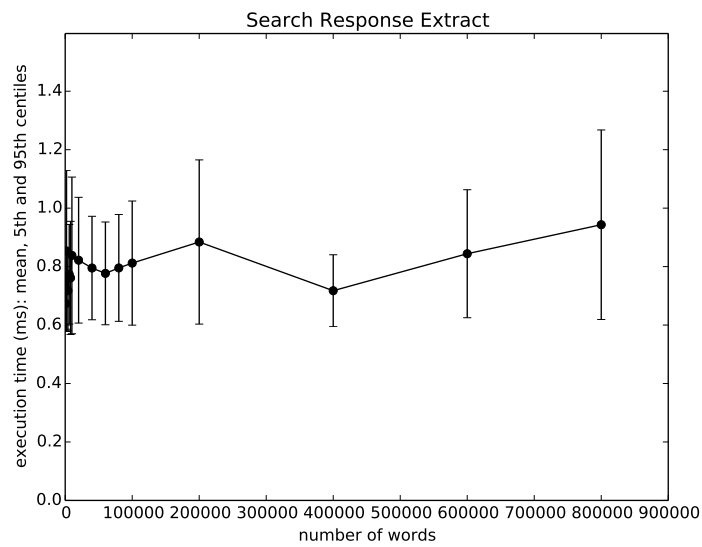


Figure 16: Extraction of the search response.

10. Summary

This deliverable presented the Data Track, a transparency-enhancing tool for (cloud) data subjects. The Data Track empowers data subjects both by informing them of who knows what about them, but also in terms of enabling them to exercise their rights to request access to, correction of, and deletion of their personal data stored remotely at online service providers. We have developed the Data Track as a stand-alone tool that not only works with A4Cloud services but conceptually can interact with a large number of different online services with different approaches to giving data subjects access to their personal data electronically.

The user interface of the Data Track is web-based, while the back-end is running locally on the data subject's computer. The user interface currently consists of two views on data disclosures: the trace view and the timeline view. The trace view "traces" attributes and service providers, letting users answer the question of "what information about me have I sent to which online services?". The timeline view shows data disclosures ordered chronologically, enabling users to answer questions like "what information did I send on this date?" and "what day of the week do I tend to disclose more personal information?".

The back-end of the Data Track is implemented in the Go programming language and uses a plug-in based architecture to facilitate support for different service providers. There is a local database storing data disclosures that are provided to the user-interface through a rich API, at the time of writing consisting of 122 different API calls. Beyond access to data disclosures, the API enables the remote access, correction and deletion requests to service providers (with the help of the plug-ins), and to retrieve Transparency Log messages sent from A4Cloud-enabled service providers.

The deliverable also presented research results on the Transparency Log tool, a literature study on how to use log analysis to detect privacy violations, a plug-in for assessing policy violations, and a cryptographic scheme for privacy-preserving word search. The cryptographic schemes that make up the Transparency Log and a use-case have been published [PP15a, PP15b, PP15c, Pul15, RPR15], and the source code of two proof-of-concept implementations have been made available under open-source licenses. The literature study presented a classification of policy validation techniques focused on the differences between the privacy anomaly detection and the privacy misuse detection approaches. The plug-in for assessing policy violations uses an action-driven analysis to assess the relevance of policy violations to end-users. A proof-of-concept implementation is ready but final integration is not complete due to delays on other parts of the A4Cloud tool-set around policy violations. The cryptographic scheme for privacy-preserving word search could be used to provide strong privacy protections for an auditing tool in A4Cloud, and has also been published [EOM14].

Moving forward, the goal is to release an open-source version of the Data Track that can visualise parts of the data provided by Google through their Google Takeout²³ service. At the time of writing, our conclusions are that the design of the Takeout service makes automatically retrieving the data hard. Therefore, the DT plug-in will guide the user through the steps necessary to perform a "takeout" and then enable the user to upload the resulting zip-file to DT. We are currently in the process of parsing parts of the content of the zip-file into data disclosures.

²³<https://encrypted.google.com/takeout/>, accessed 2015-08-27.

References

- [ABFH⁺15] Julio Angulo, Karin Bernsmed, Simone Fischer-Hübner, Christian Frøystad, Erlend A. Gjøre, Henrik Andersson, and Daniel Lindegren. D:D-5.4 User Interface Prototypes V2. Project deliverable D:D-5.4, A4Cloud Project, September 2015.
- [Acc08] Rafael Accorsi. Automated Privacy Audits to Complement the Notion of Control for Identity Management. In *Policies and Research in Identity Management*, volume 261, pages 39–48. Springer, 2008.
- [AFPW15] Julio Angulo, Simone Fischer-Hübner, Tobias Pulls, and Erik Wästlund. Usable transparency with the data track: A tool for visualizing data disclosures. In Bo Begole, Jinwoo Kim, Kori Inkpen, and Woontack Woo, editors, *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems, Seoul, CHI 2015 Extended Abstracts, Republic of Korea, April 18 - 23, 2015*, pages 1803–1808. ACM, 2015.
- [art11] Advice paper on special categories of data (“sensitive data”). Technical report, Article 29 Data Protection Working Party, 2011.
- [AS08] Rafael Accorsi and Thomas Stocker. Automated Privacy Audits Based on Pruning of Log Data. In *12th Enterprise Distributed Object Computing Conf. Workshops*. IEEE, 2008.
- [BDK⁺05] Jaijit Bhattacharya, Rajanish Dass, Vishal Kapoor, Debamitro Chakraborti, and SK Gupta. Privdam: Privacy Violation Detection and Monitoring Using Data Mining. 2005.
- [BKGOM11] Aziz Boxwala, Jihoon Kim, Janice Grillo, and Lucila Ohno-Machado. Using statistical and machine learning to help institutions detect suspicious access to electronic health records. *Journal of the American Medical Informatics Association*, 18(4):498–505, 2011.
- [BLM13] Denis Butin and Daniel Le Métayer. Log Analysis for Data Protection Accountability (Extended Version). Research report no. 8432, INRIA, 2013.
- [BNG11] Hila Becker, Mor Naaman, and Luis Gravano. Beyond trending topics: Real-world event identification on twitter. In *Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media (ICWSM’11)*, 2011.
- [BOH11] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D³ data-driven documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, 2011.
- [BPZ12] Sebastian Banescu, Milan Petković, and Nicola Zannone. *Measuring Privacy Compliance Using Fitness Metrics*, volume 7481 of *LNCS*, pages 114–119. Springer Berlin Heidelberg, 2012.

- [BSW07] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *Security and Privacy, 2007. SP '07. IEEE Symposium on*, pages 321–334, 2007.
- [CM11] You Chen and Bradley Malin. Detection of Anomalous Insiders in Collaborative Environments via Relational Analysis of Access Logs. In *Proc. of the 1st ACM Conf. on Data and Application Security and Privacy (CODASPY)*, pages 63–74. ACM, 21–23 Feb 2011.
- [cni12] Methodology for privacy risk management. Technical report, CNIL, 2012.
- [DFG⁺14] Brian Dziminski, Massimo Felici, M. Carmen Fernández Gago, Frederic Gittler, Theo Koulouris, Ronald Leenes, Jesus Luna, Maartje Niezen, David Nuñez, Alain Pannetrat, Siani Pearson, Jean-Claude Royer, Dimitra Stefanatou, and Vasilis Tountopoulos. D:C-2.1 Report detailing conceptual framework. Project deliverable D32.1, A4Cloud Project, October 2014.
- [dOoTPV⁺15] Anderson Santana de Oliveira, Võ Thành Phúc, Efthymios Vlachos, Alexander Garaga, Monir Azraoui, Kaoutar Elkhyaoui, Melek Önen, Walid Benghabrit, Jean-Claude Royer, Michela D’Errico, Ali El Kaafarani, Martin Gilje Jaatun, and Inger Anne Tøndel. D:D-3.3: Prototype for data transfer control in the Cloud. Technical Report D:D-3.3, Accountability for Cloud and Future Internet Services - A4Cloud Project, 2015.
- [dOSGJ13] Anderson Santana de Oliveira, Jakub Sendor, Alexander Garaga, and Kateline Jenatton. Monitoring personal data transfers in the cloud. *2013 IEEE 5th International Conference on Cloud Computing Technology and Science*, 1:347–354, 2013.
- [dOSoTP⁺15] Anderson Santana de Oliveira, Jakub Sendor, Võ Thành Phúc, Efthymios Vlachos, Monir Azraoui, Kaoutar Elkhyaoui, Melek Önen, Walid Benghabrit, Jean-Claude Royer, Michela D’Errico, Martin Gilje Jaatun, and Inger Anne Tøndel. D:D-3.2: Prototype for accountability enforcement, tools and services. Technical Report D:D-3.2, Accountability for Cloud and Future Internet Services - A4Cloud Project, 2015.
- [EOM14] Kaoutar Elkhyaoui, Melek Önen, and Refik Molva. Privacy Preserving Delegated Word Search in the Cloud. In *SECRYPT 2014, 11th International Conference on Security and Cryptography*, Vienna, Austria, 2014.
- [FHHW11] Simone Fischer-Hübner, Hans Hedbom, and Erik Wästlund. Trust and assurance hci. In Jan Camenisch, Simone Fischer-Hübner, and Kai Rannenberg, editors, *Privacy and Identity Management for Life*, pages 245–260. Springer Berlin Heidelberg, 2011.
- [FHnOP14a] Simone Fischer-Hübner, David Núñez, Melek Önen, and Tobias Pulls. D:D-7.2 Privacy Design Guidelines for Accountability Tools. Project deliverable, A4Cloud Project, May 2014.

- [FHNOP14b] Simone Fischer-Hübner, David Nuñez, Melek Önen, and Tobias Pulls. D:C-7.2 Privacy Design Guidelines for Accountability Tools. Project deliverable D:C-7.2, A4Cloud Project, May 2014.
- [FL11] Daniel Fabbri and Kristen LeFevre. Explanation-based Auditing. *Proc. of the VLDB Endowment*, 5(1):1–12, 2011.
- [Fre00] Linton C. Freeman. Visualizing social networks. *Journal of social structure*, 1(1):4, 2000.
- [GJD11] Deepak Garg, Limin Jia, and Anupam Datta. Policy Auditing Over Incomplete Logs: Theory, Implementation and Applications. In *Proc. of the 18th ACM Conf. on Computer and Communications Security (CCS)*, pages 151–162. ACM, 2011.
- [Gup13] Siddharth Gupta. *Modeling and Detecting Anomalous Topic Access in EMR Audit logs*. Master thesis, University of Illinois at Urbana-Champaign, 2013.
- [HO98] Sharon Kay Heatley and James R. Otto. Data Mining Computer Audit Logs to Detect Computer Misuse. *Intelligent Systems in Accounting, Finance and Management*, 7(3):125–134, 1998.
- [JL09] S. Jarecki and X. Liu. Efficient Oblivious Pseudorandom Function with Applications to Adaptive OT and Secure Computation of Set Intersection. In *Theory of Cryptography*, volume 5444 of *Lecture Notes in Computer Science*, pages 577–594. Springer Berlin Heidelberg, 2009.
- [KMSH12] Sanjay Kairam, Diana MacLean, Manolis Savva, and Jeffrey Heer. Graphprism: compact visualization of network structure. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, pages 498–505. ACM, 2012.
- [KNP10] Jan Kolter, Michael Netter, and Günther Pernul. Visualizing past personal data disclosures. In *Availability, Reliability, and Security, 2010. ARES'10 International Conference on*, pages 131–139. IEEE, 2010.
- [KZH12] Elahe Kani-Zabihi and Martin Helmhout. Increasing service users' privacy awareness by introducing on-line interactive privacy features. In *Information Security Technology for Applications*, pages 131–148. Springer, 2012.
- [Lin15] Daniel Lindegren. Visualisera personuppgifter i ett tidslinjegränssnitt & användning av font awesome-ikoner i a4cloud-projektet. Master's thesis, Karlstad University, Karlstad, Sweden, 2015.
- [MGS10] Erika McCallister, Timothy Grance, and Karen A Scarfone. NIST SP 800-122. Guide to protecting the confidentiality of personally identifiable information (pii). 2010.

- [PM14] Tobias Pulls and Leonardo Martucci. D:D-5.2: User-Centric Transparency Tools V1. Technical Report D:D-5.2, Accountability for Cloud and Future Internet Services - A4Cloud Project, 2014.
- [PP15a] Tobias Pulls and Roel Peeters. Balloon: A forward-secure append-only persistent authenticated data structure. Cryptology ePrint Archive, Report 2015/007, 2015. <http://eprint.iacr.org/2015/007>.
- [PP15b] Tobias Pulls and Roel Peeters. Balloon: A forward-secure append-only persistent authenticated data structure. In *Computer Security-ESORICS 2015*. Springer, 2015. To appear.
- [PP15c] Tobias Pulls and Roel Peeters. Insynd: Privacy-preserving transparency logging using balloons. Cryptology ePrint Archive, Report 2015/150, 2015. <http://eprint.iacr.org/2015/150>.
- [PR04] R. Pagh and F.F. Rodler. Cuckoo hashing. *Journal of Algorithms*, 51(2):122–144, 2004.
- [Pul12] Tobias Pulls. Privacy-friendly cloud storage for the data track - an educational transparency tool. In Audun Jøsang and Bengt Carlsson, editors, *Secure IT Systems - 17th Nordic Conference, NordSec 2012, Karlskrona, Sweden, October 31 - November 2, 2012. Proceedings*, volume 7617 of *Lecture Notes in Computer Science*, pages 231–246. Springer, 2012.
- [Pul15] Tobias Pulls. *Preserving Privacy in Transparency Logging*. PhD thesis, Karlstad University, Department of Mathematics and Computer Science, 2015.
- [RPR15] Thomas Rübsamen, Tobias Pulls, and Christoph Reich. Secure Evidence Collection and Storage for Cloud Accountability Audits. In *CLOSER 2015 - Proceedings of the 5th International Conference on Cloud Computing and Services Science, Lisbon, Portugal, May 20 - 22, 2015*. to appear, 2015.
- [RR13] T. Rübsamen and C. Reich. Supporting cloud accountability by collecting evidence using audit agents. In *Cloud Computing Technology and Science (Cloud-Com), 2013 IEEE 5th International Conference on*, volume 1, pages 185–190, Dec 2013.
- [RRW⁺15] Christoph Reich, Thomas Rübsamen, Tomasz Wiktor Wlodarczyk, Rui Pais, Monir Azraoui, Melek Önen, Jenni Reuben, Tobias Pulls, Mohamed Sellami, Jean-Claude Royer, Massimo Felici, and Karin Bernsmed. D:C-8.3 Automation Service for the Framework of Evidence. Project deliverable D:C-8.3, A4Cloud Project, March 2015.
- [SMP08] Dieter Sommer, Marco Casassa Mont, and Siani Pearson. Prime architecture v3, July 2008.

- [TP10] J. Trostle and A. Parrish. Efficient Computationally Private Information Retrieval from Anonymity or Trapdoor Groups. In *Proceedings of Conference on Information Security*, pages 114–128, Boca Raton, USA, 2010.
- [VOE04] Hein S. Venter, Martin S. Olivier, and Jan H.P. Eloff. Pids: A Privacy Intrusion Detection System. *Internet Research*, 14(5):360–365, 2004.
- [WP14] Tomasz Wiktor Wlodarczyk and Rui Pais. D:C-8.1: Framework of Evidence. Technical Report D:C-8.1, Accountability for Cloud and Future Internet Services - A4Cloud Project, 2014.
- [WRR⁺15] Tomasz Wiktor Wlodarczyk, Rui Pais Thomas Rübsamen, Christoph Reich, Monir Azraoui, Jean-Claude Royer, Jenni Reuben, Tobias Pulls, Karin Bernsmed, and Massimo Felici. D:C-8.2 Framework of evidence. Project deliverable D:C-8.2, A4Cloud Project, March 2015.

A. Mapping Graphical Icons to Personal Attributes

As part of the design and development of the Data Track (see Section 4) we saw the need of having a consistent set of graphical icons that represented personal data attributes. Having personal attributes as small icons would improve the scalability of the design which might require the display of many attributes in limited screen real state. To this end, we created a preliminary mapping of the vector-based standard icons provided by the open-source library Font-Awesome. Table 2 shows the mapping of common personal attributes to existing Font-Awesome icon codes. More information about evaluation of this mapping can be found in [Lin15].

Personal attribute	Font-Awesome code	Category	Personal attribute	Font-Awesome code	Category
User name/ID	fa-user	profile	Hotel	fa-bed	travel
Password	fa-asterisk	profile	Car	fa-car	travel
Profile picture	fa-picture-o	profile	Pet	fa-paw	personal
Email	fa-envelope	profile	Comments	fa-comments	socialmedia
Address	fa-home	profile	Post	fa-comment	socialmedia
Phone number	fa-phone	profile	Time	fa-hourglass-end	
Mobile phone	fa-mobile	profile	Relationship status	fa-heart	personal
Credit Card	fa-credit-card	financial	Played song	fa-music	media
First name	fa-male	personal	Played video	fa-film	media
Last name	fa-male	personal	Video	fa-file-video-o	media
Gender	fa-venus-mars	personal	File metadata	fa-file-code-o	document
Age	fa-calendar	personal	Tablet type	fa-tablet	device
Life attitude	fa-hand-peace-o	personal	Device	fa-mobile	device
Birthplace/Nationality	fa-flag	personal	Created an album	fa-file-image-o	media
Country of residence	fa-globe	personal	Camera brand	fa-camera	media
Birthdate	fa-birthday-cake	personal	Voice recording	fa-microphone	media
Education/degree	fa-graduation-cap	professional	Tagging a content	fa-tag	metadata
Profession	fa-briefcase	professional	Content list of tags	fa-tags	metadata
Job	fa-briefcase	professional	Likes	fa-thumbs-o-up	socialmedia
Employer	fa-building	professional	Mood	fa-smile-o	personal
Affiliation	fa-institution	professional	Alcohol level	fa-glass	personal
Income	fa-dollar	financial	Medical condition	fa-medkit	medical
Savings	fa-money	financial	Medical record	fa-hospital-o	medical
Coordinates	fa-location-arrow	location	Physical activity	fa-soccer-ball-o	medical
Location	fa-map-marker	location	Calories	fa-cutlery	medical
Direction	fa-compass	location	Medical item	fa-stethoscope	medical
Speed	fa-angle-double-right	location	Deletion request	fa-trash-o	accountactivity
Calendar entry	fa-calendar-o	lifeevent	Sign in request	fa-sign-in	accountactivity
School	fa-university	professional	Sign out request	fa-sign-out	accountactivity
Search history	fa-history	profile	Edit request	fa-edit	accountactivity
Operating system	fa-desktop	device	User preferences	fa-gear	accountactivity
Latest song played	fa-music	media	Customer rating	fa-star-half-full	profile
Technical literacy	fa-code	professional	Shopping habits	fa-cart-arrow-down	shopping
User list of friends	fa-group	socialmedia	Events attended	fa-ticket	lifeevent
Mother tongue	fa-language	personal	Security level	fa-unlock-alt	security / privacy
Second language	fa-language	personal	Account type	fa-diamond	accountactivity
Retweet	fa-retweet	socialmedia	Audience	fa-users	socialmedia
Added friend	fa-user-plus	socialmedia	IP address	fa-flag-checkered	device
Heartbeat	fa-heartbeat	medical	TV type	fa-tv	device
Gender	fa-venus-mars	profile	Copyright license	fa-copyright	media
HasChildren	fa-child	personal	Shared with	fa-share-alt	socialmedia
Bank name	fa-bank	personal	Mouse clicks	fa-mouse-pointer	accountactivity
Secret	fa-user-secret	personal	Views	fa-eye	accountactivity

Table 2: The listed personal attributes are represented by corresponding icons from the Font-Awesome library and assigned a category.