Literature Review

Data rules and consent modeling

Rui Zhao

September 24, 2018

Abstract

The large-scale collaboration and data-intensive research makes researchers difficult to manage data governance rules. The fact that such rules are heterogeneous and written in natural language prohibits the automation of compliance check. Some research focuses on modeling parts of the rules, especially access control, although none of them cover the area. This document present our review of the existing research, as well as a comparison among several research (especially data-flow tracking ones).

1 Introduction

With today's large scale of collaboration of research and productivity, different groups (people, organizations, companies) face a different subset of problems. Processing and storing of data are well digitized and supported, but the sharing of data is not well automated. This prohibits the prospected future that autonomous agencies are widely used to support collaborations while human researchers can focus on research only. One of the main reasons why sharing of data is not well automated is the heterogeneity of regulations, laws and consents; the second main reason is that they are not computer understandable.

Privacy and compliance (of regulations, laws, etc.) are important issues for people, organizations and companies. From one perspective, there is a conflict between people's intention of preserving their privacy (e.g. address, ID number, personal interests, etc) and companies' intention of collecting more customer data which often includes confidential information for better business, e.g. advertising. From another perspective, companies tend to maximize the utilization of the customer data they collected, but do not wish to

leak the data to other companies especially competing companies, even when they need to cooperate with other companies to conduct the processing and analysis of data. Even in some research context (e.g. online social network research [L. Hutton and T. Henderson 2018]), sharing data may also contribute much more compared to sharing methods and code, but the shared data must not violate the consent of the participants. They all demonstrate the importance of sharing data, the importance of protecting restricted data, and therefore the importance of policies of data usage.

However, different bodies have different requirements for using and sharing data, so they set up different rules focused on their needs. Regulations and laws are set up governing data usage as a basic requirement, but the fact that different regions have different laws or regulations increases the heterogeneity. In some cases, such rules may conflict with each other, making the rules incompatible, making the collaboration impossible. However, because the rules are written in natural languages, checking the compatibility takes much effort and incorrect interpretation of the combination may become a problem. Similarly, the modification of rules may cost more effort to consider the side-effects. Therefore, organizations tend to grant themselves overwhelming permissions of data usage and then carefully use the data within their own controlled context.

The situation leads to people not having proper knowledge of how their data will be used, researchers spending hard time manually complying with rules, and data collectors spending much manpower to check and ensure the compliance of rules. Therefore, we expect to provide a computer-interpretable rule language and let intelligent systems do the heavy work to improve the situation. Facing the fact that there is not yet a single standard to represent and interpret regulation terms in a computer-interpretable way, we did some reading of existing modeling of rules. Some research have been done regarding different aspects of rules and compliance in information systems, e.g. case study, modeling, enforcement, etc. Some on-going research (e.g. DL4LD[1], EOSC[2]) also tries to provide infrastructures or systems to automate the compliance checks. In this document, we try to classify them according to the types of question they are trying so solve.

The rest of this document is structured as follows: we first introduce the research scope; then we describe each typical field of research and typical work, introducing their concepts; then we make comparisons among them as our evaluation, in different themes.

[1] Laat, Cees de [2018]. *DL4LD*. URL: `http://www.delaat.net/dl4ld/` [visited on 04/16/2018].

[2] *European Open Science Cloud (EOSC) — Open Science - Research and Innovation - European Commission* [2018]. URL: `https://ec.europa.eu/...` [visited on 07/31/2018].

2 Research Scope

The statement "data rules modeling" or "privacy and compliance in information systems" itself doesn't have a clear bound of the research scope, but the scope we explored does have a bound. In this section, we present the scopes of research reviewed.

Generally, we started by exploring materials which model privacy policies, and expanded to various different types of things naming themselves "policies", "rules" and "obligations" as well as "requirements". These terms are usually mixed, and the vast majority of research we found labeled themselves "policies", usually talking about access and processing permissions.

As a result, the research we found covers a wide range of cases and they are located in different scopes:

- from single-machine data to machines connected by Internet;
- from carefully controlled range (usually inside one institution) to arbitrarily crossing boarders;
- from service-collected information to all kinds of information;
- from one-time access-control to rules attached to data;
- from control-only policies to more general obligations;
- from consent creations to the modeling after rules are specified.

Despite all the differences, our main concern is the modeling of rules, especially from the data-flow point of view, and automation. Research focusing on single institutional data and one-time access-control is considered as non-data-flow; research focusing on single-machine data can be either non-data-flow or data-flow, depending on the previous point; research concerning data across boarder is very likely in data-flow; the information type does not affect whether the research is about data-flow, but they affect the modeling target and therefore modeling method; the research about negotiation of consents is not about modeling itself, but about the modeling target and may also affect the modification of rules. In this document, they are clustered in four topics, which will be introduced in the next section.

The main part of this document will focus on languages describing rules and the surrounding systems. Some research provides languages not talking about data usage rules, e.g. ConSpec [Aktug and Naliuka 2008] talks about constraints on programs, but we see them containing useful information or insights for current or future languages for describing data usage rules. Moreover, our main focus is the data-flow tracking systems.

Nevertheless, despite trying to cover the area, because of the lack of naming consensus, we may have missed useful research. Research solely talking about representations of obligations (e.g. [Elrakaiby et al. 2012]) is

not part of the main content of this document, although they may be useful for extending policy languages. Because of the time limitation, we didn't fully read some research talking about logic used in policy languages (e.g. [Ni et al. 2007]), as we consider them less relevant after brief reading. However, we may revisit them as the logic they used could provide insights in the design of future languages.

3 Current Research

The researches can be, not strictly, clustered into four topics:

1. Consent establishment
   this topic covers research talking about the establishment process or methods of obtaining data subjects' consent
2. One-time restriction
   this topic covers research focusing on restricting data access only and do not consider rules afterwards
3. Data-flow tracking
   this topic covers research tracking the flow of rules together with dataflow
4. Deployment policy
   this topic covers policies of cloud service deployment

Topic 1 is about the phase before rules are established, focusing on the requirements and types of things rules deal with. Topic 2 and 3 focus on policies governing the use of data, while the 4th one focuses on how and where the execution (or steps of the execution) happen. We focus on topic 3 which aligns most with our goal.

It is worth mentioning here that we intend to use "consent", "policy", "rule" and "obligation" under the following definitions:

- **Policy** is the requirements for data access or processing, usually in the form of permissions;
- **Obligation** is the requirements that the data user needs to satisfy before or (especially) after using the data;
- **Consent** is the agreement between the data provider and the data user;
- **Rule** is the general term for all kinds of constraints on data usage, including **policy** and **obligation**.

3.1 Consent negotiation

Research reviewed in this section has a different target compared to the rest of this section: the process and requirement of how rules or consents are created, involving the automated negotiation of consent and the discussion of people's willingness to give consent. They do not directly provide the modeling of rules, but provide the possibilities of the content described in the rules. Therefore, we briefly introduce them and their relation to the general theme (modeling rules), but will neither focus nor evaluate on them in this document.

One view of privacy and user data is to see them as a "valuable resource" and "trade" them. Under this perspective, a user who holds protected data *is* willing to "sell" their data given a sufficient "price". Baarslag and Kaisers [2017], Baarslag, Alan, et al. [2017], and Aydoğan et al. [2017] held this kinds of view and developed approaches for an autonomous agent to negotiate with the user about the consent, given the user's preference. The details of these research does not align with the topic of this document, but generally they view the negotiation process as the trade-off between the user (who holds the privacy and permission) and the agent (which wants to obtain privacy and permission as much as it wants), using a cost function to optimize the balance between the user and the agent.

The previous mentioned research captured one important aspect: people's view of privacy and personal data is neither homogeneous nor static - negotiation for trust and consent may enable the use of protected data or open new methods of use. Some study and research (e.g. Luger and Rodden [2013], Gomer et al. [2014], and Luke Hutton and Tristan Henderson [2017]) dug further into how people/users are willing to share and allow institutions to use their data: Luger and Rodden [2013] presented several interviews with experts in different domains on various aspects of consent (acquiring, handling, negotiating, etc); Gomer et al. [2014] discussed the problems in obtaining consent and automation, and briefly discussed and compared their semi-autonomous consent agents model and two other such models; Luke Hutton and Tristan Henderson [2017] presented two cases about Facebook and one case about NHS sharing data with Google, and concluded that the current method of data-sharing didn't really involve consent from the data subjects.

3.2 One-time Restriction

Applying the data governance rules is always done in some form of data access control. An easy-to-understood way of controlling the use of data is through

the control of initial data access. This is usually employed in an enterprise context, where the company or institution which collected customer data runs the data processing in a controlled-by-itself way, i.e. either to process everything within the institution or to cooperate with a "trusted" partner.

In this part, we introduce three typical views under this topic, and enlighten the transformation to data-flow tracking systems. It is worth noticing that, because of the context, the following assumptions are usually made:

1. the institution can have full knowledge and control of its members' permission
2. the institution has pre-specified and recognized rights that allow it to process such data, and the permission is unlikely to change frequently
3. to share data, the institution ought to have a legal binding contract to the partner, which in theory effectively limits the partner's behaviour

Research in this scope usually takes these assumptions for granted, so the developed solutions don't have any concerns about where the policies come from, how to persuade every individual to use the system or how to protect the data from being copied out of the system. They implicitly or explicitly assume the system is well-designed and every piece of process is done through the system.

In 2002, W3C agreed a standard called Platform for Privacy Preferences (P3P), which was deprecated a few years later, and there is no new standard emerging. However, since its existence, several researches tried to tackle P3P's weakness to produce better (formal) representations and enable machine understanding.

E-P3P [Karjoth et al. 2002] is one of the earliest attempts we found to try to address policy issues – it extended some concepts of P3P and presented Platform for Enterprise Privacy Practices (E-P3P), to model the privacy statements in a "formal" way. It presented a framework to check the compliance of rules (especially policies) and some predicates. The rules mainly talked about the access control and privacy control, through describing data users, purposes (of using data), operations on the data and obligations. However, the expressivity is restricted to only the predicates identified and encoded in the system, which limits the scalability and flexibility. Although this method can be extended to do policy checking along with data, this is not developed in this paper. Mont et al. [2003] first presented a solution to this, which is often cited as the foundation of *sticky policy* (though this term is already used in Karjoth et al. [2002]). However, the research didn't seem to catch much attention during 2003, but regained attention several years later, and was re-described in Pearson and Casassa-Mont [2011] (explained further in 3.3).

The work by Baumann et al. [2017] presented a way of using centralized data storage to confine the data access policies, in the industry 4.0 theme. This work uses a concept of "Data Hub" which handles policy and approves data access. The Data Hub is not a central single storage, but rather, it is a concept and mechanism that any data storage (regardless of the data format) can act as, and any Data Hub can federate with each other to propagate the access and policy checking. However, the paper doesn't describe in depth how they organize and encode the rules, nor does the website (not in English). Moreover, although Baumann et al. mentioned "transitivity of rights" raised in [Zhao et al. 2010], a paper describing an algorithm and protocol for securely sharing data over network to untrusted storage providers, they didn't really "track" data – the Data Hub only checks the use of data once, but don't keep the policy for the produced data, as far as we understand. Thus this work still lies in this topic.

Compliance Rule	**Description**
$DO.type == T$	Type of data-object
$DO.range == R$	Range of data-object
$DO.loation == L$	Data location
$A.datainput == D$	Restricted data input of activity
$DO.writes <= 1$	Avoid Race Conditions
$A1.input == A2.input$	Use of same data
$A.assignee == P1$	Assignment of a Person to an Activity

Table 1: Data-related compliance rules (taken from [Schleicher et al. 2011])

In a different field, Schleicher et al. [2011] presented a method to encode data-related rules (not necessarily policies) into the business process and a checking system. Their view considered data rules as some static manually-defined relations and atomic variable/parameters (see Table 1). However, because this work is developed for business process modeling, the types of rules are very limited and the focus is to check the availability of the (process) model. They also did not pay any attention to the authoring of rules or the compliance of a deployed system.

3.3 Data-flow tracking

An alternate view is to see the use of data as a continuous process where every piece of output data may be reused later. Therefore, how data flow through the computation should be taken into account. As a result, research in this topic considers both the rules (mainly access-control policies), and the change of the level of "accessibility" during the computation.

In this part, we introduce four different views of data-flow tracking compliance systems, each by one typical research. More relevant research will also be described inside each view, as well as the high-level description of their specific views, assumptions, features and limitations. The details will be discussed and compared in Section 4.

A method was presented by Håvard D. Johansen et al. [2015] which associated policies with data and made the policies flow (and change) with the computation. Their method considers data policies as the combination of 1. "roles permitted to access" the data; 2. "meta-code" which is checked before accessing the data; 3. resulting policies after performing certain operations (e.g. processed by certain groups of software). All these metadata are associated with the data (file), and will be carried through the computation; permissions are based on groups, and a trusted institution is required to associate software with the groups, which enables the change of permissions after processing; access permissions are only boolean, but in theory the meta-code can execute arbitrary code. Their solution relies on operating system and filesystem (through *extended file attributes* available in modern filesystems on Linux) and their experiment was carried on a single machine. The same group did other work, e.g. H. D. Johansen et al. [2014] and A. T. Gjerdrum et al. [2016], relying on the same *meta-code* concept to control the access. Specifically, H. D. Johansen et al. [2014] also takes the internal properties of data (e.g. the sampling rate) into account, as part of the access controls; A. T. Gjerdrum et al. [2016] extends both Håvard D. Johansen et al. [2015] and H. D. Johansen et al. [2014] and provides more examples of the form of data policies.

Interestingly, H. D. Johansen et al. [2014] also provided a method to check the eligibility of data provided (i.e. whether the provided data is eligible to be used in some project) without actually revealing the data, by checking the properties of the data (represented in a structured computer-readable way) against the criteria of the project (represented in a logic form). H. D. Johansen et al. [2014] represented the properties of data in a tuple containing data source, time and time-span of the data in addition to the data; the criteria are represented as the logic ($\wedge$,$\vee$,$\neg$) combination of filter expressions on each of the properties. This method is similar to the work by Trani et al. [2017] in the way that they both try to minimize the data collection / transportation by checking the data "quality" beforehand. Although not mentioned in the work, this view is similar to limiting the data access using logic and expressions.

Thoth [Elnogomi et al. 2016] uses logic and expressions to deal with data policies, implemented surrounding a Linux Security Module, and it is extended (optimized) further in their following work, SHAI [Eslam Elnikety

Arithmetic/string		Conduit		Content	
add(x,y,z)	x=y+z	cNameIs(x)	x is the conduit pathname	(c,off) says	$x_1,\ldots,x_n$ is the tuple found in
sub(x,y,z)	x=y-z	cIdIs(x)	x is the conduit id	$(x_1,\ldots,x_n)$	conduit c at off
mul(x,y,z)	x=y*z	cIdExists(x)	x is a valid conduit id	(c,off) willsay	ditto for the update of c in the
div(x,y,z)	x=y/z	cCurrLenIs(x)	x is the conduit length	$(x_1,\ldots,x_n)$	current transaction
rem(x,y,z)	x=y%z	cNewLenIs(x)	x is the new conduit length	each in (c,off) says	for each tuple in c at off, assign
concat(x,y)	x ‖ y	hasPol(c, p)	p is conduit c's policy	$(x_1,..,x_n)$ {condition}	to $x_1,..,x_n$ and evaluate condition
vType(x,y)	is x of type y?	cIsIntrinsic	does this conduit connect two confined processes?	each in (c,off) willsay $(x_1,..,x_n)$ {condition}	ditto for the update of c in the current transaction
Relational		**Session**		**Declassification rules**	
eq(x,y)	x=y	sKeyIs(x)	x is the session's authentication key	c1 until c2	condition c1 must hold on the downstream flow until c2 holds
neq(x,y)	x!=y				
lt(x,y)	x<y	sIpIs(x)	x is the session's source IP address	isAsRestrictive(p1,p2)	the permission p1 is at least as restrictive as p2
gt(x,y)	x>y				
le(x,y)	x<=y	IpPrefix(x,y)	x is IP prefix of y		
ge(x,y)	x>=y	timeIs(t)	t is the current time		

Table 2: Thoth policy language predicates and connectives (taken from [Elnogomi et al. 2016])

et al. 2018], by static analysis. Their way of representing the policies involve more logic inside (Table 2), which has both benefits and limitations:

- On the one hand, the logic allows the policy to construct complex sentences using a few predicates.
- On the other hand, the number of predicates becomes the new limit of the expressive power.

A different class of concept is *sticky policy* [Pearson and Casassa-Mont 2011; Mont et al. 2003], extending the previously mentioned research by Karjoth et al. [2002] which only works inside an enterprise boundary. This concept considers the data as a whole unit and tries to "control" where the data is going to by selecting the acceptable requestors. They provide a mechanism to try to address the enforcement of policy: 1. encryption is applied to the data and the *sticky policy* comes along with the data; 2. the data requestor can only obtain the decrypted data if the "Tracing Authority" (e.g. trusted third-parties or the data owner) grants trust to the requestor. The trust granting process depends on the TA, which may be manually or automatically.

CamFlow [Pasquier et al. 2017] uses Information Flow Control (IFC) to manage policies in the system kernel level, targeted at the cloud environment at PaaS. CamFlow has the mechanism (CamFlow-MW) to deal with data flowing across machine boarder, given CamFlow is supported on the machines. However, its way of representing policies is limited (using tags and labels), which is similar to the **meta-code** approach [Håvard D. Johansen et al. 2015].

3.4 Deployment Policy

Deployment policies are the policies directing how, where and when the deployment (of execution) and/or the execution should happen (e.g. location restriction). Although not entirely the same, the deployment policies are also relevant to data policies. Therefore, we also take them into account.

In this part, some relevant work is introduced, as well as their relationship to data rules. TOSCA is a general predecessor work for many research, so it is introduced first, followed by work built on it. Finally, a different research is introduced because of its usefulness for the future. Note the research in this topic usually uses *cloud orchestration* and *deployment*, which has the similar meaning to *workflow construction* and *execution* but emphasising the cloud environment.

TOSCA[3] is a standard for describing cloud applications for automated orchestration, and there are open runtime framework implementations of it (e.g. [Binz et al. 2013]). Although developed for cloud orchestration, the use of TOSCA doesn't stop within this area only (e.g. [Wettinger et al. 2014] used TOSCA to help DevOps, by mapping to DevOps tools). The description language of TOSCA has a section dedicated to "policies", and research on this aspect are the main concern of this document.

A group of researchers in University of Stuttgart did much work (e.g. [Waizenegger et al. 2013; Breitenbucher, Binz, Kopp, et al. 2013; Breitenbucher, Binz, Fehling, et al. 2014; Fischer et al. 2017]) about cloud orchestration, provisioning and management. Provisioning is outside the scope of this document, so the related papers are not detailed. In [Waizenegger et al. 2013], a mechanism was presented to describe (within the framework of TOSCA) policies and an example system capable to execute the policies, mainly about requirements like encryption, location, etc. Later, [Breitenbucher, Binz, Fehling, et al. 2014] was presented for the *provisioning* (i.e. the work before executing computation) of cloud applications, based on their previous *Policy-Aware Management Planlets* [Breitenbucher, Binz, Kopp, et al. 2013] and other work. Breitenbucher, Binz, Fehling, et al. [2014] stated that [Waizenegger et al. 2013] requires manual construction of policies, which is time consuming and error prone.

MiCADO [Kiss et al. 2017] is a recent "overall" approach for a solution of runtime cloud orchestration and policy management based on TOSCA description language. Its ability of handling policies is further described in []. [HERE SHOULD BE THE IWSG 2018 PAPER MALCOLM SENT ME]

Despite all the differences, the "policy" in the cloud orchestration theme mainly refers to the requirements of execution and/or deployment. The most

[3] *TOSCA v1.0 Committee Specification 01 published by the TOSCA TC — OASIS* 2013.

relevant policy (to this document) is the region restriction which limits the possible geographical locations of where execution should happen (as a result of limiting the allowed transmitted region of data).

On the other hand, researches like [Zhao et al. 2010] try to address the privacy issue by presenting better protocol for sending encrypted data and keys. Although they are not directly related to this document, a protocol will be eventually needed if policies are going to be used and enforced.

4 Evaluation

In this section, we compare the core concepts of the previous mentioned research and evaluate the effort needed in setting up the rules and contexts in which they apply for each system. The research for data-flow tracking is the main focus of evaluation, but other work is also mentioned where relevant.

We first introduce the scope of evaluation, followed by an overview by examples which demonstrates the capability and expression of each approach. After that, we evaluate the research presented above in different topics, each evaluating one aspect of the design of the system / framework.

Tables 3, 4, 5, 6 summarize the comparison of a few of the most important aspects of the different rule languages. Each table usually contains the items discussed in a different segment of this section; but this is not strict as some features are cross-boarder, e.g. "expressiveness of rules". The reason of presenting them here is to give interested readers a preview of the important aspects.

Framework	Data		Processor
	Read	Update	
CamFlow	Yes	No	Yes
Meta-code (LoNet)	Yes	No	Yes
Thoth	Yes	Yes	No
Sticky policy	Yes	?	?

Table 3: Available annotation targets of rules

What component can have annotations (i.e. part of the rules) and then handled by the system. Note the detailed capability still vary depending on the approach.

Framework	Rule originator	Specify trust in rules
CamFlow	Collaboratively	No
Meta-code (LoNet)	Collaboratively	No
Thoth	Data owner	Yes
Sticky policy	Data owner	Through TA

Table 4: Rule originator and trust handling

Who designs the rules or policies and how does the framework treat whom to trust. TA means Trusted Authority, as described in the sticky policy paper [Pearson and Casassa-Mont 2011].

Framework	Work level	Required change of application
CamFlow	OS (LSM)	No
Meta-code (LoNet)	OS (FUSE)	Annotation if necessary
Thoth	OS (LSM)	No
Sticky policy	Application	Human interpolation

Table 5: System and annotation property

Which level does the system work, and what extra cost to existing applications shall present when using the system. LSM is short for Linux Security Module; FUSE is short for Filesystem in Userspace.

4.1 Aspect of evaluation

We introduce the possible evaluation aspects here, and highlight the aspects we evaluate on as well as the reason.

Introducing an additional layer upon systems always introduces overheads which can contain various aspects. Generally speaking, the overheads manifest in four sections:

1. human cost;
2. data volume;
3. processing overheads;
4. system complexity.

The human cost is due to the extra annotations of rules to the data and sys-

Framework	Rule Basis	Group permission	Change with flow	Static analysis
CamFlow	IFC	No	Yes	Yes
Meta-code (LoNet)	IFC + any code	Yes	Yes	Yes + No
Thoth	logic	Yes	Yes	Yes (in SHAI)
Sticky policy	Non-specified	Depends on policy language		

Table 6: Rule language features

Some important aspects of the features of different rules languages. IFC is short for Information Flow Control (described in [Pasquier et al. 2017]; SHAI is an extension of Thoth which mainly presents the static analysis [Eslam Elnikety et al. 2018].

tem; these annotations increase the data volume (in the metadata); to check the rules, the processing overhead is introduced; the extra efforts needed to integrate the proposed systems into existing systems increases the system complexity. We evaluate the human cost by comparing the complexity of rules. The rest of the overheads lacks the standard of comparison or is not detailed in the literature, so they are not considered. But generally, as summarized in Table 5, the data-flow tracking systems introduce another layer for checking rules, requiring no change to existing software, but only effort in annotating software.

The following characteristics are considered as the issues to assess:

1. assumptions made;
2. the kinds of rules supported;
3. how to deal with the infringement of rules;
4. requirements after rules are set up.

The assumptions made are described in 4.3; the kinds of rules supported are described in 4.4; the requirements after rules are set up are discussed as two topics, in 4.5 and 4.6. The method of how to deal with the infringement of rules is not discussed dedicated, but is inherited from the organizational structure and therefore can be changed with extension (between halting execution and alerting the administrator); however, the capability of detecting infringement of rules is determined by the kinds of rules supported, and is therefore contained in 4.4.

4.2 Overview by example

We present a few examples demonstrating the encoded form (of the same rule) in different systems in this part. We only consider the data-flow tracking systems which explicitly expressed their rule languages, because they are most closely related to our research.

The aim of this section is not to provide a full demonstration towards the capability of the languages in different systems, which will be discussed later. Instead, the aim here is to provide a glance of the effort, length and representation of expressing some rules in different systems. As a result, we only explain important aspects critical to understanding the expressions; please refer to the original papers for full details and explanations if needed.

Although these data-flow tracking systems are mostly related to our topic, they still diverge from our topic. Because of the differentiation in the design goals between their research and ours, the rules they can express differ from what we aim to support. Therefore, in this part, we only consider some rules that can be expressed in every system, and leave the discussion of their capabilities to later. They can express only limited abstraction of rules, so the examples are also in more specific forms.

Allow certain groups of people to access

A very simple yet important rule is to restrict the access of data to certain people, because, for example, the data contains sensitive information. It can be considered the basic capability of a rule framework, but not an advanced feature. We can consider a rule sentence "every piece of data belonging to *Alice* can only be accessed by people with role *medic*". Here, *Alice* is merely a code name: it can be a single person or a group of people (e.g. the "patient" group).

Because not all systems can directly specify "data categories" (but can, instead, associate the same rule with every data within the category), we simply the rule to "*this data file* belongs to *Alice* and can only be accessed by people with role *medic*" for systems not capable to represent data categories.

- Thoth: Thoth does not allow people to express groups in the language demonstrated. Instead, one can do traversal of a (plain) file and check the content by lines. We assume the check of accessor's identity is based on (public and secret) keys, which is one type supported by Thoth.
  - list every member of the group *medic* in a file / location called, e.g., *Alice.acl*
    - we define the content of the file is composed of multiple entries of format isMedic(k_X), where k_X is medic X's pubkey

  - Write the following rule to the data file:
    read :- sKeyIs(k_X) $\wedge$ ("Alice.acl", off) says isMedic(k_X)
  - the sKeyIs(k_X) predicate means public key k_X authenticates the current session
  - the ("Alice.acl", off) says isMedic(k_X) predicate means to loop the file *Alice.acl* and find a line matching isMedic(k_X) where k_X is the same k_X as the previous one; off represents the offset of the match, which is no used in this example

- CamFlow: CamFlow can specify the category of the data (through the S label group), but it can not directly specify access permission by group. However, we can assume, enforced by some other mechanisms, only people in the *medic* group are allowed to use certain processes or software. Therefore, we can achieve the rule by:
  (note we can start at step three and ignore the previous steps, because those steps are not directly said in the rule; however, we demonstrate this process here because this method is also used in the next rule; the examples below follow the same convention)
  - Initially data is accessible only to *Alice*
    * $S(A) = \{alice\}$ & $I(A) = \emptyset$
  - Define a process (call it consent checker) with the following property for *Alice* which ensures consent (i.e. a software or human process which ensures that *Alice* has given consent to use data as *personal* data)
    * $S = \{alice\}$ & $I = \emptyset \implies S = \{personal\}$ & $I = \{consent\}$
  - Ensure (outside of CamFlow) only *medic* group is allowed to execute processes with the following input tags
    * $S = \{personal\}$ & $I = \{consent\}$

- Meta-code: Meta-code has an explicit expression for *roles*. The way to achieve the goal is to define a role *medic*, and have a few relevant policies concerning data (defining permissions) and processes (`ttype`). The goal is to define a legal route to create a file derived from the original file (which is accessible only to *Alice*) and to automatically assign the new file with read permission to group *medic*
  - Initially, specify in, e.g. `/pol/priv`, that data is not accessible for anyone (except for the owner, *Alice*); transition (`ttype`) *project* will changes the access policy to `/pol/project`

```
[permissions]
roles = {alice}
```

```
[transitions]
project=/pol/project
```

- Specify in the `/pol/project` policy file that *medic* is allowed to access the data, given they satisfy the metacode in `/code/sample_metacode`

```
[permissions]
roles = {medic}
[metacode]
onAccess=/code/sample_metacode
[transitions]
...
```

- In the place for specifying `ttype` and programme, defined by the meta-code system, specify any process considered suitable to perform *project* `ttype`, e.g.

```
[cat]
exe_path = /bin/cat
ttype = project
```

- These in total define a possible route to change the access permission of (the content of) a file from the owner only to everyone in the *medic* group, through executing the relevant software with *project* `ttype` and producing a new output data file with the new access permission

It is worth noticing that for CamFlow and Meta-code, the change of labels or policies is explicitly defined through annotating processes; however, for Thoth, there is no modeling done on the processes. CamFlow does not require the policy for a file to know anything about policy change in advance, while Meta-code requires the policy file to list every potential policy change (`ttype`) and the new policy file.

The difference does not make a huge impact for this simple example, but will differ a lot in the next example.

Permission delegation

For some cases, one may want to delegate his/her permission of reading some dataset to someone else. For example, a university will usually allow research staff and students to use datasets it has access to; the data provider only speaks to the university but not every individual in the university.

The rule can, sometimes, be observed as "you grant us the permission to share your data with carefully selected partners" in the *Terms and Conditions*

on services or artifacts. We take the following rule as the example: "every data belonging to *Alice* can be accessed by every *hospital* and every *medic* within each *hospital* can also access the data". The important point is that each *hospital* does not know the members (*medics*) of other *hospitals*.

- Thoth: Similar to the previous example, but more complex logic shall be used
  - list every *hospital* in a file called, e.g., hospital.list with format isHospital(k_X, X_{demic}) where k_X represents the pubkey of *hospital* X and X_{medic} is the name of the file containing all *medic* within *hospital* X
  - every *hospital* lists their *medics* in their separate files, with format isMedic(k_X) for each file
  - define the rule:

    read :- sKeyIs(k_M)
    $\wedge$ ("hospital.list", off$_1$) says isHospital(k_H, X_{medic})
    $\wedge$ (X_{medic}, off$_2$) says isMedic(k_M)

  - The rule can be interpreted as "store the current pubkey (which authenticates the session) to variable k_M; loop the file hospital.list and decompose every line into the form isHospital(k_H, X_{medic}) and store the offset to off$_1$ (which is ignored); for every pair k_H and X_{medic}, loop the file X_{medic} and try to find a line matching isMedic(k_M) where k_M is the same k_M used to authenticate the session"
- CamFlow: Similar to the previous example, we assume the permission of executing certain processes is enforced by some other mechanisms. We define a route for the *hospital* to transform the data to allow *medic* within each *hospital* to access.
  - suppose we already have a concent checker which can do $S = \{alice\}$ & $I = \emptyset \implies S = \{personal\}$ & $I = \{consent\}$, as in the previous example; only *hospital* can execute this process (i.e. concent checker)
  - define several processes for each *hospital* (call it hospitalX) with the following signature

    $S = \{personal\}$ & $I = \{consent\} \implies$
    $S = \{patient.hospitalX\}$ & $I = \{consent\}$

  - for each hospital, define processes that can consume data with labels $S = \{patient.hospitalX\}$ & $I = \{consent\}$ and (as in the

previous example) restrict only *medics* (of that hospital) can execute these processes outside of CamFlow

- Meta-code: Just like in CamFlow, a set of roles for *medics* of different *hospitals* shall be defined, as well as corresponding `ttype`(s).
  - the method is similar to the previous Meta-code example, but with one more stage (for each of the *hospitals*. The first part is almost the same as the previous example: two policy files for original data and one `ttype` for transition from *Alice*-only to *hospital*
    * Original data policy

      ```
      [permissions]
      roles = {alice}
      [transitions]
      patient=/pol/patient-data
      ```

    * Policy to allow hospitals to access (suppose the filename is /pol/patient-data)

      ```
      [permissions]
      roles = {hospital}
      [metacode]
      onAccess=/code/sample_metacode
      [transitions]
      ...
      ```

    * specify the `ttype` in the relevant location defined by the framework

      ```
      [cat]
      exe_path = /bin/cat
      ttype = patient
      ```

  - then for each *hospital* (hospitalX), do the following
    * define `ttype` hospitalX in the relevant location specified by the framework

      ```
      [cat_hospitalX]
      exe_path = /bin/cat
      ttype = hospitalX
      ```

    * define a new policy file to allow *medic* in this *hospital* to access (suppose the filename is /pol/hospitalX-medic)

      ```
      [permissions]
      roles = {medic-hospitalX}
      ```

```
[metacode]
onAccess=/code/sample_metacode_hospitalX
[transitions]
...
```

* add the `ttype` hospitalX to /pol/patient (the previous policy file for data accessible to *hospital*), and specify the new policy as /pol/hospitalX-medic

```
[permissions]
roles = {hospital}
[metacode]
onAccess=/code/sample_metacode
[transitions]
...
hospitalX=/pol/hospitalX-medic
```

In this example, the small difference mentioned above is amplified because of the increase of participants and possible transitions. Especially the number of [transitions] defined in the policy file, which has to be modified every time a new hospital joins the community. This alerts us the possible design pitfalls in a federated context where naming *everything* in advance is not possible.

4.3 Assumptions

In this part, we discuss the assumptions each system (and each type of the systems) hold. These assumptions contain both explicit assumptions discussed in their relevant literature and implicit assumptions held by their context.

As described above, research in non-data-tracking area (e.g. [Karjoth et al. 2002; Baumann et al. 2017; Schleicher et al. 2011]) holds assumptions that the institution has full knowledge, control and static assignment of permissions and policies internally and with the cooperator. The policy assigned to the data is designed by the institution and the same policy applies to all customers / users (given only a few choices, if any). Therefore, the control of who, when, where and how to use the data is the overall concern.

For the same reason, in these non-data-tracking systems, the original data is stored on servers controlled by the institution, the produced data is also stored back to that server, and all access is controlled by the institution as well. The work in Baumann et al. [2017] has a strong notion that data should be stored and fully governed by the institution, while Karjoth et al. [2002] has a weak assumption of this, and is further removed by Mont et al. [2003] and

Pearson and Casassa-Mont [2011] in their sticky policy framework through encryption; Schleicher et al. [2011] didn't talk about this, but because their model is for *process modeling* and the execution is done by the institution, it still, in fact, assumes the data is in control.

Meta-code [Håvard D. Johansen et al. 2015] (including [H. D. Johansen et al. 2014; A. T. Gjerdrum et al. 2016]), CamFlow [Pasquier et al. 2017] and Thoth [Elnogomi et al. 2016] (including SHAI [Eslam Elnikety et al. 2018]) all work on OS level, and view the datasets as data files on hard drive. These frameworks, in theory, can work across machines, as long as these machines have the same framework installed. Specifically, CamFlow has an explicit mechanism (CamFlow-MW) to check the runtime compliance across the boarder of machines [Pasquier et al. 2017].

For these dataflow-tracking systems / frameworks, the originator who assigns the policies is assumed to be the owner of the data. This is true for all the research we investigated in this category. That said, generally, the owner of the data is closely collaborated with the institution (or is "represented by" the institution) which uses the data or manages the use of data, because defining the policy requires knowing the user of data, specifying the software used to process the data and controlling the environment of processing the data in advance (which is only known from the processor side). Despite of this, each type of research has a different flavour:

- **Thoth** Thoth allows the policy-maker to specify "who is allowed to do what under what condition" (read, update, delete, declassify), which allows a fine-grained third-party permission management, initiated by the data owner.
- **Sticky policy framework** Because negotiation should happen prior to data access in order to get the decryption key, the data owner doesn't have to know in advance all possible scenarios if he/she acts as a TA.
- **Meta-code** and **CamFlow** The Meta-code approach and CamFlow both (implicitly) consider the data is collected by (or in collaboration with) the data controller.

For example, assuming in medical data area (where most of the examples are in): the medical data is collected from the patient who wears a medical device (and the data is transmitted from the device to the hospital); the data is then used by the hospital; the hospital possesses the ability to process the data and/or send the data to third-party (in compliance with the policies). For **Thoth**, the patient can specify (in the policy) that a certain hospital (identified by its key in cryptographic view) can perform certain action (e.g. process the data, allow third-party to process the data, allow third-party to

process the data but no further, etc); for **sticky policy**, the hospital should negotiate with the patient (or a TA) to assure him/her that the policies will be complied in order to acquire the decryption key; for **Meta-code** and **CamFlow**, the patient should specify the roles (e.g. doctor, nurse) and purposes (e.g. research) that are compatible of processing the data, but there needs to be a consistency of the roles and purposes between the patient and the hospital, and the hospital is in charge of annotating all the processing steps.

4.4 Expressiveness of Rules

One important benefit of machine-readable rules is that reasoning can be done on top of the rules. The capability is governed by the rule language. We examine the expressiveness of the languages in different topics.

Access control target

Most systems are designed for limiting the *use of data*, rather than the *use of a data file*, so their policies focus on limiting the read of data and have nothing to do with the update of data content. The only exception in our review is Thoth [Elnogomi et al. 2016] which focuses more on the OS and file (datasets are explicitly seen as different files), and it allows to specify different policies for *read* and *update*; whereas in other work, the data is assumed to be static. Table 3 contains a summary of the control target on data (with the addition of the capability of modeling processors, which is introduced later).

Even though, from our point of view, this is not a real difference since the primitives of other approaches can be easily extended to support this, by treating the "updated data" as a new dataset and control (in their respective ways) the process of creating the updated dataset.

Capability of terms

The terms allowed to be used to check against the data access is one big difference among the research. This acts as the most important difference in the expressiveness of policies in different approaches.

CamFlow centres around *tags*, drawing from the IFC (Information Flow Control) concept, and it limits the access by checking the tags of data [Pasquier et al. 2017]. Following the concept in DIFC [Myers and Liskov 1997], CamFlow uses two set of tags (called *labels*), *secrecy* label and *integrity* label, for different types of things, but the handling is the same. The

decision of using which label is not mentioned in the literature, but because their handling is the same, using which label does not make a real difference.

Meta-code uses roles and the meta-code to limit the access. The meta-code itself can use arbitrary operations (because it *is* a program) which gains the expressivity but sacrifices the capability of static analyzing and reasoning. Even this, the labels in LoNet (drawing some concepts from IFC) [Håvard D. Johansen et al. 2015] (our main examination target of Meta-code) provides some static analysis capabilities, because the processor needs to satisfy the label constraints.

Thoth, on the other hand, presents a set of predicates to allow the data owner to specify the policy. Elnogomi et al. [2016] has categorized the type of predicates into four categories (and two categories of connectives), as we re-mentioned in Table 2. The same team's following research SHAI [Eslam Elnikety et al. 2018] demonstrated the possibility of conducting static analysis (accompanied by dynamic analysis).

Sticky policy itself doesn't specify or limit the policy language, but the mechanism doesn't provide any help for policies in the form like Thoth, CamFlow and Meta-code, making them less suitable. Policy specification like the one-time restriction way may be a better approach as used in implementations (e.g. [Li et al. 2015]).

The one-time restriction approaches don't have thematic properties in their ways of representing policies. However, because of their nature, not tracking data-flow, the policy can represent arbitrary relevant information as long as the system supports. From our observation, these systems mainly model the data access policies, and none of them model the change of policies through computation.

Object of annotation

The third important difference is the object of annotation and the restrictions (apart from the expressiveness capability). The one-time restriction approaches (except [Schleicher et al. 2011]) focus solely on data and the environment (operator people), so the annotation only applies to the data. Sticky policy held the same view, even if we categorize it as data-flow tracking.

The only exception in the one-time restriction field is the industrial process modeling [Schleicher et al. 2011] which models the data and the process. From this perspective, this approach lies more along with the approaches talked about in Deployment Policy, focusing on cloud orchestration. They also model both the data and the process, emphasizing the process (which is the subject of deployment), and binds some characteristics of data with

process.

On the other hand, Meta-code and CamFlow annotate both the data and the processor (program), as well as the environment (i.e. the operator people, in their contexts): the annotation on data states the restrictions while the annotation of processor states the capability (of the processor) and the ability of the processor. "Capability of the processor" states the restrictions that the *input* data should have; "ability of the processor" states what the properties (the restrictions) of the *output* data will have regarding the input and the processor. For example, in Meta-code, a "annonymizer" processor can make the data less restrictive . In addition to this, CamFlow also restricts that each processor can only add or remove one tag (of each label), which limits the granularity of the processors.

Thoth views these from a different perspective: it attaches everything in the annotations of data, but also provides the ability in this set of annotations to model the change of rules given the processing steps. By this means, Thoth also models the processor in an indirect way, elaborating its ability to specify trust.

Trust handling

The way of handling trust is another important difference.

The one-time restriction implicitly assumes the institution has good control of the behaviour of its staffs and the cooperating body. Therefore, they do not take any effort in this aspect.

Sticky policy changes this by introducing TAs (Trusted Authorities), explicitly specified by the data owner (and can be the data owner himself/herself), to handle whether to trust the data consumer.

Thoth allows the data owner to specify in the policy that whom to trust. This can be based on cryptographic identities (pubic/secret key pairs), IP addresses, and trust delegation.

Meta-code and CamFlow don't have this mechanism, assuming the processing body is trustworthy.

Even though all the expressiveness capability given, all approaches rely on human judgment eventually. Table 4 contains the summary of trust handling.

4.5 Change of Policies

The processing of data is usually not copying input to output, so the processing can change the content and therefore the privacy/sensitive level of the data. Research dealing with data-flow usually takes this in their design, while the rest doesn't.

CamFlow and Meta-code both support annotating declassification capability for the process, and the resulting data can automatically be used accordingly; Thoth also has a term for declassification after some condition is met; the sticky policy framework itself doesn't have any notations for this and we didn't find any research targeting this aspect.

On the other hand, research dealing with one-time restriction had no notion of declassification. But this doesn't mean data can not be declassified under these approaches: the central storage can (manually) assign a new policy to the output data if it is considered less sensitive.

4.6 Update of Policies

Although not directly mentioned for most cases, most research hold the assumption that the policies, themes and participants won't update as time flows by.

Thanks to the encryption and TAs, the sticky policy framework [Pearson and Casassa-Mont 2011] unexplicitly supports changing of policies for future use of the data; for systems dealing with deployment policies (e.g. MiCADO [Kiss et al. 2017]), there doesn't seem to be a need for supporting such a use case, because the orchestration can be changed and re-deployed; for systems not dealing with data-flow tracking, the control power of the institution makes it easy to update the policy.

However, for most data-flow tracking systems, updating policies becomes a challenge because of the difficulty of tracking and changing the downstream policies, unless the log (if any) can correctly and completely identify which input data changes through which process to which output data, which is essentially provenance. For systems limiting the processing scope inside, this is a little easier compared to systems without such restriction. All of CamFlow, Meta-code and Thoth are seeing processing as inside one (big) environment, so they can track using the lineage / audit log if the log is present.

References

A. T. Gjerdrum, H. D. Johansen, and D. Johansen (June 2016). "Implementing Informed Consent as Information-Flow Policies for Secure Analytics on eHealth Data: Principles and Practices". In: *2016 IEEE First International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE)*. 2016 IEEE First International Confer-

ence on Connected Health: Applications, Systems and Engineering Technologies (CHASE), pp. 107–112.

Aktug, Irem and Katsiaryna Naliuka (Feb. 21, 2008). "ConSpec – A Formal Language for Policy Specification". In: *Electronic Notes in Theoretical Computer Science*. Proceedings of the First International Workshop on Run Time Enforcement for Mobile and Distributed Systems (REM 2007) 197.1, pp. 45–58.

Aydoğan, Reyhan, Pinar Øzturk, and Yousef Razeghi (Oct. 30, 2017). "Negotiation for Incentive Driven Privacy-Preserving Information Sharing". In: *PRIMA 2017: Principles and Practice of Multi-Agent Systems*. Lecture Notes in Computer Science. Springer, Cham, pp. 486–494.

Baarslag, Tim, Alper T. Alan, Richard Gomer, Muddasser Alam, Charith Perera, Enrico H. Gerding, and m.c. schraefel m.c. (2017). "An Automated Negotiation Agent for Permission Management". In: *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*. AAMAS '17. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, pp. 380–390.

Baarslag, Tim and Michael Kaisers (2017). "The Value of Information in Automated Negotiation: A Decision Model for Eliciting User Preferences". In: *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*. AAMAS '17. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, pp. 391–400.

Baumann, Felix W., Uwe Breitenbücher, Michael Falkenthal, Gerd Grünert, and Sebastian Hudert (Sept. 17, 2017). "Industrial Data Sharing with Data Access Policy". In: *Cooperative Design, Visualization, and Engineering*. International Conference on Cooperative Design, Visualization and Engineering. Lecture Notes in Computer Science. Springer, Cham, pp. 215–219.

Binz, Tobias, Uwe Breitenbücher, Florian Haupt, Oliver Kopp, Frank Leymann, Alexander Nowak, and Sebastian Wagner (Dec. 2, 2013). "OpenTOSCA – A Runtime for TOSCA-Based Cloud Applications". In: *Service-Oriented Computing*. International Conference on Service-Oriented Computing. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, pp. 692–695.

Breitenbucher, Uwe, Tobias Binz, Christoph Fehling, Oliver Kopp, Frank Leymann, and Matthias Wieland (2014). "Policy-Aware Provisioning and Management of Cloud Applications". In: p. 23.

Breitenbucher, Uwe, Tobias Binz, Oliver Kopp, Frank Leymann, and Matthias Wieland (2013). "Policy-Aware Provisioning of Cloud Applications". In: p. 11.

Elnogomi, Eslam, Aastha Mehta, Anjo Vahldiek-Oberwagner, Deepak Garg, and Peter Druschel (Aug. 10, 2016). "Thoth: Comprehensive Policy Compliance in Data Retrieval Systems". In:

Elrakaiby, Yehia, Frédéric Cuppens, and Nora Cuppens-Boulahia (Jan. 1, 2012). "Formal enforcement and management of obligation policies". In: *Data & Knowledge Engineering* 71.1, pp. 127–147.

Eslam Elnikety, Deepak Garg, and Peter Druschel (Jan. 14, 2018). "Shai: Enforcing Data-Specific Policies with Near-Zero Runtime Overhead". In: *arXiv:1801.04565 [cs]*. arXiv: 1801.04565.

European Open Science Cloud (EOSC) — Open Science - Research and Innovation - European Commission (2018). URL: https://ec.europa.eu/research/openscience/index.cfm?pg=open-science-cloud (visited on 07/31/2018).

Fischer, Markus Philipp, Uwe Breitenbücher, Kálmán Képes, and Frank Leymann (2017). "Towards an Approach for Automatically Checking Compliance Rules in Deployment Models". In: *Proceedings of The Eleventh International Conference on Emerging Security Information, Systems and Technologies (SECURWARE)*. Xpert Publishing Services (XPS), pp. 150–153.

Gomer, Richard Charles, m c schraefel m c, and Enrico Gerding (Sept. 13, 2014). "Consenting agents: semi-autonomous interactions for ubiquitous consent". In: Workshop, UbiComp'14: How Do You Solve a Problem like Consent? In collab. with Richard Charles Gomer, m c schraefel m c, and Enrico Gerding, pp. 653–658.

H. D. Johansen, W. Zhang, J. Hurley, and D. Johansen (Apr. 2014). "Management of body-sensor data in sports analytic with operative consent". In: *2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*. 2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), pp. 1–6.

Håvard D. Johansen, Eleanor Birrell, Robbert van Renesse, Fred B. Schneider, Magnus Stenhaug, and Dag Johansen (2015). "Enforcing Privacy Policies with Meta-Code". In: ACM Press, pp. 1–7.

Hutton, L. and T. Henderson (Jan. 2018). "Toward Reproducibility in Online Social Network Research". In: *IEEE Transactions on Emerging Topics in Computing* 6.1, pp. 156–167.

Hutton, Luke and Tristan Henderson (2017). "Beyond the EULA: Improving Consent for Data Mining". In: *Transparent Data Mining for Big and Small Data*. Studies in Big Data. Springer, Cham, pp. 147–167.

Karjoth, Günter, Matthias Schunter, and Michael Waidner (Apr. 14, 2002). "Platform for Enterprise Privacy Practices: Privacy-Enabled Manage-

ment of Customer Data". In: *Privacy Enhancing Technologies*. International Workshop on Privacy Enhancing Technologies. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, pp. 69–84.

Kiss, Tamas, Peter Kacsuk, Jozsef Kovacs, Botond Rakoczi, Akos Hajnal, Attila Farkas, Gregoire Gesmier, and Gabor Terstyanszky (Sept. 27, 2017). "MiCADO—Microservice-based Cloud Application-level Dynamic Orchestrator". In: *Future Generation Computer Systems*.

Laat, Cees de (2018). *DL4LD*. URL: `http://www.delaat.net/dl4ld/` (visited on 04/16/2018).

Li, S., T. Zhang, J. Gao, and Y. Park (Mar. 2015). "A Sticky Policy Framework for Big Data Security". In: *2015 IEEE First International Conference on Big Data Computing Service and Applications*. 2015 IEEE First International Conference on Big Data Computing Service and Applications, pp. 130–137.

Luger, Ewa and Tom Rodden (2013). "An Informed View on Consent for UbiComp". In: *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. UbiComp '13. New York, NY, USA: ACM, pp. 529–538.

Mont, M. C., S. Pearson, and P. Bramhall (Sept. 2003). "Towards accountable management of identity and privacy: sticky policies and enforceable tracing services". In: *14th International Workshop on Database and Expert Systems Applications, 2003. Proceedings*. 14th International Workshop on Database and Expert Systems Applications, 2003. Proceedings. Pp. 377–382.

Myers, Andrew C. and Barbara Liskov (1997). "A Decentralized Model for Information Flow Control". In: *Proceedings of the Sixteenth ACM Symposium on Operating Systems Principles*. SOSP '97. New York, NY, USA: ACM, pp. 129–142.

Ni, Qun, Dan Lin, Elisa Bertino, and Jorge Lobo (Sept. 24, 2007). "Conditional Privacy-Aware Role Based Access Control". In: *Computer Security – ESORICS 2007*. European Symposium on Research in Computer Security. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, pp. 72–89.

Pasquier, Thomas F. J.-M., Jatinder Singh, David Eyers, and Jean Bacon (July 1, 2017). "CamFlow: Managed Data-sharing for Cloud Services". In: *IEEE Transactions on Cloud Computing* 5.3, pp. 472–484. arXiv: `1506.04391`.

Pearson, S. and M. Casassa-Mont (Sept. 2011). "Sticky Policies: An Approach for Managing Privacy across Multiple Parties". In: *Computer* 44.9, pp. 60–68.

Schleicher, D., S. Grohe, F. Leymann, P. Schneider, D. Schumm, and T. Wolf (Dec. 2011). "An approach to combine data-related and control-flow-related compliance rules". In: *2011 IEEE International Conference on Service-Oriented Computing and Applications (SOCA)*. 2011 IEEE International Conference on Service-Oriented Computing and Applications (SOCA), pp. 1–8.

TOSCA v1.0 Committee Specification 01 published by the TOSCA TC — OASIS (Mar. 22, 2013). URL: `https://www.oasis-open.org/news/announcements/tosca-v1-0-committee-specification-01-published-by-the-tosca-tc` (visited on 04/13/2018).

Trani, Luca, Mathijs Koymans, Malcolm Atkinson, Reinoud Sleeman, and Rosa Filgueira (Sept. 1, 2017). "WFCatalog: A catalogue for seismological waveform data". In: *Computers & Geosciences* 106, pp. 101–108.

Waizenegger, Tim, Matthias Wieland, Tobias Binz, Uwe Breitenbücher, Florian Haupt, Oliver Kopp, Frank Leymann, Bernhard Mitschang, Alexander Nowak, and Sebastian Wagner (Sept. 9, 2013). "Policy4TOSCA: A Policy-Aware Cloud Service Provisioning Approach to Enable Secure Cloud Computing". In: *On the Move to Meaningful Internet Systems: OTM 2013 Conferences*. OTM Confederated International Conferences "On the Move to Meaningful Internet Systems". Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, pp. 360–376.

Wettinger, J., U. Breitenbücher, and F. Leymann (Dec. 2014). "Standards-Based DevOps Automation and Integration Using TOSCA". In: *2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*. 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing, pp. 59–68.

Zhao, G., C. Rong, J. Li, F. Zhang, and Y. Tang (Nov. 2010). "Trusted Data Sharing over Untrusted Cloud Storage Providers". In: *2010 IEEE Second International Conference on Cloud Computing Technology and Science*. 2010 IEEE Second International Conference on Cloud Computing Technology and Science, pp. 97–103.