Log Design for Accountability

Denis Butin, Marcos Chicote, Daniel Le Métayer

Log Design for Accountability

Denis Butin, Marcos Chicote and Daniel Le Métayer
Inria, Université de Lyon
INSA-Lyon, CITI-Inria
F-69621, Villeurbanne, France
denis.butin@inria.fr; mchicote@dc.uba.ar; daniel.le-metayer@inria.fr

***Abstract*—Accountability is a requirement to be included in the initial design phase of systems because of its strong impact on log architecture implementation. As an illustration, the logs we examine here record actions by data controllers handling personally identifiable information to deliver services to data subjects. The structures of those logs seldom consider requirements for accountability, preventing effective dispute resolution. We address the question of what information should be included in logs to make their a posteriori compliance analysis meaningful. Real-world scenarios are used to show that decisions about log architecture are nontrivial and should be made from the design stage on. Four categories of situations for which straightforward solutions are problematic are presented. Our contribution shows how log content choices and accountability definitions mutually affect each other and incites service providers to rethink up to what extent they can be held responsible. These different aspects are synthesized into key guidelines to avoid common pitfalls in accountable log design. This analysis is based on case studies performed on our implementation of the PPL policy language.**

I. Introduction

As software presence in daily life increases, so does the exchange of information between individuals, companies, government agencies and other entities. In particular, Personally Identifiable Information (PII) is often shared by *data subjects* (DS) in exchange for services. PII is also frequently transmitted between companies for outsourcing purposes.

The flow of PII is of particular concern as the information transmitted can be used to uniquely identify a single individual. The illicit collection and reselling of PII can lead to profitable business. Even more unpleasant consequences such as stalking or identity theft can arise through criminal obtainment of PII.

To address these threats, legislation on how PII can be collected, distributed and accessed is becoming more specific, especially in the EU where a Data Protection Regulation [1] is expected later this year, building on the existing 1995 Directive [2]. Notably, in 2010, the Article 29 Data Protection Working Party published an Opinion specifically about the accountability principle [3], which is expected to influence the upcoming regulation. Legislation in the US is less comprehensive, although the 1996 HIPAA [4], 1998 COPPA [5] and 1999 GLBA [6] provide partial safeguards for the private sector and the 1974 Privacy Act [7] does so for government agencies.

In practice, the entities collecting the data — *data controllers* (DCs) — address these issues in their privacy policies. The traditional approach includes preventive techniques, such as access control, encryption or anonymization. This modus operandi however falls short in certain situations and has thus lost part of its potency. Consider for instance the case of a physician in presence of a patient who needs emergency assistance, presented in [8]. Under these conditions, a preventive approach would not allow the physician to access the patient's medical records.

In order to address the shortcomings of the traditional approach, a posteriori compliance control has been proposed. In this paradigm, DCs are allowed to manipulate data a priori and are trusted to follow the rules, but must create a data handling registry. In case a claim is held against a DC, the information on how data was handled, available from the registers, can help determine if a rule was breached. As a result, accountability becomes one of the primary means to implement policies.

A precise definition of accountability is needed for this approach to have validity. Discussion of possible definitions can be found in [9], [10], [11], [12]. Bennett [13] distinguishes between accountability of policy, of procedures, and of practice. *Accountability of practice* is our focus: in this context, it regards the actual, practical processing of data and the representation of that processing. Another view of accountability, due to Raab, focuses on the nature of the evidence that enables accountability, which he calls the *account*. We investigate the conditions that PII handling traces must meet to constitute meaningful accounts.

Several frameworks for a posteriori compliance control have been described, but little attention has been paid to the design of logs supporting accountability. This paper provides a general discussion on what information must be included in logs to support a posteriori analysis. We present, by means of examples, some of the challenges that need to be considered and possible solutions. While it may initially be tempting to simply require "exhaustive" logs, the solution is not as simple as it may seem. Firstly, excessively detailed records go against the requirements of data minimization and data sanitization. Minimization helps reduce the tension that logging creates with DS privacy. For instance, if compliance checking is performed by a third party on the behalf of a DS, it is in his interest to minimize trust requirements directed toward that third party by including in the logs only information essential for compliance checking. In addition, even if strong security measures must obviously be applied for logs, log minimization is also a sound policy to limit possible data leaks due to

potential security attacks since no solution provides absolute protection. In the same spirit, logs provided to auditors should not leak data linked to other DS than the one for which the compliance checking is performed. On the other side, in some cases the DC may require data sanitization to keep operational details of his system confidential: by revealing only selected data, less is known to auditors about the mechanisms of the platform he operates. Let us note that a posteriori compliance control ought to be distinguished from forensics to this respect and their requirements in terms of logging are rather different. Forensics applies in unexpected circumstances and must be able to provide information on security attacks that were not necessarily foreseen. Therefore, keeping as much information as possible in the logs can be a reasonable strategy. By contrast, a posteriori compliance verifications follow predefined rules arising from contracts, which makes it possible to tailor logs to the actual needs of the analysis. Comprehensive logging, while it can be attractive for forensics, is hence not necessarily the right approach to a posteriori compliance control. Last but not least, the notion of "exhaustiveness" is not clear in this context: as we will see in the next sections, in some cases contextual information is necessary to decide upon compliance. Therefore, a "compliance aware" log is not merely a log recording all system events.

We first recall related work regarding a posteriori compliance control frameworks and log design (§II). A discussion of the main challenges of log design for accountability (§III) and a number of guidelines to overcome them (§IV) follow. Conclusions and prospects for future work complete the paper (§V).

II. Related Work

Several frameworks for a posteriori compliance control have already been developed. Etalle and Winsborough [14] present a logical framework for using logs to verify that actions taken by the system are authorized. Cederquist et al. [15] introduce a framework to control compliance of document policies where users may be audited and asked to justify that an action was in compliance with a policy. The reader can refer to [8], [16], [17], [18], [19] for more information on accountability for privacy with a posteriori compliance control frameworks.

Log design for accountability is not a topic much discussed in the literature. Nonetheless, some work has been done, mostly related to log architecture. For related work on log design, the reader can refer to [20], [21], [22]. Work presented in [23] proposes criteria for acceptable log architecture depending on the features of the system and the potential claims between the parties.

The general approach of enforcing PII privacy through policy languages appeared in the TAS^3 project architecture[24], where the PERMIS authorization infrastructure [25] and XACML policy language [26] are supported. The related issue of usability is tackled by the project through the automatic translation of access control policies from a controlled natural language interface into machine-processable formats [27]. Another part of the project, described in [28], discusses broadly an accountability framework based on security policies. However, the focus of that discussion is the prevention of user misconduct by making *them* accountable, not the DC.

III. Log Design Pitfalls

Our contribution is based on an effort to build a formal definition of the PrimeLife Privacy Policy Language (PPL). PPL, parts of which build on XACML, was first presented in [29] and a specification appeared in May 2011 [30]. Its purpose is to express access and usage control rules in a symmetric way for the DS and the DC.

Following this work, we have defined an abstract version of a subset of PPL, which serves as a basis for our accountability framework. We then formalized and implemented a log compliance analyzer. Fig. 1 gives an overview of the relevant components.

In the course of this work, several questions regarding what information to include in the events definition arose. This paper is the result of an effort to address those questions and to derive general lessons applicable to any accountability framework. In this section, we present four categories of log design problems and corresponding examples.

We will show how some of the examples could be modeled in PPL. PPL provides a mechanism called *obligations* which allows the DS to define a series of obligations with which the DC should comply once the PII is sent. Obligations are defined in terms of triggers and actions. This means that upon

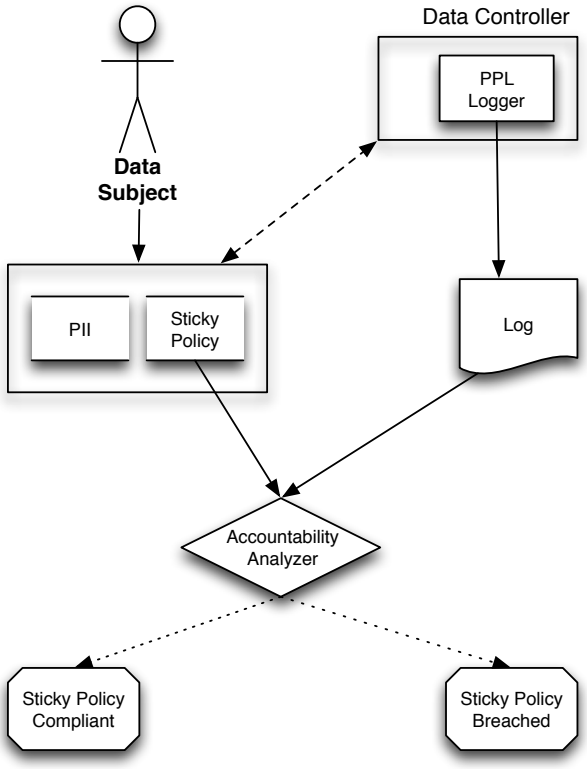

Fig. 1. PPL accountability analysis framework

the occurrence of a specific event, a particular action must take place within a defined period of time. Many trigger events are defined in PPL, including the deletion of a PII, its update by the DS, the sending of a PII from a DC to a third party (also called *downstream data controller*), the DS accessing his own PII remotely and the use by the DC of a PII for a specific purpose. Two other trigger events that will feature in our illustrative scenarios below are `TriggerAtTime` and `TriggerPeriodic`. While `TriggerAtTime` is used to define actions that must happen at least once, between a specified starting time and a deadline, `TriggerPeriodic` is used for obligations where a given action must occur several times with a set periodicity (and a periodic deadline).

Furthermore, *authorizations* are used to declare whether PII can be transmitted to downstream data controllers and for which specific purposes it may be used. Purposes are codified by standardized URIs.

The DC also initially defines obligations and authorizations with which it is willing to comply. Obligations and authorizations defined by the DS (*data handling preferences*) and the DC (*data handling policies*) are then matched automatically by the PPL engine, resulting in an agreed-upon *sticky policy* if a match can be found. In case of a mismatch, the process can only continue if either the data handling policies or data handling preferences are changed first and a new matching succeeds.

A. Example Scenario

Let us consider the example of data handling events for a private bank account. The bank's customer is the DS, and the bank is the DC. The PII may consist of the DS' name, address, phone number and email address. Let us also consider a downstream data controller: the company managing the credit cards of the bank's customers.

The following Fig. 2 is an example of a possible sticky policy expressed in PPL:

```
Obligation 1:
TriggerAtTime [2013-05-23, 7]
==> ActionNotifyDS [post, 335 Powell Street]

Obligation 2:
TriggerAtTime [2013-05-25, 3]
==> ActionNotifyDS [post, 335 Powell Street]

Obligation 3:
TriggerPeriodic [2013-05-24, 2014-05-24, 2, 5]
==> ActionNotifyDS [sms, 555-2106]

Authorizations:
AuthorizationForPurpose [marketing]
AuthorizationDownStreamUsage [False]
```

Fig. 2. Example PPL sticky policy

Even though our code formatting is slightly different, the parameters of Fig. 2 strictly follow the PPL specifications. The motivations for the obligations in Fig. 2 are the following: the bank should send its customer a notification via postal mail once his new credit card is available for retrieval at the branch, which is assumed to take place no later than May 30th. Additionally, the bank has a second obligation to send the customer another notification by postal mail within three days from May 25th; this time, the letter should contain the PIN code for the new card. The sticky policy also requires the bank to send the customer a notification by text message every five days. The text message should contain the client's current balance, and should be sent periodically every five days with a maximal delay of two days for each period. Those notifications should be sent for a total duration of one year.

Note that the PPL sticky policy arising from these informal obligations does not include all the parameters one may expect, such as the body of the notifications. It was our goal to base our analysis on an actual policy language, without any extension or deviation, so as to root the resulting synthesis in reality. The definition in of Fig. 2 is the closest way to translate the above obligations from natural language into PPL. The fact that this definition does not feature full detail merely showcases a limitation of policy languages in general (PPL is not an exception in this respect).

Two tables present the PPL trigger events (see Table I) and action events (see Table II) appearing in the subsequent example log. Other events also exist in PPL, but are not used in our running example.

Now, consider an example log of the PII handling events between the customer, the bank managing the customer's account and the credit card company (see Fig. 3). In the remainder of this section, we investigate a number of issues that can arise when analyzing this kind of log.

B. Insufficient Event Information

The first set of issues we address comes from the ambiguity that can arise from missing parameters in the log entries for given events.

Consider the reception of a new credit card by the DS. The DS has provided the bank with PII and expects the bank to send the card's PIN number by post within a week from May 25th. The DS also expects the DC to send a letter stating the availability of the new card at the bank's branch, within a week from May 23rd. The DC has agreed to these obligations, as shown in the sticky policy earlier (see Fig. 2).

The log listing (see Fig. 3) shows that both triggers for `Obligation 1` and `Obligation 2` have been fired. Additionally, a notification has been sent to the DS' address by postal mail.

Now consider a compliance check for this log on May 29. At this stage, the analyzer cannot conclude whether the log is compliant or not. The undecidability comes from the fact that it is not possible to tell whether the notification was related to `Obligation 1` or `Obligation 2`. What was included in the notification to the DS? The PIN, or the card availability confirmation? If we assume that the notification corresponds to `Obligation 1`, the log is noncompliant because no

TABLE I
PPL TRIGGER EVENTS

Trigger event name	Parameter 1	Parameter 2	Parameter 3	Parameter 4
`TriggerAtTimeEvent`	Start date	Maximal delay	✗	✗
`TriggerDSAccess`	URL for PII accessed by DS	✗	✗	✗
`TriggerPersonalDataAccessedPurpose`	Purpose	Maximal delay	✗	✗
`TriggerPeriodic`	Start date	End date	Maximal delay	Period
`TriggerPersonalDataSent`	Downstream DC identifier	Maximal delay	✗	✗
`TriggerUpdate`	Maximal delay	✗	✗	✗

```
2013-05-23T16:23  TriggerAtTimeEvent [2013-05-23, 7d]
2013-05-23T20:15  TriggerDSAccess [https://mybank.net/johndoe/pii1729]
2013-05-24T10:47  TriggerPersonalDataAccessedPurpose [marketing, 2d]
2013-05-24T14:52  TriggerPeriodic [2013-05-24, 2014-05-24, 2d, 5d]
2013-05-24T14:53  ActionNotifyDS [sms, 555-2106]
2013-05-25T17:01  TriggerAtTimeEvent [2013-05-25, 3d]
2013-05-26T23:32  ActionNotifyDS [post, 335 Powell Street]
2013-05-27T12:07  ActionAnonymizePersonalData
2013-05-28T03:18  TriggerPersonalDataSent [Mastercard, 5d]
2013-05-29T14:51  TriggerPeriodic [2013-05-24, 2014-05-24, 2d, 5d]
2013-05-29T14:51  ActionNotifyDS [email, johndoe@comcast.net]
2013-05-29T14:54  ActionNotifyDS [sms, 555-2106]
2013-06-02T08:21  TriggerUpdate [60m]
2013-06-02T21:50  ActionNotifyDS [sms, 555-2106]
```

Fig. 3. Example PPL PII events log

TABLE II
PPL ACTION EVENTS

Action event name	Parameter 1	Parameter 2
`ActionNotifyDS`	Media	Address
`ActionAnonymizePersonalData`	✗	✗

notification was sent for `Obligation 2`. Assuming that it corresponds to `Obligation 2`, the log is compliant, because the deadline for `Obligation 1` is May 30th.

Ambiguity due to the lack of explicitness of the action event propagates to the level of the compliance analysis. The issue is that the relationship between action events and the triggers they relate to is not reflected by the log. A solution is to add new parameters to both trigger and action events. Every trigger event should carry a unique `TriggerID`, and action events should feature a `TriggerReference` parameter that refers explicitly to the trigger it satisfies. This additional information, which eliminates the aforementioned undecidability, illustrates the notion of "compliance aware" logs. It illustrates the fact that simply recording all the events of the system is not necessarily sufficient: extra information can be required depending on the policies the system has to comply with.

C. Incomplete Support for Third Party Interaction

In this subsection, we consider situations for which contextual information needs to be taken into account.

Consider `Obligation 3` in the sticky policy of Fig. 2. This obligation requires the bank to notify the customer with the account's balance by text message periodically, every five days, with a maximal delay of two days. The obligation also expires after one year.

In case the customer does not receive the text message in due time, he should rightfully consider that the agreement was breached. However, the message may indeed have been sent by the bank but never reached its destination due to a SMS gateway malfunction on the telecommunications operator side.

Logging of this communication should therefore not be limited to an event stating that the DC attempted to send a text message. Communication between the DC and a third party, in this case a telecommunications operator, should be included in the log so the event history is expressive enough to pinpoint the issue.

Such situations also raise the question of the DC's accountability under these conditions and how errors are handled. Depending on the precise legal terms defining the liability of the DC (obligation of means or obligation of performance), the bank could be held accountable or not.

There is an additional issue. Note that the third party is actually a downstream data controller since PII (the account's balance) is shared with the telecommunications operator. The security policy language (PPL in our case study) lacks expressiveness for the definition of downstream PII handling: a global switch, `AuthorizationDownStreamUsage`, has a boolean value. Authorizing downstream usage for a whitelist of entities to the exclusion of others is unsupported. It is only possible to globally enable downstream usage and then define triggers for specific downstream data controllers. Stronger accountability calls for more fine-grained downstream usage parametrization.

D. No Support for Break-glass Situations

Break-glass [31] situations (referring to the breaking of glass to trigger an alarm) refer to circumstances under which exceptional access to data should be granted to an entity that does not possess the required privileges. Situations that fall under this category, like the one presented in Section I about a physician in need of a patient's medical records in a life-threatening situation, need to be taken into consideration when building accountability mechanisms. In particular, they must be considered when designing log architecture for such mechanisms. Feigenbaum et al. [8] present the case of military information classification systems, for which this requirement applies.

Returning to our running example of a DS interacting with a bank, let us consider the situation in case of credit card fraud. In uneventful circumstances, customers do not want banks to share their PII with third parties. In case of suspicious activity, however, the bank may need to contact the company managing the customer's credit card and share the customer's contact information with it. This is logged through the event `TriggerPersonalDataSent [Mastercard, 5d]` in our example log, which models the sharing of PII with a downstream DC. Since one of the authorizations in the sticky policy (`AuthorizationDownStreamUsage [False]`) forbids downstream usage, this event breaches the predefined data-handling agreements.

Another example is illegal activity that would prompt the bank to contact law enforcement authorities. This example raises the more general issue of how laws interact with user defined obligations negotiated with the DC. In most states, the law would take precedence over particular obligations and should be taken into consideration during the a posteriori compliance control. For this process to be automatic, the DC should include evidence in the log that explains why the obligation agreed upon with the DS was breached. In terms of accountability, the DC answers to two entities simultaneously: the DS and the state.

While it may be possible to include the analysis of some break-glass situations in automatic compliance analyzers, in general this kind of actions require human analysis. Current usage policy languages such as PPL lack expressiveness for this type of situations.

E. No Integration With Manual Verification

While both the sticky policy and log used in our running example include notifications in obligations and action events, they do not carry comprehensive information such as the actual contents of the notification messages. Other obligations may require DCs to perform actions that cannot be integrated in the log available to the auditor because they are by essence informal or defined too vaguely and thus cannot be formalized in a policy language. In such situations, complementary checks must be performed by human agents in addition to the mechanized compliance analysis. Two main issues have to be solved to ensure a proper integration of the manual and automatic phases: (i) the policy language (and thus the logs) must integrate links to documents defining (e.g. in natural language) the informal requirements to be checked by the human auditor and (ii) the analyzer must account for the complementary manual verifications (either through an interactive mode or by outputing all verifications to be carried out by the auditor in a second stage).

IV. Guidelines for Accountable Log Design

Four major guidelines for log design emerged on the basis of our work with PPL:

a) Log architecture and precise definitions of accountability are intertwined. Their joint design constitutes an iterative process: Log design is not a purely technical activity. Managers, lawyers and functional analysts should be involved. This principle is illustrated in Section III-C. Changing circumstances can define or alter the extent of responsibility for the activities of the DC, making it accountable for actions that were not part of the initial scope.

In addition, log design should be seen as an iterative process because as the definition of accountability for the business under consideration changes, new log definition issues may emerge, requiring a review of initial design choices. As a result of such review, accountability definition might require changes, restarting the cycle. The issue of downstream data controllers is also relevant to this point. Policy languages often allow agreements between the DC and the DS on how third parties can handle PII. It is the responsibility of the DC to forward those agreements to the third party. The DC can be responsible for the third party complying with them, depending on previously agreed-upon liabilities. Again, log architecture goes hand-in-hand with accountability definitions: it may be necessary for part of the third party's event logs to be available for inspection, or even incorporated in the DC's logs.

b) Log architecture should reflect full policy language semantics: Log designers ought to consider all aspects of policy language semantics. Explicitness is paramount. Languages tend to express more than what is explicitly stated. Everything that can be expressed with a policy language is potentially a claiming point. Logs should therefore feature enough expressiveness to elucidate whether a policy has been breached, which includes its full semantic content. The need for this guideline is illustrated by the PPL obligations discussed earlier (§III-B). PPL obligations not only include semantics for triggers and actions, but also for the causal relationship between them. This relationship must be reflected by logs. Contextual conditions not explicitly described by an obligation should also be taken into consideration in this light.

c) It should be possible to model break-glass situations in logs: Exceptional situations occur in almost every system and should therefore be supported from the start in systems, and therefore in logs. Even if complementary human analysis is required for such cases, logs should still be able to reflect their existence. This implies preliminary planning to decide how unusual circumstances should be dealt with, increasing the soundness of future compliance analysis. In practical terms, logs ought to support the specification of conjunctions of

trigger events and contextual data so more expressiveness can be achieved. Indeed, break-glass situations are generally characterized by unusual combinations of circumstances that need to be precisely describable in the logs.

d) Links between formal specifications and policies requiring human verification are needed: Some obligations expressed by policy languages may entail events that cannot be checked mechanically, for instance because they entail physical realization and are therefore beyond the scope of formal semantics. Checking these obligations involves human intervention. Verification tools can still partially integrate this aspect by outputting instructions to be followed by human agents to carry out manual compliance checking, or providing a semi-interactive mode prompting the auditor for information about the informal assumptions during the audit. Compliance can then be justified more strongly through a complete argumentation that ties in formal and manual verification.

V. Conclusion

In this position paper, we discuss the issues raised by log design for accountability and address the question of which information should be included in logs for meaningful a posteriori compliance control. Real-world examples are analyzed to demonstrate how log design for accountability is not a trivial task and should be taken into consideration from the design stage on. These examples are generalized and categorized under four classes of situations that need to be addressed to design "compliance aware" logs. Key guidelines to avoid common pitfalls are presented.

In future work, we plan to provide a formal framework for the verification of properties of accountability architectures. This framework would make it possible to characterize precisely the guarantees provided by a posteriori compliance checking and the underlying assumptions. Another avenue for research is the study of the minimality of the logs with respect to the policies to be checked and the application of data sanitization techniques to remove sensitive information that is not crucial for accountability. It would also be interesting to be able to reason about the comprehensiveness of the logs (with respect to the policies) when requirements of the DC have to be taken into account: for example, is it still possible to decide upon compliance if a certain type of action or data cannot be kept in the logs?

Acknowledgment

The authors thank Slim Trabelsi and Francesco Di Cerbo at SAP Research for clarifications about PPL. This work was partially funded by the European project FI-WARE / FP7-2012-ICT-FI and the Inria Project Lab CAPPRIS (Collaborative Action on the Protection of Privacy Rights in the Information Society).

References

[1] European Commission, "Proposal for a Regulation of the European Parliament and of the Council on the Protection of Individuals with Regard to the Processing of Personal Data and on the Free Movement of such Data (General Data Protection Regulation)," 2012.

[2] European Parliament and the Council of the European Union, "Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the Protection of Individuals with Regard to the Processing of Personal Data and on the Free Movement of such Data," 1995.

[3] Article 29 Data Protection Working Party, "Opinion 3/2010 on the principle of accountability," 2010.

[4] United States Congress, "Health Insurance Portability and Accountability Act," 1996.

[5] ——, "Children's Online Privacy Protection Act," 1998.

[6] ——, "Gramm-Leach-Bliley Act," 1999.

[7] ——, "Privacy Act," 1974.

[8] J. Feigenbaum, J. Hendler, A. D. Jaggard, D. J. Weitzner, and R. N. Wright, "Accountability and deterrence in online life (extended abstract)," in *ACM WebSci'11*, June 2011, pp. 1–7.

[9] D. J. Weitzner, H. Abelson, T. Berners-Lee, J. Feigenbaum, J. Hendler, and G. J. Sussman, "Information accountability," *Commun. ACM*, vol. 51, no. 6, pp. 82–87, Jun. 2008.

[10] S. Eriksén, "Designing for accountability," in *Proceedings of the second Nordic conference on Human-computer interaction*, ser. NordiCHI '02. New York, NY, USA: ACM, 2002, pp. 177–186.

[11] D. Le Métayer, "A formal privacy management framework," in *Formal Aspects in Security and Trust*, ser. Lecture Notes in Computer Science, P. Degano, J. D. Guttman, and F. Martinelli, Eds., vol. 5491. Springer, 2008, pp. 162–176.

[12] ——, "Formal methods as a link between software code and legal rules," in *Proceedings of the 9th international conference on Software engineering and formal methods*, ser. SEFM'11. Berlin, Heidelberg: Springer, 2011, pp. 3–18.

[13] Colin J. Bennett, "Implementing Privacy Codes of Practice," *Canadian Standards Association*, 1995.

[14] S. Etalle and W. H. Winsborough, "A posteriori compliance control," in *Proceedings of the 12th ACM symposium on Access control models and technologies*, ser. SACMAT '07. New York, NY, USA: ACM, 2007, pp. 11–20.

[15] J. G. Cederquist, R. Corin, M. A. C. Dekker, S. Etalle, J. I. den Hartog, and G. Lenzini, "Audit-based compliance control," *Int. J. Inf. Secur.*, vol. 6, no. 2, pp. 133–151, Mar. 2007.

[16] J. Feigenbaum, A. D. Jaggard, and R. N. Wright, "Towards a formal model of accountability," in *Proceedings of the 2011 workshop on New security paradigms workshop*, ser. NSPW '11. New York, NY, USA: ACM, 2011, pp. 45–56.

[17] J. Cederquist, R. Corin, M. Dekker, S. Etalle, and J. den Hartog, "An Audit Logic for Accountability," in *Proceedings of the Sixth IEEE International Workshop on Policies for Distributed Systems and Networks*, A. Sahai and W. Winsborough, Eds. Los Alamitos, California: IEEE Computer Society Press, 2005, pp. 34–43.

[18] R. Jagadeesan, A. Jeffrey, C. Pitcher, and J. Riely, "Towards a theory of accountability and audit," in *Proceedings of the 14th European conference on Research in computer security*, ser. ESORICS'09. Berlin, Heidelberg: Springer, 2009, pp. 152–167.

[19] R. Thion and D. Le Métayer, "FLAVOR: a Formal Language for A posteriori Verification of Legal Rules," in *Proceedings of the 2011 IEEE International Symposium on Policies for Distributed Systems and Networks*, ser. POLICY '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 1–8.

[20] B. Schneier and J. Kelsey, "Secure Audit Logs to Support Computer Forensics," *ACM Trans. Inf. Syst. Secur.*, vol. 2, no. 2, pp. 159–176, 1999.

[21] M. Bellare and B. S. Yee, "Forward integrity for secure audit logs," University of California at San Diego, Tech. Rep., 1997.

[22] B. Waters, D. Balfanz, G. Durfee, and D. K. Smetters, "Building an encrypted and searchable audit log," in *The 11th Annual Network and Distributed System Security Symposium*, 2004.

[23] D. Le Métayer, E. Mazza, and M.-L. Potet, "Designing log architectures for legal evidence," in *SEFM*, J. L. Fiadeiro, S. Gnesi, and A. Maggiolo-Schettini, Eds. IEEE Computer Society, 2010, pp. 156–165.

[24] "TAS^3: Trusted Architecture for Securely Shared Services," http://vds1628.sivit.org/tas3/, 2011.

[25] D. W. Chadwick, G. Zhao, S. Otenko, R. Laborde, L. Su, and T.-A. Nguyen, "Permis: a modular authorization infrastructure," *Concurrency and Computation: Practice and Experience*, vol. 20, no. 11, pp. 1341–1357, 2008.

[26] The OASIS technical commitee, "XACML: eXtensible Access Control Markup Language," https://www.oasis-open.org/committees/xacml/, 2005.
[27] L. L. Shi and D. W. Chadwick, "A controlled natural language interface for authoring access control policies," in *SAC*, W. C. Chu, W. E. Wong, M. J. Palakal, and C.-C. Hung, Eds. ACM, 2011, pp. 1524–1530.
[28] N. Zannone, M. Petkovic, and S. Etalle, "Towards data protection compliance," in *SECRYPT*, S. K. Katsikas and P. Samarati, Eds. SciTePress, 2010, pp. 213–216.
[29] S. Trabelsi, A. Njeh, L. Bussard, and G. Neven, "PPL Engine: A Symmetric Architecture for Privacy Policy Handling," *W3C Workshop on Privacy and data usage control*, 2010.
[30] S. Trabelsi, G. Neven, and D. Raggett, *PrimeLife Deliverable D5.3.4: Report on design and implementation*, 2011.
[31] Joint NEMA/COCIR/JIRA Security and Privacy Committee (SPC), "Break-glass: An approach to granting emergency access to healthcare systems," 2004.