Run-Time Monitoring of Data-Handling Violations

Jassim Happa[1], Nick Moffat[2], Michael Goldsmith[1], and Sadie Creese[1]

Department of Computer Science, University of Oxford, UK
[1]firstname.surname@cs.ox.ac.uk, [2]nick@sysverity.com

Abstract. Organisations are coming under increasing pressure to respect and protect personal data privacy, especially with the European Union's General Data Protection Regulation (GDPR) now in effect. As legislation and regulation evolve to incentivise such data handling protection, so too does the business case for demonstrating compliance both in spirit and to the letter. Compliance will require ongoing checks as modern systems are constantly changing in terms of digital infrastructure services and business offerings, and the interaction between human and machine. Therefore, monitoring for compliance during run-time is likely to be required. There has been limited research into how to monitor how well a system respects consents given, and withheld, pertaining to handling and onward sharing. This paper proposes a finite state machine method for detecting violations of preferences (consents and revocations) expressed by Data Subjects regarding use of their personal data, and also violations of any related obligations that might be placed upon data handlers. Our approach seeks to enable detection of both accidental and malicious compromises of privacy properties. We also present a concept demonstrator to show the feasibility of our approach and discuss its design and technical implementation.

Keywords: Privacy, Run-Time Monitoring, Policy-Violation Checking.

1 Introduction

As legislation and regulation evolve to incentivise protection of personal data, so too does the business case for demonstrating compliance to the privacy requirements of "Data Subject" (DS). Some believe there is also an ethical obligation on enterprises to consider how to enable individuals to take better, more informed responsibility for the sharing of their personal data, and to support consent and revocation. Whether driven by law, ethics or competitive advantage, we expect demonstration of compliance will become a necessary component of future security and privacy governance and operations. Indeed this is sign-posted clearly by the General Data Protection Regulation (GDPR) [12], and GDPR impacts organisations that handle EU citizens' personal data beyond EU boarders.

It is necessary to evolve the risk-management methods employed by enterprises to ensure that they can manage their own risks associated with personal-data handling whilst also supporting enhanced individual-centric controls. Personal data (e.g. date of birth, address, name, likes, photos, etc.) can be thought

of as a commodity with service providers holding a licence to use it, but usually not to use it without the express consent of the person who is the subject of the data, the DS. An important enabler for compliance will be to monitor for compliance violations, enabling organisations to detect and analyse violations and so understand how they occur and how to prevent them recurring.

1.1 Contributions of the paper

It is necessary to evolve risk-management methods to ensure that enterprises can manage their own risks associated with handling personal data whilst also supporting enhanced individual-centric controls. We base our work on objectives from the EnCoRe project [11]. This paper considers the challenge of **designing a compliance-monitoring framework for detection of data-handling violations in real time**. Specifically, our focus is on violations of the personal-data use preferences (consents and revocations) expressed by DSs, and also violations of any related obligations that might be placed upon data handlers. To the best of our knowledge, the work in this space is in designing policy and policy languages for data handling against criteria, but not in detecting where these policies and supporting technologies have failed. Our approach is loosely inspired by Intrusion Detection Systems (IDSs) [26], and adds a finite-state machine to identify potential misuses of personal data. From a concept demonstrator, we outline how false positives and false negatives may occur as well as mitigation strategies to minimise them.

2 Related Work

Standards bodies, legislation and regulation have ruled that organisations must respect the privacy of individuals. Examples include OECD [21], EU [19], UK DPA [4]. Updates to the EU regulations regarding are making this requirement more explicit, particularly with the enforcement of the General Data Protection Regulation (GDPR) in 2018. The GDPR embraces 'privacy by design' without detailing how it can or should be applied [16].

We believe that what is required is a run-time monitoring approach that if designed into systems handling personal data would provide a genuine privacy by design feature capable of taking account of all systems and service evolutions – which supports data usage as opposed to preventing it, and will continue to work no matter how the system changes. This is a property not achievable through verification of system-component integrity and their behaviours alone.

Many commercial products exist, but these are mainly focused on compliance with information security regulation and standards such as Sarbanes-Oxley [24] and ISO27001. There are no equivalent products focused on privacy and particularly compliance with consents given regarding data handling. This is likely to be because the common practice is to seek blanket consents with limited ability for change (by users or DSs).

The EU project COMPAS has developed a business compliance framework for *Service-Oriented Architectures* (SOAs), specifically for process-driven SOAs [9, 27]. It uses *Complex Event Processing* (CEP), achieved using state machines, to recognise patterns of 'low level' events performed by the monitored system [18]; low level events are system-level events that have significance for compliance. A pattern captures one way the system might violate a specific compliance requirement. CEP signals the occurrence of any such pattern using a corresponding 'high level' event. In particular, Mulo [20] describes how to map business activities to so-called 'event trails' and thence to CEP queries/rules. It is a model-aware approach in the sense that monitors have run-time access to models of correct behaviour. Monitors designed using the COMPAS approach aim to check compliance to particular business processes. In contrast, we seek monitors that check for satisfaction of particular 'compliance criteria' (which we will define) concerning the efficacy of controls made available to DSs in relation to the privacy of their data. Our approach is also based on state machines.

Liu et al. [17] describe a static compliance-checking framework targeted at showing that executing business processes satisfy certain specifications. This involves transformation of *Business Process Execution Language* (BPEL) models to pi-calculus processes, and model checking of these processes against LTL models derived from BPSL (a specification language for business processes). The work focuses on business process compliance and not on monitoring actual data-flow compliance as we concern ourselves with here. Garg et al. [15] present an algorithm called: "*reduce*", that checks audit logs for compliance with privacy and security policies. The paper proves correctness, termination, time and space complexity results of reduce. Chowdhury et al. [8] outlines an approach to temporal mode-checking for run-time monitoring of privacy policies by checking online event trace compliance from caching satisfying instances when it can and fall back to brute force checking when it cannot.

Basin et al. [3] states that existing logic-based policy monitoring is currently limited in their support for aggregations. They take inspiration from aggregation operators found in database query languages like SQL develop a monitoring algorithm for this language. Basin et al. [2] proposes an approach to identify a purpose (of data) with a business process, and show how formal models of interprocess communication can be used to audit or even derive privacy policies. From this assumption, they then propose a methodology for auditing GDPR compliance from a interprocess dataflow model, aspects of GDPR compliance can be determined algorithmically. They also highlight aspects that cannot become GDPR compliant by algorithmic means (i.e. where human action is required). This is an interesting complementary work to ours, which assumes one can design and implement a correct run-time monitor (such as might be achieved using our method) and then investigates efficiency in features.

Other related work that is not aimed at similar run-time monitor designs include: Soto-Mendoza et al. [25] proposed a mechanism to compose privacy policies based on semantic-web technologies. Their composition of rules is based on the data usage context and deduces implicit terms. Their approach uses basic

operators and ontology-based rules to consider data-usage context. The authors point out that inconsistencies can be minimised with contextual rules that incorporate priorities. Barth et al. [1] explore contextual integrity by proposing a conceptual framework for understanding privacy expectations and their implications. They formalise a logical framework for expressing and reasoning about norms of transmission of personal information. Datta et al. [10] describe a semantic model that is designed with the goal of enabling specification and enforcement of practical privacy policies. The model consists of a set of interacting agents in roles who perform actions involving personal information in a given context. It is then possible to use traces where each trace is an alternating sequence of states and actions performed by agents that update state.

Privacy-by-design is an approach to systems engineering with seven key principles aimed at taking human values, such as privacy into account in a system's design [6]. The Privacy Management Reference Model (PMRM) [7] is one example of a methodology for understanding and analysing privacy policies and their privacy management requirements in defined use cases. The National Institute of Standards and Technology (NIST) [5] discusses the concepts of privacy engineering and risk management for federal systems and aims to establish the basis for a common vocabulary to facilitate better understanding and communication of privacy risk within federal systems. Fisk et al. [14] define three engineering privacy principles that guide sharing security information across organisations: Least Disclosure, Qualitative Evaluation, and Forward Progress.

3 Establishing Monitoring Requirements

3.1 Assumptions

We assume the system being monitored provides DSs with an ability to express constraints on the handling of their personal data via *Consent and Revocation* (C&R) controls [11]. We say a DS chooses (or makes) particular 'C&R choices' from among available 'C&R options' presented by the monitored system (as dictated by the enterprise and service being operated), where data includes all 'personally identifiable information' pertaining to the DS. We assume enterprises seek to respect the C&R choices made by each DS, within the bounds of the law (i.e. unless a legal warrant will make an enterprise overlook a DS's preference "*not-to-share*"). We allow for the possibility that certain obligations are placed on data handlers regarding how revocation functionality is delivered, whether onward sharing of data is permitted (and to what degree), and how the DS must be kept informed of any data handling.

Our particular focus is on monitors that signal violations of specific types of C&R choice, rather than all types of choice that might arise. We define the control flows related to particular forms of data sharing, parametrized by variables capturing specific instances. The monitors we design then detect in real time any relevant data flows that might violate the wishes of the individual. We recognise that latency within the system could result in false positives, and have

developed a strategy for reducing them. In this way we seek to enable detection of accidental or malicious compromises of privacy properties. We describe a concept demonstrator that shows the potential capability of such a system.

We consider how a specific monitor to be deployed must be informed by the data available for collection on a system, and present a general architecture for sensor placement which can be mapped easily onto multiple conforming architectures. The service principles we use are a subset of those adopted by the EnCoRe project [22, 11] (a research project that focused on establishing a logic for how to handle consent and revocation of data from data subjects), which are designed to meet privacy law and regulation.

3.2 Service Principles and Compliance Criteria

Relevant guiding principles to monitor privacy-compliant systems are given below. We consider these to be key to best practice and highly relevant to the satisfaction of privacy regulations in general:

- **Revocation Management**: DSs must be able to revoke previously given consents (explicit or otherwise). Service providers must provide a declared minimal revocation functionality, and respect and act upon all revocation requests except to the extent that the law mandates otherwise.
- **Service Responsiveness**: Clear commitments must be made with regard to availability of service and the speed with which changes in preferences (new consents and revocations) will be implemented. DSs must be offered the facility to be informed whenever the service does not meet pre-specified commitment levels, and of the nature of any resulting non-consented data exposure.
- **Choice Flow-Down**: Data passed between systems will be protected such in that the DS's consent and revocation choices are respected by the receiving party. Projected choices at least as restrictive will be respected as the DS's will accompany the data[1].

These principles can be satisfied by a number of criteria that should be met by compliant systems, and which can be used to determine events to monitor. These criteria are generic in the sense that they are independent of the nature of the service or any particular technology platform:

1. Where neither explicit nor implicit consent has been given for storage / processing / sharing of particular personal data, the data should not be used in this way. All revocations of consent must be supported and respected by the system (except where not permitted by law).

[1] For the types of CR considered in this paper, projection amounts to removal of 1-step sharing consents. This enables their interpretation at receiving systems without regard for where the data and choices came from. If, on the other hand, projection is trivial (the identity function) then only original choices are ever communicated, which would mean they must be interpreted according to whether the data was received directly from the DS or instead from an upstream system.

2. A published commitment to service performance must be made, specifically including speed of response in acting fully on new choices and changes in choices, in both cases for explicit choices and implicit choices. Furthermore, the speed-of-response commitment must be reasonable, actual service performance relative to commitments must be monitored, and DSs. must from the outset be given the option to be notified when violations occur.
3. When service providers pass personal data to third parties, they must ensure that DSs' consent and revocation choices are passed on with the data and they must seek to protect the data in accordance with these choices. Forwarding of projected choices should be mandatory, which for us are original choices strengthened by removal of 1-step sharing consents.

3.3 A Simple Architecture

Suppose there are some systems, such as the one to be monitored, that are joined together in a chain. We focus on personal data pertaining to a particular DS, and suppose for simplicity that this data may be passed unchanged from DS to one or more systems in turn, one of which is the monitored system. So personal data originates at DS and may pass along a chain of systems. This is depicted in Figure 1 with DS on the left and data passing only left-to-right. The analysis extends straightforwardly to trees. Suppose DS makes some choices about how his provided data may be processed and/or shared – C&R choices – and that any such choices pertaining to particular data should be passed on faithfully whenever the data is passed between systems (where no explicit choices have been made we suppose default choices have been presented to DS and he has accepted them.) We allow DS to provide new data and choices at any time, or new choices pertaining to data provided previously. We suppose the architecture of the monitored system is as shown in Figure 2. This architecture contains three components:

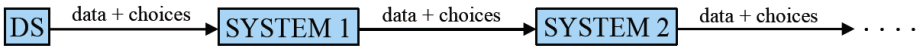


Fig. 1. A system of systems that may pass a particular DS's data and choices along a chain of systems, where one particular system is to be monitored.

- "**Application**" represents the essential functionality provided by the system, though unconstrained by any Consent and Revocation (C&R) choices that arrive.
- "**DR**" stands for Data Registry and represents a store for C&R choices.
- "**Decider**" represents a component that decides whether to permit or deny requests to process or share particular data (access requests).

The application could be a legacy system onto which are added C&R controls implemented using the Data Registry and Decider. The arrow numbering shows

the intended sequence of messages. First, some data and associated choices arrive at the Application component from upstream (either the DS or the closest upstream system). Then a reference to the data is passed, along with the choices, to the DR for storage. Some time later the Application generates an Access Request (AR) to request permission to handle the data in a certain way, and sends this AR to the Decider, which then requests the relevant choices from the DR. On receiving the response, the Decider decides whether to allow the requested access, sending either a permit message or a deny message to the Application. Finally, if the request was to share the data and a suitable permit message was received, the data and associated C&R choices may be passed downstream. Note that we could alternatively assume that data and choices are passed directly to the Data Registry. This would require only small changes to the analysis presented here.

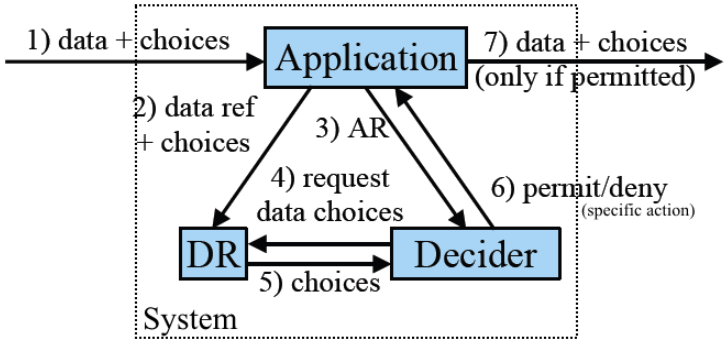


Fig. 2. Architecture of the monitored system, with an intended sequence of messages.

3.4 Sensor Locations

Naturally a monitor for checking end-to-end behaviour of the system would monitor system inputs (message 1 in Figure 2) and outputs (message 7) and compare the two, looking for unacceptable patterns of these events over time. We call this a "*1:7 monitor*". In the case of the monitors developed here, 'unacceptable patterns of events' means behaviour in violation of C&R choices. Given in-depth knowledge of a system's architecture, it becomes feasible to define further monitors, each corresponding to a particular choice of events to monitor.

Any such choice determines the types of sensors needed and the locations at which to place them. In the following we will focus on specifying "*1:6 monitors*", i.e. monitors that look for violations of C&R choices as evidenced by patterns of system inputs and messages from the Decider to the Application that permit or deny individual data handling requests.

A 1:6 monitor alone gives only partial protection because the Application might misbehave by releasing data when not permitted by the Decider. However, it is fair to expect non-malicious service providers to take steps to avoid

Application misbehaviour, by implementing the Application to respect the Decider's decision (using a simple 'final-gate' check, probably easier to assure than the full Application) and/or by using a separate 6:7 monitor. A separate 6:7 monitor may also be suitable in the case of a malicious service provider, or simply a single 1:7 monitor (not relying on the Application respecting the Decider's decision, or even on the system being architected as supposed above). Suitable placement and configuration of monitor components would of course be required for any monitoring solution to be certified as acceptable.

The system handles each DS's personal data, and gives permission for its handling by the operational environment only as allowed by the current records pertaining to that data. The run-time monitoring requirement is to monitor system data-flows at the point decisions are communicated internally (output of the Decider component) and at the point of interface with third parties to check onward sharing, and to compare such flows with the choices retrieved from the repository.

4 Defining Individual Monitors

Individual monitors focus on particular types of data handling: *processing*, *sharing one step*, and sharing in a way that allows *sharing 'downstream'* (see respective sections below). These notions are explained in the following subsections. C&R choices available to a DS amount to his consenting to certain types of data handling (for particular data) or his revoking of such consents.

4.1 Data Processing

The state machine in Figure 3(a) specifies a very simple data processing monitor. It is only concerned about processing of a particular datum (item of data) d. The initial state is at the top left of the figure. The state machine observes (events denoting) consent and revocation actions pertaining to processing of d and also to the monitored system giving permission to process d. Events are written using a CSP[2]-like notation [23]. These forms of event are involved:

- **consent.Process.b.d** = the DS gives consent for b to process d;
- **revoke.Process.b.d** = the DS revokes this consent;
- **permit.Process.b.d** = the monitored system gives local permission for b to process d;
- **violation.Process.b.d** = the monitor raises an alert reporting that b has incorrectly given permission to process d.

Here b, c = identifiers for individuals, while d, d' = identifier for datum. The dots (".") within an event separate it into distinct fields, where the first field is known as the channel. So the events of this state machine occur on the channels consent, revoke, permit and violation. Notice that events on the consent channel

[2] Communicating Sequential Processes

denote giving of consent to the monitored system, whereas those on the permit channel denote the monitored system giving local permission for a particular instance of processing. The two states at the top of Figure 3(a) keep track of whether or not permission to process (given locally by the system) is acceptable given the earlier consents and revocations; the intention is to raise an alert when the system incorrectly gives permission to process d. Both top states allow the permission to be given, but the monitor reacts differently according to its state at the time: if in the right-hand state it simply performs a self-loop (silently accepting the giving of permission); if in the left-hand state it moves to the bottom state and can then only report violation.

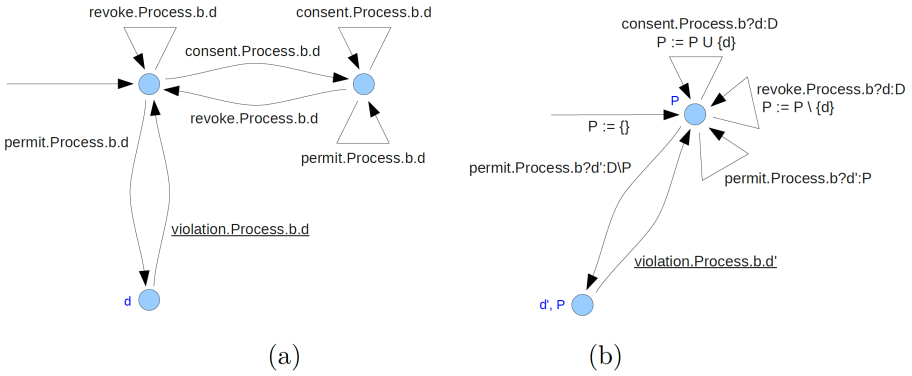


Fig. 3. (a) ProcessDatum(b,d) models compliance monitoring of principal b for processing of a single item of data d. (b) ProcessData(b,D) models compliance monitoring of principal b for processing any elements of a data set D.Underlining in figures denotes an output event; the rest are inputs events to it ('output by'/'internal to' the system).

Figure 3(b) extends the previous state machine to one specifying a monitor for processing any data in a given data set D. This machine uses state variables (sometimes called parameters) to capture some aspects of state. In particular, it maintains a state variable P, which records the set of data items for which local permission to process is acceptable. The bottom node also uses state variable d', which records the data for which permission was (erroneously) given. Each state of this machine is represented partly by an explicit node of the machine and partly by the values of any state variables that annotate the node.

The same forms of event are involved in this machine as in that of Figure 3(a), though the transition label notation now involves question marks "?") to denote input of a value from a specified set of values, in particular "*?d:D*" denotes input of the datum d from the data set D. Transitions with labels involving "?" are shorthand for multiple transitions, each labelled by a particular event where the input variable (d or d' in Figure 3(b)) has been replaced by a particular value from the selection set (D, D\P, or P in the figure) where '\' is setminus.

By maintaining the variable P the machine in Figure 3(b) avoids having to move between explicit nodes to keep track of when particular local permissions are acceptable. P is always a subset of D. It is initialised to the empty set, as shown in this Figure by the action "$P := \{\}$". P expands and contracts as actions are performed that correspond to consents and revocations. Occurrence of any unacceptable permission event moves the machine to a state in which an alarm is then raised; these about-to-alarm states are all represented by the bottom node. (State variable d' is used in these alarms and promptly forgotten, while P is maintained at all nodes.)

Machines with at least one state variable are known as symbolic and those without are known as explicit. Use of state variables does not increase expressiveness but is a notational convenience that enables a much more succinct graphical representation than would be possible with explicit state machines.

4.2 Data Sharing: One Step

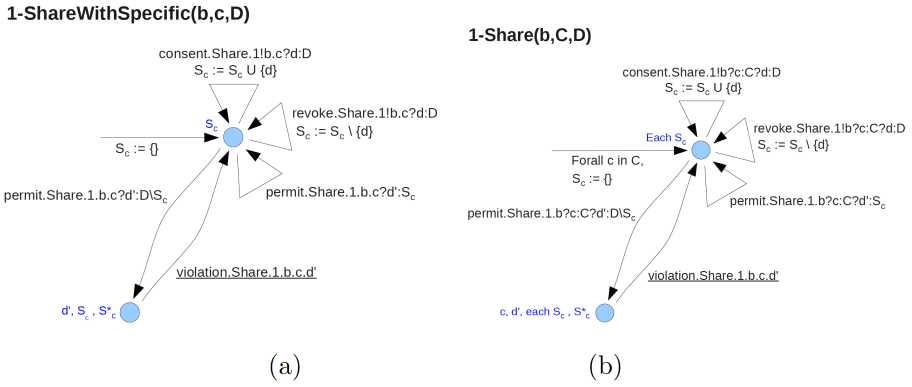


Fig. 4. Monitoring single-step sharing of elements of a data set D (a) with a specific third party c, or (b) with elements of a fixed set C of third parties.

By consenting to one-step sharing the DS permits the recipient to process the data locally and to share it just a single step (in such a way that the next recipient down the chain is permitted to process the data but not to share it). The following new forms of event are involved:

- **consent.Share.1.b.c.d** = the DS gives consent for b to share d with c;
- **revoke.Share.1.b.c.d** = the DS revokes this consent;
- **permit.Share.1.b.c.d** = the monitored system gives local permission for b to share d with c;
- **violation.Share.1.b.c.d** = the monitor raises an alarm reporting that b has incorrectly given permission to share d with c.

Figure 4(a) specifies a monitor for b's one-step sharing of data in D with a specific third party c. Exclamation marks ("!") in the transition labels indicate output of some data. In the state machines shown they are equivalent to dots. Figure 4(b) extends this machine to one for sharing with third parties chosen from a fixed set C.

4.3 Data Sharing: Multiple Step

Figure 5 specifies a monitor for sharing transitively with any third party in C, where "transitive sharing" means enabling the recipient to process the data locally and to share it onward just a single step (if he so chooses) or transitively (enabling the next recipient in the chain to share similarly). Although transitive sharing enables multi-step sharing, it is a consent action between only two principals: the data owner and b here, though in a next step b and a principal with whom b chooses to share transitively.

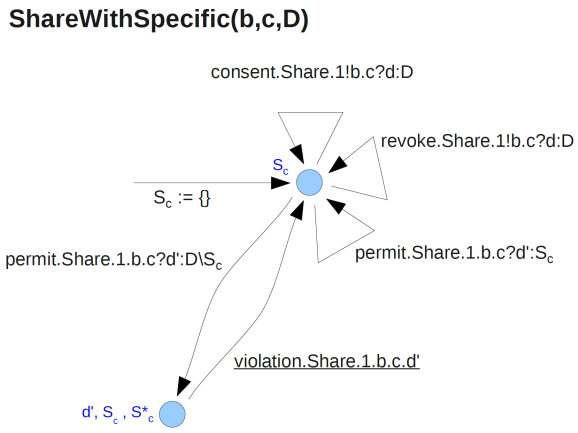


Fig. 5. Monitoring transitive sharing of elements of a data set D with elements of a set C of third parties.

5 Composing Monitor Specifications

The individual monitors of Section 4 (or variants of them) can be composed together to yield a monitor capable of reporting any and all of the violations addressed by the individual monitors. Individual monitors described thus far use sets P, S_c and S_c^* having simple interpretations: at all times these sets contain exactly those data items d that the monitored system has permission to process, to share one step (only) or to share transitively. In our discussion we omit subscript c, supposing a particular third-party c is understood.

In many situations it is possible to view consenting to transitive sharing as also consenting implicitly to one-step sharing and processing, and consenting to one-step sharing as also consenting implicitly to processing. It is reasonable to view revoking consent to process as also revoking consent to share one step and revoking consent to share transitively, and to view revoking consent to share one step as also revoking consent to share transitively.

There is thus a natural subset ordering between the sets P, S_c and S_c^* if we continue to use these sets to record precisely when the monitored system has permission to process (in the case of set P), to share one step (in the case of set S), or to share transitively (in the case of set S^*): i.e., $S^* \subseteq S \subseteq P$. Unfortunately this simple interpretation of the sets P, S_c and S_c^* would require each monitor to observe all those consents relevant to these conditions, e.g. the monitor that maintains set P would have to observe all consent events for data sharing, whether one step or transitively (since these are taken to imply consent to process). Similarly, the monitor that maintains set S would have to observe consents to share transitively and also revocations of processing permissions.

We choose to adjust the meaning of the sets P, S_c and S_c^* to reduce the complications needed when composing monitors: we continue to specify that the individual monitors observe precisely those consents and revocations pertaining directly to processing, to one-step sharing, or to transitive sharing, but now interpret the sets as recording whether the corresponding individual monitor would report a violation from its perspective. With this interpretation, after any sequence of Cs and Rs the composite monitor will report a violation if any individual monitor does so. This can be achieved by synchronizing individual monitors on permit events and leaving them to interleave on all others.

6 Accounting for Delays

The simple monitors discussed thus far make no allowance for system latencies. Consequently they can generate false positives (raising alarms when not appropriate due) and false negatives (failing to raise alarms when alarms should be raised) – in both cases because the monitors may judge acceptability of processing/sharing according to out-of-date C&R choices. We now study this issue and attempt to extend the monitors to cope better. Recall that we suppose the monitored system to be architected, see Figure 2. Each component and each communication path will introduce some delays into the system, causing system latency. Communication delays in the type of system we consider are likely to be very small compared with delays across components, so they may be expected to contribute relatively little to system latency. So for a first approximation we may reasonably disregard the communication delays, or include them in the component delays. For example, in Figure 6 any delay between the Application sending message 2 and that message being received at DR will be treated as part of the Application's delay.

Recall further that we assume the existence of a published commitment to service performance. It would be sensible for the organisation to allow in this

commitment for reasonable delays in processing and communication. We accept the possibility that a change of C&R choices may be implemented in a staged fashion – it would be quite demanding to insist that all changes received together are implemented together simultaneously – but we insist that all data is handled at all times according to at least some complete set of choices expressed by the DS up to some reasonably recent time. Accordingly we propose the notion of 'recent snapshots', where a 'snapshot' captures all the latest choices at some time and a snapshot is 'recent' if its time is at least as recent as necessary to satisfy the service performance commitment. A recent snapshot need not be the most recent snapshot, but it must not be too old (we call any that are too old 'stale'). The service performance agreement should make clear exactly when snapshots would become stale, and we would expect a service provider to offer commitments to service levels according to their understanding of likely system latency, and to propose a notion of "recent" which they intend to deliver against.

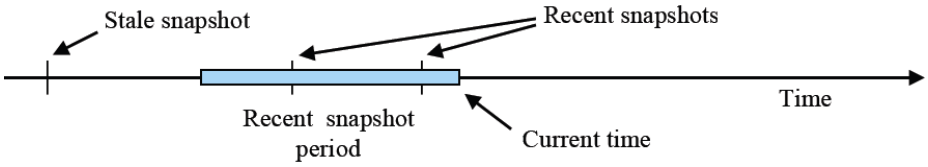


Fig. 6. A recent snapshot is a complete record of a DS's latest C&R choices up to a recent point in time; the time period is chosen to satisfy the service performance agreement. This Figure shows three points at which snapshots were taken, but only the later two are considered to be recent snapshots.

We call the monitors defined in earlier sections "latest-choice monitors" as they work w.r.t. the most recently made relevant C&R choices. Let "recent-choice monitor" mean a monitor that judges action events ("permit" decisions in the case of 1:6 monitors) in accordance with at least one recent snapshot, so requiring a relevant consent EITHER within the "recent snapshot period" OR before it and with no revocation occurring after it. We anticipate that recent-choice monitors can be obtained from corresponding latest-choice monitors by:

1. (additionally) maintaining certain state information that enables recently-consented-to activities (processing or sharing) to be determined even when these are not permitted by the most recent snapshot (i.e. the latest complete set of C&R choices);
2. using this extra state information, not the most recent snapshot, to judge acceptability of action events.

7 Concept Demonstrator

We built a concept demonstrator with an analyst in mind who is responsible for reporting consent and revocation violations. These reports are intended to be

used to incentivise protection of DS personal data, but also for demonstrating compliance. Our standalone application assumes an analyst monitors all DS permissions on a service. Implementing our method requires a staged process using the following steps. It is necessary to:

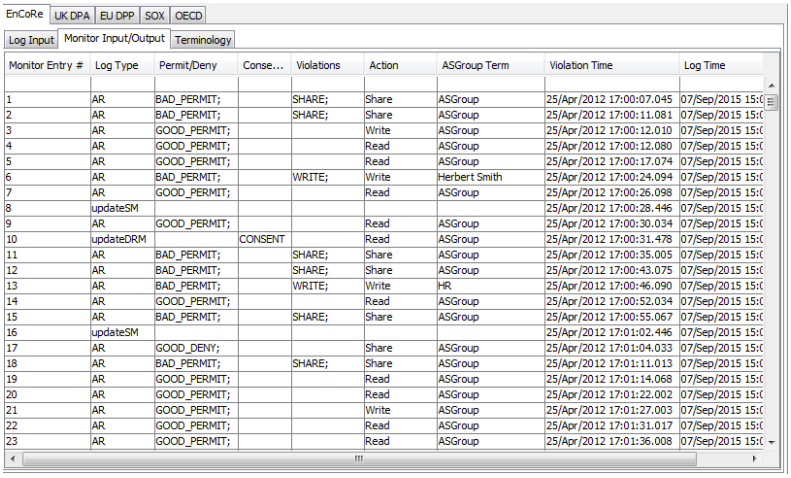
1. determine types of system events relevant to the privacy properties addressed;
2. use generic privacy monitors that notice the occurrence of these events and announce violations deemed to have occurred;
3. choose particular architectural locations at which to place sensors to detect these events;
4. implement a generic monitor fed by sensors placed at the chosen locations.

Any particular choice of locations yields a monitor for the generic property but specialised to these locations. Multiple monitors could be deployed simultaneously. The choice of which monitors to deploy, and at what locations, should be driven by risk assessments.

The tool was built in Java using Swing, JFreeChart and JUNG libraries to handle GUI components. The tool has four main components: the data parsers, the monitors themselves, the archiver and a visual dashboard. The monitors check for violations from parsed events, the archiver stores the outcome of the monitor checks for archiving purposes and finally the dashboard presents the output of the monitors. The monitor creates violation logs that the visualizations make use of. In the future, we envisage monitors could accommodate for enhanced understanding and investigative purposes. The tool parses log files as they appear in a folder, sends the content of these assumed to be in the correct order to a data handler. The data handler stores a current set of actions, whether these be permission changes or requests to read, write or share personal data, and passes them to monitors. Once the monitors raise violations, these are both written to a file for archiving purposes, but also passed to a manager that sends content to its relevant visualization panes and presented to the end-user individual, either the DS or an analyst. For our demonstrator, we ran simulations to synthesize EnCoRe events by maintaining a list of permissions per DS and created a list of how permissions had been handled by an existing data-handling system. Our implementation then checked that list of actions against a DSs permission profile using our method. We ran several simulations on our concept demonstrator to show a proof of concept with several thousand permission requests with a handful of DSs (including groups of DSs). Our simulation did not take delays into account.

We built a state machine to check for read, write and share violations monitors. (Note: this is share once, we do not control for whether data that has been shared once and reached outside our system is shared further). **At the simplest level, we check if the Access Subject [11] exists, then check the action intended to be performed, then check purpose of said action, then check the parameter of the purpose and action. If all these access request checks report non-violation, the monitor assumes there to be no violations of the event.** The states we can return are akin to those

described by Fawcett [13]. However, instead of reporting False Positive, False Negative, True Positive and True Negative, our monitors check events to be **GOOD PERMIT, BAD PERMIT, GOOD DENY, BAD DENY** (see Figure 7), and finally **MAINTAIN** when no event is occurring.



Monitor Entry #	Log Type	Permit/Deny	Conse...	Violations	Action	ASGroup Term	Violation Time	Log Time
1	AR	BAD_PERMIT;		SHARE;	Share	ASGroup	25/Apr/2012 17:00:07.045	07/Sep/2015 15:(
2	AR	BAD_PERMIT;		SHARE;	Share	ASGroup	25/Apr/2012 17:00:11.081	07/Sep/2015 15:(
3	AR	GOOD_PERMIT;			Write	ASGroup	25/Apr/2012 17:00:12.010	07/Sep/2015 15:(
4	AR	GOOD_PERMIT;			Read	ASGroup	25/Apr/2012 17:00:12.080	07/Sep/2015 15:(
5	AR	GOOD_PERMIT;			Read	ASGroup	25/Apr/2012 17:00:17.074	07/Sep/2015 15:(
6	AR	BAD_PERMIT;		WRITE;	Write	Herbert Smith	25/Apr/2012 17:00:24.094	07/Sep/2015 15:(
7	AR	GOOD_PERMIT;			Read	ASGroup	25/Apr/2012 17:00:26.098	07/Sep/2015 15:(
8	updateSM						25/Apr/2012 17:00:28.446	07/Sep/2015 15:(
9	AR	GOOD_PERMIT;			Read	ASGroup	25/Apr/2012 17:00:30.034	07/Sep/2015 15:(
10	updateDRM		CONSENT		Read	ASGroup	25/Apr/2012 17:00:31.478	07/Sep/2015 15:(
11	AR	BAD_PERMIT;		SHARE;	Share	ASGroup	25/Apr/2012 17:00:35.005	07/Sep/2015 15:(
12	AR	BAD_PERMIT;		SHARE;	Share	ASGroup	25/Apr/2012 17:00:43.075	07/Sep/2015 15:(
13	AR	BAD_PERMIT;		WRITE;	Write	HR	25/Apr/2012 17:00:46.090	07/Sep/2015 15:(
14	AR	GOOD_PERMIT;			Read	ASGroup	25/Apr/2012 17:00:52.034	07/Sep/2015 15:(
15	AR	BAD_PERMIT;		SHARE;	Share	ASGroup	25/Apr/2012 17:00:55.067	07/Sep/2015 15:(
16	updateSM						25/Apr/2012 17:01:02.446	07/Sep/2015 15:(
17	AR	GOOD_DENY;			Share	ASGroup	25/Apr/2012 17:01:04.033	07/Sep/2015 15:(
18	AR	BAD_PERMIT;		SHARE;	Share	ASGroup	25/Apr/2012 17:01:11.013	07/Sep/2015 15:(
19	AR	GOOD_PERMIT;			Read	ASGroup	25/Apr/2012 17:01:14.068	07/Sep/2015 15:(
20	AR	GOOD_PERMIT;			Read	ASGroup	25/Apr/2012 17:01:22.002	07/Sep/2015 15:(
21	AR	GOOD_PERMIT;			Write	ASGroup	25/Apr/2012 17:01:27.003	07/Sep/2015 15:(
22	AR	GOOD_PERMIT;			Read	ASGroup	25/Apr/2012 17:01:31.017	07/Sep/2015 15:(
23	AR	GOOD_PERMIT;			Read	ASGroup	25/Apr/2012 17:01:36.008	07/Sep/2015 15:(

Fig. 7. Monitoring Log Input/Output as shown in the visualization dashboard.

Figure 8 shows the dashboard. Its aim is to show where are violations happening, the health status of the system, what violations relate to particular DSs (Lists), what is the distribution of violations is over time, and finally, what is the network of total violations is. Figure 8 shows the default visualizations: Architecture, Lists, Plot and Graphs. These have been selected for the purpose of communicating the key questions relevant to understand violations, including: "Where are violations happening (on the Architecture)?", "What violations relate to particular DSs (Lists)?", "What is the distribution of violations over time (Plot)?" and "What is the network of total violations (Graph)?"

To the left of the diagram, there is the Tool Bar. The Tool Bar contain various configuration buttons such as refresh/play button (refresh a data capture or continue data input), stop button (stop the intake of new events), zoom-in button (let the last clicked visualization occupy the whole visualization space (as opposed to a quadrant)), zoom out button (show four visualizations), snapshot dump button (create a screen shot and a textual dump of the current state of the visualization), and an exit button (quits the tool). The Menu Bar (top of the screen) contains the same options as the Tool Bar.

These have been added for the purpose of communicating the key questions deemed relevant for understanding violations, including: Where are violations happening (on the Architecture)? What violations relate to particular DSs

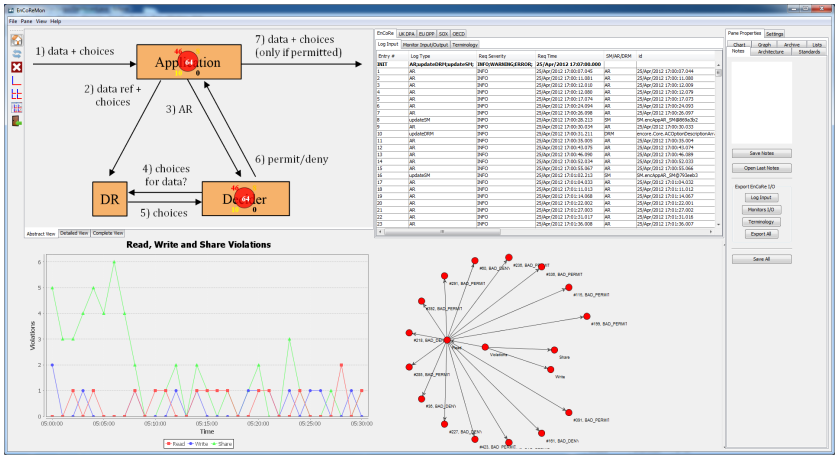


Fig. 8. Screenshot of concept demonstrator.

(Lists)? What is the distribution of violations over time (Plot)? What is the network of total violations (Graph)? To the left of the diagram, there is the Tool Bar. The Tool Bar contain various configuration buttons such as refresh/play button (refresh a data capture or continue data input), stop button (stop the intake of new events), zoom-in button (let the last clicked visualization occupy the whole visualization space (as opposed to a quadrant)), zoom out button (show four visualizations), snapshot dump button (create a screen shot and a textual dump of the current state of the visualization), and an exit button (quits the tool). The Menu Bar (top of the screen) contains the same options as the Tool Bar. The Control Panel (right-hand side) contains the parameters for each visualization, but also the program as a whole. The Control Panel also has the ability to export notes that analysts and DSs make during usage of the tool. Each of visualization pane inherits from a generic pane that describes the basic aspects of what a visualization has to contain, such as listeners from the control panel and monitor, tool bar and menu bar input.

8 Discussion and Future Work

Privacy Impact Assessments presently mostly take no account of run-time performance or the evolution of systems between assessment periods. We believe our method can supplement the existing frameworks of principles by adding to engineering approaches. We believe this engineering space will eventually coalesce to three key ideas:

– **creating and maintaining company policies** that adhere to legislative frameworks and non-disclosure agreements. These policies should allow for developers to understand what data inputs go into a system in the first place, and how the data is handled at the business-decision level. E.g. GDPR and

purpose limitation of data collection – data should not be collected for any other purpose than its original intention, and certainly not behind the scenes with the data owner not knowing.
- **creating and maintaining software that adheres to privacy by design principles**, with both the design and implementation of any system being developed with privacy as a *feature*, and *not as a limitation* of new and existing systems.
- **creating and maintaining run-time monitoring systems** that provide added levels of assurance to *document how data has been handled*, but also *enforce data-handling rules* in the interest of DSs as well as the business. Run-time monitoring of compliance to data-handling requirements of DSs may help organisations manage the risk they are exposed to should they act contrary to requirements of any system.

One may consider our approach to be privacy monitoring, as the particular properties we aim to detect would be risks for the privacy of personal data that indicate violations of associated preferences. We observe that there are necessarily performance limitations associated with such a monitoring approach, as system latency can introduce false positive alerts and create situations where violations are missed. However, we demonstrate that our method can be evolved to develop a sensor system that can take account of expected latency and in particular the service performance commitments, which should be developed in a manner cognisant of any expected latencies. Future work will include a detailed design for obligation monitoring, including extensions which allow us to detect violations of time-triggered notification requirements.

- **Information Sharing Enforcement.** In our approach we describe the detection process of permission violations. This is akin to an IDS, whereas there may be cases in which driving the monitors akin to an Intrusion Prevention System (IPS) may be more desirable. In such a system, we envisage key GDPR actions such as anonymisation, pseudonomisation, aggregation of data, but also simply dropping data will play vital roles moving forward to preserve privacy of DSs. We are currently exploring this in the PROTECTIVE project [3] for Cyber Threat Intelligence.
- **Scalability.** Our implementation focused on a locally-hosted solution, in which throughput performance concerns would unlikely be observed. We tested the system with several thousand actions on a handful of DSs as well as a handful of group DSs [11]. We synthesized test data by maintaining a list of permissions per DS and created a list of how permissions had been handled by an existing data-handling system. Naturally, performance will greatly depend on implementation decisions. In production environments: volume, throughput and permission-checking capabilities of permission requests are essential to building a platform that is scalable, specifically: throughput performance, latency and utility.

[3] https://protective-h2020.eu/

- **More Detailed Sensor Architectures.** For any given system, further architectural details could be used to decide on further locations at which to place sensors, and the method described below could then be used to design corresponding monitors. For example, what we call the Decider is implemented in the EnCoRe technical architecture [22] using a set of *Privacy Enforcement Points* (PEPs) and a single *Privacy Decision Point* (PDP). PEPs act as gatekeepers: wherever data handling might occur they request permission to do it from the PDP. This strongly suggests placement of sensors within, or at the interfaces to, the PEPs and PDP when such an architecture is used. In principle it is possible to have sensors detect any intrasystem communications (even distinguishing between sends and receives) or internal processing (notably C&R choice storage or retrieval), and to define monitors that consume these sensor outputs.
- **Monitor Extensions**. *Strict Monitors* – The monitor represented by the state machine can be strict in the sense that it makes no allowance for delays within the system. Such delays are inevitable and could lead to false positives and false negatives. We should ensure the logged data appears (or at least is processed) in suitable order, i.e. ordered according to the right timestamps. *Lenient Monitors* – Conversely, the monitor represented by the state machine can lenient in the sense that it makes allowance for delays within the system. A tock for instance represents the passing of a unit of time, which may for example be a second. The tock can be used as a control self-loop i.e. it leaves the state machine in its current control state but has an associated action that has the effect of dropping any choices (consents, revocations) made since the start of the time window, and taking of account of them in a maintained 'Start of window Snapshot'. The test to decide whether to report a violation is replaced by a test for some suitable consent in the snapshot so consenting. We suspect it will also be necessary to consider consent periods, i.e. timeouts.
- **Usability Considerations**. Our framework assumes the DS is able to understand all of the access and sharing details and consequences and knows when to consent and revoke their consent, which may not always be the case. Future work should assess the usability of any implementation in order to propose best practices, including new visualization methods.

9 Conclusion

In this paper we have presented a novel approach to designing run-time privacy-compliance monitors using a simple finite-state machine to check for permission violations of the various preferences expressed by DSs. We also check for violations of any related obligations that might be placed upon data handlers. We used the EnCoRe [11] policy framework as the basis of method. We designed and implemented a demonstrator intended to be used similarly to IDS tools by analysts and outlined some of the benefits and remaining challenges in implementation. Finally, we also provided a discussion on the broader topic of run-time monitoring and its role moving forward. We believe monitoring of privacy

preserving mechanisms will be vital in the years ahead to feasibly demonstrate that 'privacy by design' is designed and implemented in digital systems both in spirit and to the letter, through appropriate documentation of permission violations (that can be reported to DSs more straightforwardly), but also through information sharing compliance enforcement.

Acknowledgement

This research was conducted as a part of the PROTECTIVE project. This project has received funding from the European Unions Horizon 2020 research and innovation program under grant agreement No. 700071. This output reflects the views only of the author(s), and the European Union cannot be held responsible for any use which may be made of the information contained therein.

References

1. Adam Barth, Anupam Datta, John C Mitchell, and Helen Nissenbaum. Privacy and contextual integrity: Framework and applications. In *Security and Privacy, 2006 IEEE Symposium on*, pages 15–pp. IEEE, 2006.
2. David Basin, Søren Debois, and Thomas Hildebrandt. On purpose and by necessity: Compliance under the GDPR. 2018.
3. David Basin, Felix Klaedtke, Srdjan Marinovic, and Eugen Zălinescu. Monitoring of temporal first-order properties with aggregations. *Formal methods in system design*, 46(3):262–285, 2015.
4. British Parliament. Data Protection Act. London Stationery Office, 1998.
5. Sean Brooks, Sean Brooks, Michael Garcia, Naomi Lefkovitz, Suzanne Lightman, and Ellen Nadeau. *An Introduction to Privacy Engineering and Risk Management in Federal Systems*. US Department of Commerce, National Institute of Standards and Technology, 2017.
6. Ann Cavoukian. Privacy by design. 7 foundational principles. https://www.ipc.on.ca/wp-content/uploads/Resources/7foundationalprinciples .pdf, 2011.
7. Ann Cavoukian, D Jutla, F Carter, John Sabo, Frank Dawson, J Fox, T Finneran, and S Fieten. Privacy by design documentation for software engineers version 1.0. *(PbD-SE) Burlington, MA: Organization for the Advancement of Structured Information Standards (OASIS)*, 2014.
8. Omar Chowdhury, Limin Jia, Deepak Garg, and Anupam Datta. Temporal mode-checking for runtime monitoring of privacy policies. In *International Conference on Computer Aided Verification*, pages 131–149. Springer, 2014.
9. Florian Daniel, Fabio Casati, Vincenzo D'Andrea, Emmanuel Mulo, Uwe Zdun, Schahram Dustdar, Steve Strauch, David Schumm, Frank Leymann, Samir Sebahi, et al. Business compliance governance in service-oriented architectures. In *Advanced Information Networking and Applications, 2009. AINA'09. International Conference on*, pages 113–120. IEEE, 2009.
10. Anupam Datta, Jeremiah Blocki, Nicolas Christin, Henry DeYoung, Deepak Garg, Limin Jia, Dilsun Kaynar, and Arunesh Sinha. Understanding and protecting privacy: Formal semantics and principled audit mechanisms. In *International Conference on Information Systems Security*, pages 1–27. Springer, 2011.

11. EnCoRe project partners. Encore: Ensuring consent and revocation. http://www.hpl.hp.com/breweb/encoreproject/index.html, 2008.
12. European Commission. General Data Protection Regulation. https://ec.europa.eu/commission/priorities/justice-and-fundamental-rights/data-protection/2018-reform-eu-data-protection-rules_en, 2018.
13. Tom Fawcett. An introduction to ROC analysis. *Pattern recognition letters*, 27(8):861–874, 2006.
14. Gina Fisk, Calvin Ardi, Neale Pickett, John Heidemann, Mike Fisk, and Christos Papadopoulos. Privacy principles for sharing cyber security data. In *Security and Privacy Workshops (SPW), 2015 IEEE*, pages 193–197. IEEE, 2015.
15. Deepak Garg, Limin Jia, and Anupam Datta. Policy auditing over incomplete logs: theory, implementation and applications. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 151–162. ACM, 2011.
16. Bert-Jaap Koops and Ronald Leenes. Privacy regulation cannot be hardcoded. A critical comment on the 'privacy by design' provision in data-protection law. *International Review of Law, Computers & Technology*, 28(2):159–171, 2014.
17. Ying Liu, Samuel Muller, and Ke Xu. A static compliance-checking framework for business process models. *IBM Systems Journal*, 46(2):335–361, 2007.
18. David Luckham. The power of events: An introduction to complex event processing in distributed enterprise systems. In *International Workshop on Rules and Rule Markup Languages for the Semantic Web*, pages 3–3. Springer, 2008.
19. Lauren B Movius and Nathalie Krup. US and EU privacy policy: comparison of regulatory approaches. *International Journal of Communication*, 3:19, 2009.
20. Emmanuel Mulo, Uwe Zdun, and Schahram Dustdar. Monitoring web service event trails for business compliance. In *Service-oriented computing and applications (SOCA), 2009 IEEE international conference on*, pages 1–8. IEEE, 2009.
21. Daniel E. O'Leary, S Bonorris, W Klosgen, Yew-Tuan Khaw, Hing-Yan Lee, and W Ziarko. Some privacy issues in knowledge discovery: the OECD personal privacy guidelines. *IEEE Expert*, 10(2):48–59, 1995.
22. Nick Papanikolaou, Sadie Creese, Michael Goldsmith, Marco Casassa Mont, and Siani Pearson. Encore: Towards a holistic approach to privacy. In *Security and Cryptography (SECRYPT), Proceedings of the 2010 International Conference on*, pages 1–6. IEEE, 2010.
23. Bill Roscoe. The theory and practice of concurrency. 1998.
24. Sarbanes-Oxley Act. Sarbanes-oxley act of 2002. *Public Law*, (107-204), 2002.
25. Valeria Soto-Mendoza, Patricia Serrano-Alvarado, Emmanuel Desmontils, and Jose Antonio Garcia-Macias. Policies composition based on data usage context. In *Sixth International Workshop on Consuming Linked Data (COLD2015) at ISWC*, 2015.
26. Aurobindo Sundaram. An introduction to intrusion detection. *Crossroads*, 2(4):3–7, 1996.
27. Huy Tran, Ta'id Holmes, Ernst Oberortner, Emmanuel Mulo, Agnieszka Betkowska Cavalcante, Jacek Serafinski, Marek Tluczek, Aliaksandr Birukou, Florian Daniel, Patricia Silveira, et al. An end-to-end framework for business compliance in process-driven soas. In *Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2010 12th International Symposium on*, pages 407–414. IEEE, 2010.