

Legally Sustainable Solutions for Privacy Issues in Collaborative Fraud Detection

Ulrich Flegel, Florian Kerschbaum, Philip Miseldine, Ganna Monakova, Richard Wacker, and Frank Leymann

1 Introduction

One company by itself cannot detect all instances of fraud or insider attacks. An example is the simple case of buyer fraud: a fraudulent buyer colludes with a supplier creating fake orders for supplies that are never delivered. They circumvent internal controls in place to prevent this kind of fraud, such as a goods receipt, *e.g.*, by ordering services instead of goods. Based on the evidence collected at one company, it is often extremely difficult to detect such fraud, but if companies collaborate and correlate their evidence, they could detect that the ordered services have never actually been provided.

There are many other cases with higher economical impact which cannot be detected by one party alone, *e.g.*, money laundering and insurance fraud. An aris-

Ulrich Flegel

SAP Research Center Karlsruhe, Vincenz-Prießnitz-Str. 1, 76131 Karlsruhe, Germany, e-mail: ulrich.flegel@sap.com

Florian Kerschbaum

SAP Research Center Karlsruhe, Vincenz-Prießnitz-Str. 1, 76131 Karlsruhe, Germany, e-mail: florian.kerschbaum@sap.com

Philip Miseldine

SAP Research Center Karlsruhe, Vincenz-Prießnitz-Str. 1, 76131 Karlsruhe, Germany, e-mail: philip.miseldine@sap.com

Ganna Monakova

Institute of Architecture of Application Systems, University of Stuttgart, Universitätsstraße 38, 70569 Stuttgart, Germany, e-mail: ganna.monakova@iaas.uni-stuttgart.de

Richard Wacker

Karlsruhe Institute of Technology (KIT), Institut für Informations- und Wirtschaftsrecht (IIWR), Haid-und-Neu-Strasse 6, 76131 Karlsruhe, Germany, e-mail: richard.wacker@kit.edu

Frank Leymann

Institute of Architecture of Application Systems, University of Stuttgart, Universitätsstraße 38, 70569 Stuttgart, Germany, e-mail: frank.leymann@iaas.uni-stuttgart.de

ing challenge is therefore collaborative fraud detection where organizations employ fraud detection algorithms on their joint data. It seems that the prerequisite for collaborative fraud detection is data sharing, *e.g.*, as proposed in Section 2. Data sharing brings along with it a number of new security and privacy challenges. Even in a risk-neutral setting no party is inclined to share its data, if the benefit does not exceed the risks plus the costs involved. Precisely assessing the risks of data sharing is very difficult, in particular for fine-grained data, such as event logs, since it is unclear what can be inferred from the data. Furthermore decision makers are often risk averse, such that the perceived risks can be very large, and therefore many companies share data very reluctantly. If the data to be shared is related to people, privacy legislation often prohibits or at least regulates and limits the possibilities for data sharing. We examine the legal ramifications of detecting fraud in detail in Section 6. As a consequence collaborative fraud detection should be performed in a privacy-respecting way, minimizing the data sharing risks.

In Section 2 we give an introduction into monitoring modern distributed business systems built according to the *Service Oriented Architecture* (SOA) paradigm. Section 3 shows how evidence produced by monitoring such systems can be correlated by use of heuristic rules and proposes a suitable architecture for that purpose. In Section 4 we switch from the technical view to a legal perspective. We motivate analyzing the use of monitoring data according to six basic privacy rules in Section 5.1, forming a common denominator of pertinent privacy acts in use. These legally motivated rules are applied in Section 7 to technical systems for fraud detection, as described in Section 2 and 3. We comprehend the results from this analysis as the benchmark for improvement for technical privacy mechanisms for fraud detection. To be able to propose practical privacy mechanisms we investigate the main multilateral requirements for detecting fraud and derive technical requirements for privacy solutions for fraud detection in Section 7.1. In Section 7.2 and 7.3 we present appropriate mechanisms for pseudonymizing evidence for fraud detection. We switch again to the legal perspective in Section 8 to determine the improvement that the additional effort for the proposed privacy mechanisms can afford us, before we conclude in Section 9.

2 Monitoring Modern Distributed Systems

Service Oriented Architectures are a way for a software system to expose its functionality via components, called services. Each service encapsulates a set of functions, or actions that it can perform, which form its functionality. Standards have been developed to describe services from an operational perspective using WSDL [30] and abstract BPEL [22], from a semantic perspective describing the actions it performs using OWL-S [29] or WSMO [32], as well as from a non-functional perspective using WS-Policy [31]. To ensure correct operational behaviours were conducted, observations of a system are needed. This section analyses observational characteristics of a service, shows how these characteristics can be modelled by way

of the evidence that can attest to service execution, and discusses how the provided evidence enables detection of fraudulent service behaviour.

To specify the unified content the observational characteristics of every service should contain, different service types in a SOA are analysed by way of the actions they can perform, and their interactions. SOA systems can be layered based on the type of services provided as shown in Figure 1.

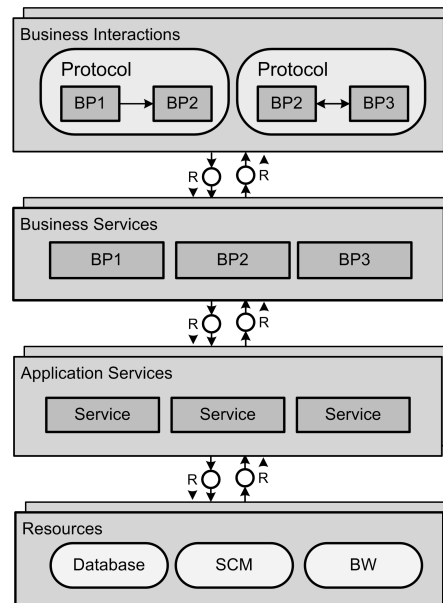


Fig. 1 Implementation levels of a SOA

The bottom layer contains system resources, which are accessed via *application services* shown on the next layer. Application services can be viewed as resource wrappers (or resource abstractions/virtualisations), which provide an interface enabling actions to be performed by the underlying resources.

Application services can be orchestrated to provide composite actions. Orchestrations define a flow of service invocations, commonly modelled as business process specifications. In classical business process design, as exemplified by the BPEL or BPMN standard [22, 25], services provide the functional implementation of business activities within a process, that lead to the achievement of a business goal. Business processes can be themselves exposed as services, encapsulating the composite action as a single action that can then be re-used within other orchestrations. As services perform their actions upon other services to achieve a business goal, they can be considered as *business services*. On this layer, the orchestrated services represent the application services upon which they perform, however do not directly expose these services. Instead, a business service can provide an action of type *in-*

voke that invokes the process represented, and thus perform the actions as defined in the process that are performed by the application services.

At all layers, services are invoked via the interface they expose. The service requester sends an invocation message as specified in a WSDL file. An Enterprise Service Bus (ESB) is a middleware component that acts essentially as a sink for these messages. ESBs can manipulate, redirect, and transform messages sent to and from services. This enables location transparency, which contributes to the flexibility of the SOA systems. Services can be dynamically discovered and chosen as long as the selection criteria, which include desired functional and non-functional properties, are satisfied. Viewing an ESB as a single logical component responsible for the message delivery, the complete service communication, also called service choreography [6], is visible in the ESB. Because an ESB itself can be exposed as a service, we specify the third type of services called *interaction* services, which are contained within the top layer in Figure 1. The interaction services perform, but are not limited to, actions *deliver* and *transform* on the messages which represent communication between business services.

Observations of service behaviours can be derived from multiple points within the SOA implementation, depending on the layer being analysed. For example, a service invocation that occurred within a business process can be observed via monitoring the business service encapsulating the business process (observing evidence signifying that the business process has invoked the service), via monitoring the ESB that captures the message sent to invoke the service (observing evidence signifying that the invocation message was delivered, accepted or deleted), and via monitoring the invoked service (observing evidence signifying the execution of the invoked operation).

As such, SOA provides rich sources of evidence attesting to system behaviour. This makes SOA systems particularly of interest to companies wishing to simplify compliance management, where observations of a system are needed to ensure correct operational behaviours were conducted. SOA systems however are not typically described by way of the evidence that can attest whether particular functionality was performed by a service. As stated previously, services are described by their functional and non-functional characteristics. The non-functional properties can describe security aspects of a service [21] or reliable messaging [23]. The observational characteristics however have been neglected. In the next section, we define an evidence model that describes such observational characteristics.

2.1 Evidence Model

The observational characteristics of a service describe its ability to provide evidence attesting its behaviour. The provided evidence, together with the evidence received from the other services in a system, describe the observational characteristics of a service.

A service behaviour is specified by actions a service performs, the order in which these actions occur, and conditions which enable or disable execution of certain actions. At runtime a service produces an instance of the specified behaviour. In a complex system consisting of multiple services, the overall behaviour of the system is based on the current behaviours of the participating services. Because the single services in a SOA are unaware of the other existing services (the loosely coupled principle of SOA), from the point of view of a single service the behaviours of the other services and effects produced by these behaviours form the operational context of the system. Because the execution of an action in a service can depend on the current system context, which in its turn depends on the behaviours of the other services, monitoring of the system as a whole is required to detect fraudulent behaviour of a service.

The current state of the system is defined by the current state of every single service in a system. The current state of a service is defined by the current states of the actions the service performs. To enable the monitoring of a complex system, the states of the participating services must therefore be observed. For this purpose a service must provide evidence on action state changes, which implies the state change of the service and the whole system. Because an action state change can cause a state change of resources the action operates on, the evidence must include information about resources the action operates on. To enable correlation and aggregation of the evidence from the different services in a heterogeneous environment, a common model of the evidence is required.

Existing service description standards do not provide a model rich enough to capture all information specified above. WSDL describes services in terms of operations they perform and messages they accept and produce. The order, in which the specified operations must be invoked, is not specified in WSDL. For the services which require stateful interaction and multiple messages, the order in which these messages need to be sent, called interaction protocol, must be specified in addition to WSDL. The interaction protocol essentially describes the ordering of operations in a service and WSDL specifies which messages are required for each operation. The ordering can be specified for example using abstract BPEL. The specification of the operations a service performs together with their ordering can be considered as description of the external behaviour of a service.

To be able to detect a fraudulent behaviour of a service however, the description of external behaviour alone is not sufficient. In addition, the internal behaviour of the service and provided evidence attesting to both external and internal behaviour must be specified. Currently there are no standard ways to describe the required information. One way to do this is to extend the existing descriptions of the external service behaviour with the specification of the internal actions performed on the invocation of a service operation. To enable the common understanding of the actions specified, they can refer to the shared vocabulary captured in an action ontology, which can be standardised for every specific domain. The evidence of the internal behaviour of the service can be provided in the form of events on the state changes of the actions, as it indicates the state change of the service, plus information the events can contain, including the description of the current state of the resources.

In general, a system in a SOA can be described as follows. Let Σ denote a SOA system, then $\Sigma = \mathcal{S} \times \mathcal{R} \times \mathcal{I}$, where $\mathcal{R} = \{R_1, \dots, R_l\}$ denotes the set of all resources, $\mathcal{S} = \{S_1, \dots, S_n\}$ denotes the set of all services, and $\mathcal{I} = \{I_1, \dots, I_m\}$ denotes the set of all interactions between services (or service choreographies).

Every service S_i performs a set of actions $\{A_{i_1}, \dots, A_{i_k}\}$. At any point of time every action is located in a specific state denoting the action progress. Let Γ denote the set of all action states. The state function $\sigma_{\mathcal{A}} : \mathcal{A} \times \mathcal{T} \rightarrow \Gamma$ returns the state of an action at specific time point, where \mathcal{T} denotes time. The state of the service S_i at time t is defined by the states of its actions: $\sigma_{\mathcal{S}} : \mathcal{S} \times \mathcal{T} \rightarrow 2^{\Gamma}$, or $\sigma_{\mathcal{S}}(S_i, t) = (\sigma_{\mathcal{A}}(A_{i_1}, t), \dots, \sigma_{\mathcal{A}}(A_{i_k}, t))$

The state of the system is defined by the current state of the system services, resources and interactions. As the previous section shows, every resource can be abstracted through an application service, which captures all accesses to this resource. This means that any state change of this resource implies invocation of an action of the application service. Therefore any resource state change can be related to the state change of the corresponding action of the application service. Thus, the resource states of the system can be derived from monitoring application service action state changes. For example if an action *update* on resource *data* changes its state to *completed*, then it can be derived that resource *data* is in state *updated*.

All service interactions happen through the service bus. Thus, monitoring of the state changes of the service bus actions provides enough information to derive the current state of service interactions. A similar approach for monitoring predefined choreographies in the service bus was described in [17].

Thus, the complete state of a SOA system can be derived from the state changes of the actions on the application service level, business service level and the interaction service level. Therefore the evidence model presented in this section considers the evidence that can show state changes of the service actions and provide information about the current state of the resources the action operates on, which allows monitoring for the resource state changes.

Fig. 2 shows a graphical representation of the evidence model based on the action state changes.

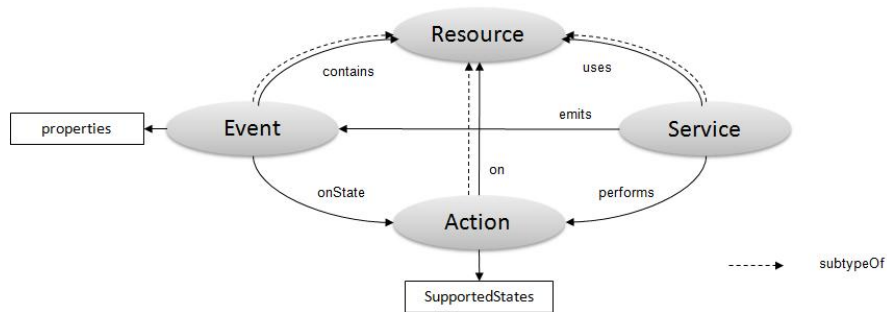


Fig. 2 Evidence model

The evidence model contains the following concepts:

- *Service* - represents the described service.
- *Action* - captures the actions service performs. A set of all actions builds action taxonomy. The sub- and super-class relationships between actions can be used to enhance modelling of the observational requirements. For example, if an evidence on execution of an action of type *AccessData* is required, and it is known that the actions *ReadData*, *UpdateData*, *DeleteData* are sub-classes of the action *AccessData*, then the evidence requirement can be propagated to all sub-class actions. Every action can specify a set of supported states as attributes. Every action can have its own set of supported states, we assume however that a superset of all possible states exists. This means that an action specific state set must always be a subset of the superset. An example of such a superset can be based on the BPEL activity state diagram [15] and consists of states *Started*, *Running*, *Faulted*, *Repaired*, *Suspended*, *Terminated*, *Completed* and *Compensated*
- The *Event* - describes events a service can emit which are related to a certain action state change. The relation *onState* between *Event* and *Action* concepts is an abstract relation. It can be refined with the *onStarted*, *onRunning*, *onFaulted*, *onRepaired*, *onSuspended*, *onTerminated*, *onCompleted* and *onCompensated* relations, depending on the states the corresponding action supports. Events can have properties, for example event timestamp. As a payload, events can contain information about resources the action operates on. This provides contextual information at the current execution state.
- *Resource* - Describes the resources which can be used by a service and on which the actions are performed. In an abstract way, everything can be considered as a resource: an action can be executed on a service, action or an event. Therefore a resource can be viewed as a super concept. Resources can have relations to other resources, which are captured in an ontology. A specific type of the resource ontology is an action taxonomy described above.

3 Observing Fraudulent Service Behaviours

A major advantage of SOA is its ability to connect multiple applications together to exchange and reuse functionality present in each. As an example, in the SAP Business Suite 7.0, many business processes are defined to provide Enterprise Resource Planning (ERP) behaviours over multiple SAP components. One such set of behaviours is defined for Accounts Receivable (AR) processes that record and manage accounting data of all customers for a business. This data forms the knowledge the business has about its customers, and is thus known as Customer Master Data (CMD). It is stored in a singular location, and is exposed via a set of services in the Business Suite.

Manipulation of Customer Master Data can yield multiple frauds. The SAP MIC (Management of Internal Controls) component for the Business Suite that controls AR processes defines a set of controls to prevent fraud. We consider how an example

by the control, not the applications, and accordingly it cannot be assumed that SAP CRM is aware what is or what is not sensitive in terms of CMD.

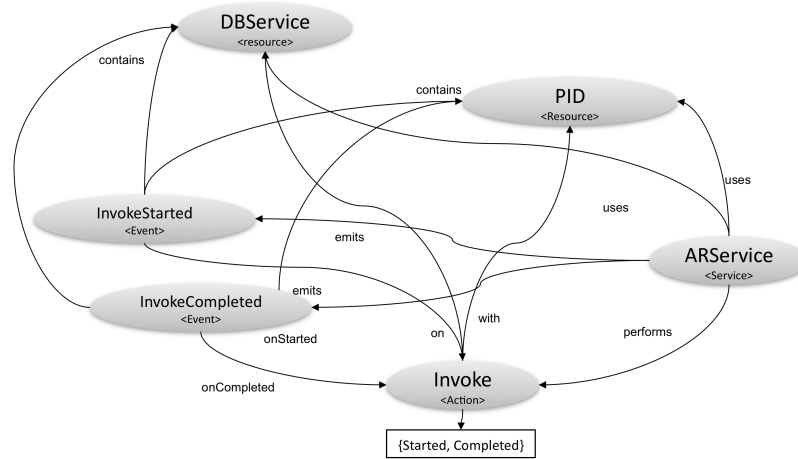


Fig. 4 AR Service evidence model instantiation

We assume events are produced by the DBService on the states start and complete on update action, with the state of the CMD record included into the event payload. For the control, we need to ensure that the update action for the CRM system does not add or change sensitive fields in the CMD record. We assume the sensitive field to be simply *bankNb*.

Using the evidence model, we define an abstract view of both components with regards to their use of CMD. To be able to monitor the specified control, we need to capture the events on start and complete of the update action in the DBService, including information of the sensitive fields at the current state and the ID of the service which triggered this update. Furthermore we need an event from the AR-Service indicating invocation of the DBService. The values of the sensitive fields in the events indicating the start and end of the update action must be equal, unless the update operation was invoked by the ARService. The graphical representation of the observational characteristics required to monitor this constraint is shown in Figure 3, where *PID* denotes the ID of the current service run and *RequesterPID* denotes the ID of the service run which invoked the DBService.

Assuming these events are provided, the monitoring rule can be specified as follows:

$$\begin{aligned} &\forall e_1 \in UpdateStarted, \forall e_2 \in UpdateCompleted : \\ &\quad e_1.CMD.BankNb \neq e_2.CMD.BankNb \rightarrow \\ &\quad (\exists s_1 \in ARService, \exists e_3 \in InvokeStarted : \\ &\quad \quad Emits(s_1, e_3) \wedge (e_3.PID = e_1.RequesterPID)) \end{aligned}$$

This example shows how correlation of the events from different services enables detection of a database update from a service different from the ARService. Note that the events can be adapted to operate in a synchronous manner, which would mean that the service which emits the event waits for permission to continue. In this case the security would improve, as some violations can be prevented by blocking the execution of a service action by observing a synchronous event on the start of an action, but service performance would suffer in this case.

3.1 Architectural Support

We now detail how such observation and evaluation could take place in practice. In Figure 5, a simplified architecture of a system is given that evaluates rules using evidence gathered from events described in the proposed model. The basic design of the system is described, with a description of the information flow.

The architectural schematic describes two entities: the Process Owner (in our examples, the *Hospital*), and a third-party in a Business Process Choreography (for example, an *Insurer*). As the process owner is unaware of the actual implementation of the services of the third-party, the architecture reflects that only a service model provided by the third-party is available for querying. We assume additional components are required for the service model to be managed, and exposed for query.

A business process execution component (here the *BPEL Engine*) instantiates a business process, and queries the *Rule Repository* to return predicates that govern the current execution state of the process. These predicates refer to the events described by the service models of each party in the choreography. Such predicates can be serialised in an XML Query language with Window support to support temporal conditions. Such work is detailed in [5].

As the *BPEL Engine* is aware of which services are being invoked through the process, the *Repository* can build a list of rules that reference the events that can be produced by these services. It determines this relationship based on a query to each party service model manager. The matching rules are relayed to a *Rule Evaluator* who instructs a *Monitoring* component to inform it when events needed to evaluate the rules are observed. The *Rule Evaluator* instructs each service model manager to inform it when the events needed occur.

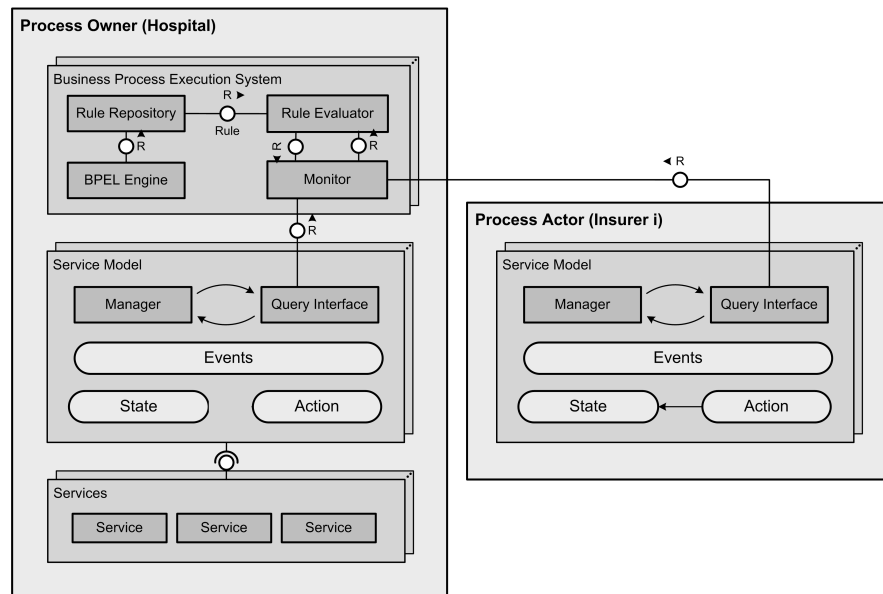


Fig. 5 Monitoring architecture

During the business process execution, services are invoked by the *BPEL engine* to provide the activities described in the process. Upon an invocation of an operation it provides, a service emits an event as described in its service model. This includes the properties that are represented in the event payload. The local service model manager informs the *Monitoring* component of this event if the *Monitoring* component indicated this event was of interest. This could be achieved in multiple ways: the use of a service bus for example, allows invocations and messages sent by and to services to be monitored. The *Monitor* forwards this to the *Rule Evaluator*. The event payload (detailed information regarding the event) is also sent so that the operational state leading to the event can be captured and represented in the rules too. The *Rule Evaluator*, upon receiving this event and the others it must observe to evaluate the rule, then evaluates the predicate.

4 Introduction to the Legal Perspective

Fraud often spans different organizations, and in the face of outsourcing, new fraud opportunities will be created. Detecting fraud requires a complete picture of the executed business processes, which can be implemented as described in Section 2. This necessitates collaboration of the involved organizations for detecting fraud. A main obstacle to collaborative fraud detection is data confidentiality or privacy where

parties are reluctant to share their possibly sensitive data. We analyse the statutory situation, and the effects of some fraud-detection specific privacy-enhancing technologies.

A fundamental problem in the discussion about international data privacy law is raised by the diverse viewpoints of data privacy as a result from different legal systems. A legal evaluation can hold only for a special legal system. On the other hand the objective, the protection of a citizen against the misuse or limitless propagation of his personal data is the same in any law system. Thereby aside from the dogmatic foundation the approach to constrain the spreading of personal data in a society leads to very similar basic ideas.

In consequence these basic fundamentals can be and have already been reduced by several institutions and jurisprudence to a set of basic principles, which can be seen as the least common denominator of data privacy law. Compliance to a set of rules is no guaranteed compliance to a special legal act, but it is surely an improvement towards a more privacy aware solution.

In many practical scenarios the processing of personal data is necessary and cannot be avoided completely. The processing of personal data serves the interests of the controller that in many cases are congruent to the interests of the person concerned himself – for example as indispensable element of an agreement. In consequence in most legal acts data privacy law allows handling of personal data in three cases. Firstly there is no need for a legal interference when the controller's interests impartially match the interests of the person concerned (see [8] Article 7, [12] §28). Secondly, if the controller's legitimate interests outweigh the interests of the person concerned. These cases arise, when one fundamental principle of democracy collides with another.¹ Any basic right of one person is limited by the basic rights of others. Thirdly the person concerned can legitimize the handling of his data by an act of his own free will for a specified purpose.

5 Basic Principles of Data Privacy Law

In this section the approach to form generally accepted privacy principles shall be discussed, several examples shall be presented and finally it shall be tried to gain a condensed set of rules that cover the main issues of all presented examples.

Firstly it should be mentioned, that there are two different intentions that drive the development of simple rules instead of very complex and detailed legal regulation. On the one hand, the practical side often demands a less complex and universal set of rules to achieve a portable solution for, *e.g.*, software compliance. On the other hand the issue of harmonization in the field of data privacy is much more critical than in other fields of law. Information flow easily overcomes national borders an isolated application of data privacy acts is nearly ineffective. Data that is collected in a country with high data privacy standards can easily be transmitted into such coun-

¹ For example the data privacy right would undermine the right of expression.

tries, that do not regulate at all or where the national regulation is less restrictive. This is underlined by the titles of the most important harmonization attempts which often contain the *transborder flow* of data (see in [8] *free movement of data* and *transborder flows* of the OECD). To overcome these problems privacy acts usually contain norms that prohibit the transmission of personal data into countries without sufficient data privacy legislation, *i.e.*, the safe-harbour principle that is part of the EU-Directive 45/95/EC (see [8] Article 4 (a)). In times of global companies these rules are hard to monitor and it would be much more efficient to guarantee an international minimum standard of data privacy like the EU-directive in the European Union. As a result of different cultures, constitutions and viewpoints this minimum standard can hardly be achieved in the shape of a detailed legal act. Like in most multilateral political attempts it is more likely to settle on a set of basic ideas.

5.1 A Set of Six Basic Rules

We reviewed three guidelines to determine their essence in a form of common rules that agree with international privacy law: The *OECD guidelines on the Protection of Privacy and Transborder Flows of Personal Data* [24] reduce data privacy to a set of eight principles and inspired most of the other initiatives, such as the basic principles for Hippocratic Databases (HDB) [1], which are also based on the US privacy act of 1974 [27]. Bizer's *Seven Golden Rules of Data Privacy* [4] are mainly derived from the German data privacy law, the German conversion of the EU directive 46/95/EC [8].

As assumed, even though these rule sets originate or are inspired of different legal approaches, their basic ideas are widely identical. As basis for the evaluation of FDS, these ideas shall now be aggregated under six labels and briefly defined. They shall cover all aspects of data privacy that have been identified by now.

5.1.1 Data Avoidance

“The personal information that is collected should be exactly the amount that is necessary!”

The whole issue of data privacy arises only, if personal data is collected, stored or processed. The basic idea of balancing the interests of controller and person concerned claims, that there must be a legitimate interest of the controller for any single element of personal data that he stores. In other words, if there is no necessity for an element of personal data, it should not be collected or stored. The amount of data that is collected should be this minimum that really serves the controllers interests.

5.1.2 Transparency

“The controller should ex ante and ex post inform the person concerned about anything that affects the person’s personal data.”

It is beyond question, that the person concerned can only assert its rights or give his acceptance, when the controller is forced to inform ex ante about the kind of personal data, the purpose under which it is collected or stored and the recipients, that will or may get the data afterwards. In addition a controller has to inform ex post, when a person is interested in the personal information that is stored and in the authorization he has given. All these sanctions can be subsumed to a principle of transparency. A person concerned can only monitor the controller’s actions, if these requirements are fulfilled. As a result of the directive of transparency it is a fundamental requirement that the controller has to store these facts in a way, that the person concerned is enabled to access them or at least get them from the controller, when he demands it.

5.1.3 Purpose Specification and Binding

“Any handling and the existence of personal data has to be strictly bound to a well documented purpose.”

After the collection² or storage³ of personal data this data moves out of sight from the person’s point of view. The *Directive of Transparency* can only be adhered to, if the picture the controller draws of the handling of personal data is guaranteed to reflect what really happens with them. Therefore the controller has to be forced to ex ante specify the purpose for which the data shall be used and to strictly bind any handling of personal data to the specified purpose. In any case he has to justify in which way his handling of the data serves this purpose and why this handling could not be avoided. In addition the purpose binding has a temporal aspect. The permission to handle or store personal data is strongly tied to the purpose. If the reason for the storage of the data no longer exists, the data must be erased, unless there is no other justification.

5.1.4 Prohibition Without Explicit Permission

“Any handling of personal data that is permitted by legal permission or consent of the person has to be understood as an explicit exception of a general prohibition.”

In the considered legal systems, permission for the handling of personal data can have its source either in the person concerned, who gives his consent to a specified handling of a specified quantum of personal data, or in law itself. The latter cases are

² Collection in the sense of data privacy law is acquiring data about a person concerned (see [12] §3 (3)).

³ Storage in this sense is a process that results in personal data, without a visible action—*i.e.*, the application of a system log or installation of a video camera.

special circumstances where the interests of the controller and the person concerned do not differ⁴ or where the legitimate interest of the controller in preventing harm to himself or his property impartially outweighs the interest of the person concerned.⁵ Independent from the source (consent of the person concerned or legal permission) this authorization is an exception of a general rule that prohibits any handling of personal data. If the given circumstances do not exactly match the requirements of such an exception, the handling of personal data is prohibited.

5.1.5 Data Quality

“Any personal data that is stored and used should be kept accurate and up to date.”

The controller has the obligation to keep stored personal data accurate and up to date. Data that is inaccurate or out of date must be corrected or deleted.

5.1.6 Data Security

“Personal data must be treated as sensitive data.”

All these rules till now deal with the behavior of the controller, which is in any legal case someone, who is known and can be monitored by the person concerned. In cases, where the controller intends to transmit the data to someone else, adherence to the principle of transparency demands that he informs the person concerned also about the recipients. Aside from an intended action that can result in a misuse of personal information by the controller himself or a third person, there is still the possibility, that the data is stolen by a third party. To prevent that the controller has to treat personal data he stores as any other kind of sensitive data. State-of-the-art methods of data security need to ensure that personal information cannot be spied out.

6 General Legal Requirements of Fraud Detection Systems

In this section the general privacy-relevance of state-of-the-art FDS shall be legally evaluated. A special focus lies on the questions, if there is a legitimate foundation for the adoption of an FDS and what the source of this permission can be.

⁴ *I.e.*, contractual relationships like ordering a product over the internet, where the data is needed for the delivery and similar cases.

⁵ *I.e.*, a scenario where a person concerned as user damages the controller's property.

6.1 Privacy Relevance of Fraud Detection Systems

As a starting point of this analysis the question about the general applicability of data privacy law to Fraud Detection has to be answered more formally. The applicability of data privacy law is tied to the concept of *personal data*. If the event data of an FDS can be qualified as *personal data*, the data privacy law is applicable to these systems. The EU-Privacy-Directive 95/46/EG in article 2 defines personal data as “... *any information relating to an identified or identifiable natural person (‘data subject’); an identifiable person is one who can be identified, directly or indirectly, in particular by reference to an identification number or to one or more factors specific to his physical, physiological, mental, economic, cultural or social identity.*”

A Fraud Detection System usually stores and processes information about the behavior of customers and insiders of an institution. While a customer may be a natural person, an employee definitely is a natural person. It is also beyond question that these persons are usually known to the controller – in general the institution that uses an FDS. The recorded data consists of elements that easily allow identification, so that any data record can be accounted to the person concerned.

The data that is collected by a fraud detection system is personal data by this definition and therefore underlies data privacy law. Thus the principles of data privacy law apply to FDS.

6.2 Necessary Data for Fraud Detection

A system that adheres to the principle of data avoidance needs to restrict the personal data that is collected to the minimum that is necessary to achieve the underlying aim. The data that is collected by an FDS can be limited in two dimensions. One dimension is the number of different data records. The other dimension is the number and the kind of data attributes that are stored. In both cases the limit of necessity cannot be fixed exactly. The only approach to this problem can be given by statistics. In the first dimension there might be a limit to an amount of data that is needed to detect harmful behavior with a specified probability. In the second dimension the limit is clearly exceeded, if data attributes are collected, that are obviously inappropriate to detect harmful behavior.

Summing up it can be stated, that a Fraud Detection System is no justification to observe arbitrary data that is accessible from the technical side. There must be a strong reference to the underlying purpose for any element of data that is stored. This could be achieved by identifying the actual data attributes evaluated by the FDS and only collecting these in cleartext.

6.3 Transparency in the Fraud Detection Context

The directive of transparency demands that the person concerned is previously informed about the kind and amount of data, which is stored by the controller. Even though the controller might consider it as inefficient to warn potential culprits, that they might be observed, the adoption of a Fraud Detection System should always be communicated to any person concerned. On the other hand detailed information of the methods that are used by the system would enable the other side to avoid detection. The extensiveness of details that have to be known to a person concerned therefore has to be found by balancing the interests⁶.

In summary the adherence to the directive of transparency demands, that the controller informs about an adopted FDS system, but not about all technical details.

6.4 Purpose Specification and Binding in Fraud Detection

The purpose of the adoption of an FDS is by definition the detection of deceitful behavior. As one of the most important points an adopted FDS has to comply with the strict purpose binding principle. The personal data that is collected as basis for the search of behavioral patterns can potentially be used for a large number of other purposes, like for example monitoring work performance or analyzing ones personal habits and social relationships. Thus the use of collected data has to be strictly bound to the specified purpose. As another consequence data has to be erased or at least blocked directly after the fraud analysis is completed without raising an alarm involving that data.

To sum up the adoption of an FDS requires that the use of the collected personal is restricted to the purpose of detecting fraud.

6.5 Permissibility of Fraud Detection

The legal definition of fraud varies depending on the pertinent legislation. In general fraud can be defined as an “intentional deception made for personal gain or to damage another individual” (cf. German Criminal Code [13] §278 (1)). In the fraud detection scenarios this criminal behavior usually targets the controller. A manifested suspicion of criminal behavior is clearly a case, where the interests of the controller outweigh the interests of the person concerned. In the considered legal systems this suspicion would justify a processing of the data without consent of the person concerned. A problem arises from the fact that a fraud detection system,

⁶ An FDS is thereby comparable with monitoring devices outside computer networks which possibly affect a person’s privacy like video cameras. In a working environment in many countries there are other legal acts that require transparency about the existence and application of such systems. Transparency at least requires that the monitored person knows about being watched and recorded.

comparable to a video camera, stores data without such a manifest suspicion against the person in focus. On the other hand the controller is permitted to protect himself and his property under the condition of proportionality. This demands that it efficiently serves the purpose and there is no less inculpatory way of protection. Under this premise adoption of a FDS will be permitted on the basis of a legal permission. The prohibition as default principle demands, that if it is questionable, that a handling of personal data is allowed without the authorization of the person concerned it should be avoided. In other words, only if it is without question that a certain data attribute, processing step or the collection of additional personal information really contributes to a better detection of fraud, it may be done without consent of the person concerned.

In summary the legal permission for the adoption of FDS is bound to the legal concept of proportionality. The FDS must guarantee an effective way of protection and among the possible ways the least inculpatory.

6.6 Quality of Event Data

The data quality aspect in the underlying scenario is especially important. The data is stored to detect and prove criminal behavior. The impact of an inaccurate data basis can thereby possibly be a false suspicion against a person concerned.

6.7 Security of Event Data

Like in any other scenario, where personal data is collected, stored or processed the data that is collected or processed by an FDS has to be treated as sensitive data. As a result data safeguards must be adopted. With respect to the subject of fraud detection to detect and prove criminal behavior, the event data is of a high degree of sensitivity.

7 Technical Solutions for Privacy-respecting Fraud Detection

One might ask whether additional technical effort geared towards privacy can improve the overall legal situation with fraud detection and ease its application in a lawful way. We therefore switch back to a technical perspective and present mechanisms that we will analyse for their legal effect in Section 8.

Collaborative fraud detection, just as many other collaborative algorithms, can be performed in two main architectures in distributed systems. The first architecture (examples are [18, 19]) employs the help of a semi-trusted, central third party: every party collaborating in the fraud detection algorithm prepares its data locally, hiding

as much information as possible, *e.g.*, by pseudonymization or encryption, and sends it to the central party. The detection algorithm based on the input of all parties is then performed at the central party. The outcome, *i.e.*, the detected fraud cases, are returned to the parties. Different forms of result sharing can be imagined, *e.g.*, the third party learns the result or does not. Ideally the central party remains oblivious to the input data, *i.e.*, it is not trusted to treat data confidentially, but is trusted to perform the detection honestly. This central party may offer fraud detection as a service in the corresponding business model. In the second architecture (examples are [2, 33]) there is no third party and the parties directly interact. Powerful security techniques, such as Secure Multi-Party Computation [3, 14, 35], exist in order to provably protect the data of each party in this architecture.

These two architectures differ in a number of crucial aspects. By many the first architecture is considered to be more practical, since it requires significantly less communication and the necessary computations are simpler by an order of a magnitude. In recent years several implementations of collaborative detection algorithms have arisen [26], all following the first architecture. The second architecture for which algorithms with provable security exist, remain theoretical research, probably due to the expected high communication and computation effort. It is important to compare absolute numbers in this respect and not merely complexities in the “big-O” notation, since constants matter a lot.

We will provide approaches to solve central problems in collaborative fraud detection in the first architecture. For the second approach we analyse the security it provides. It can therefore serve as an example of the kind of trade-offs that need to be made in order to realize practical, privacy-preserving collaborative fraud detection. For the first approach we determine the improvement from a legal perspective in Section 8.

7.1 Technical Requirements

Considering overall legal and technical requirements, we identify the following four potentially conflicting goals:

1. *detection effectiveness* of cooperation alliances,
2. *privacy* of honest individuals,
3. further organizational *confidentiality* requirements of process actors and process owners, and
4. *efficiency*.

The solution for the general problem requires domain knowledge about the cooperation method to account for efficiency and cooperation effectiveness. For considering privacy and confidentiality, organization-specific knowledge is required. Particularly, in practice we need solutions with realistic and implementable trust requirements, because cooperation partners may be within different organizations.

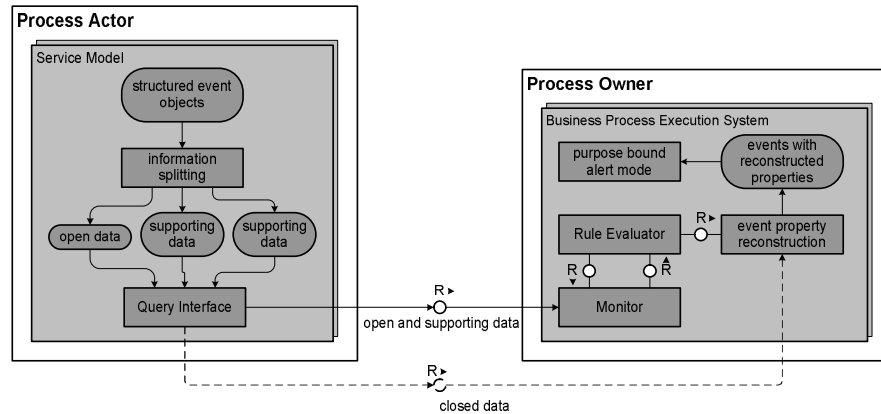


Fig. 6 Functional model for information reductions

Our solution extends the functional model from Section 3.1 (see Figure 5). A process actor service model generates *events* or just *information*, being consumed by its query interface (cf. Figure 5). The *process actor* represents the events or information using *structured event objects*. Appropriate information reductions process the structured event objects to satisfy detection effectiveness, privacy, confidentiality and efficiency. We distinguish lossless reductions and lossy reductions, depending on whether the original information from the structured event objects can be reconstructed or not, respectively:

Lossy reductions: remove information from structured event objects before forwarding it as *open data* to the process owner. The removed information must not be needed for further remote processing, and it should be definitely kept secret from the process owner and the other process actors, even under *inferences*. When information is coarsened, the detection effectiveness of the *Rule Evaluator* should not be affected unreasonably (see Figure 6, cf. Figure 5).

Lossless reductions: work by *splitting the information* contained in structured event objects into *open data* and *covered (masked, blinded) data* before forwarding it to the process owner (see Figure 6).

The open data of a lossless reduction is sufficient for the normal rule evaluation of the business process owner, possibly in conjunction with some *supporting data* that must be additionally generated depending on the evaluated rules. If a specific detection purpose is met during rule evaluation, a *purpose alert* is triggered. The respective open data together with the covered data allows for the reconstruction of the original information, subject to the detection purpose, *e.g.*, sufficient suspicion. The *data with the reconstructed information* can be used in the *alert mode*, *e.g.*, to hold fraudsters accountable.

7.1.1 Requirements for Open Data

The following requirements are crucial for sustaining the functionality of the relevant operations of the *Rule Evaluator* (see Figure 6):

- R1: certain event properties (except for timestamps) need to be compared to certain event properties for equal content, or equal prefix content
- R2: certain event properties (except for timestamps) need to be compared to values outside of the open data, *e.g.*, constant values, entries of a database
- R3: distances of event timestamp properties need to be computed and compared to values outside of the open data, *i.e.*, a constant value
- R4: the order of event timestamp properties needs to be determined

As a result, in order to sustain the effectiveness of the *Rule Evaluator* of the process owner, lossy reductions must be designed, such that they do not remove (timestamp) properties that are evaluated by the aforementioned operations.

Lossless reductions must be designed, such that the above operations can still be computed on the described event timestamps and event properties, and such that the results of the operations are still meaningful, *i.e.*, for an operation $\circ \in \{=_p, <, >\}$, where $=_p$ compares the properties up to a suitably determined prefix p , including $=_\infty$ for the full length, and two operands op_1 and op_2 in the open data and for a lossless reduction $r(\cdot)$ holds $op_1 \circ op_2 = r(op_1) \circ r(op_2)$.

Note that sustaining the ability to compare event properties or operation results on event properties to values outside of the open data may require to provide *supporting data*, *i.e.*, the reduction $r_s(\cdot)$ uses some parameter s , such that $op_1 \circ op_2 = r_s(op_1) \circ r_s(op_2)$ holds, where $r_s(op_1)$ is computed by the process actor and $r_s(op_2)$ is computed by the process owner.

7.1.2 Specific Requirements for Pseudonyms in Open Data

In the following, we consider pseudonymization as a special case of lossless reductions, and we focus on the specific requirements for pseudonyms in the open data, such that the *Rule Evaluator* of the process owner(s) sustains its detection effectiveness. Hence, the lossless reduction $r_s(\cdot)$ replaces some property f with an appropriate pseudonym $r_s(f)$, where s is a parameter that can be used to generate distinct pseudonyms for f . Note that $r_s(f)$ needs to preserve the comparability of property prefixes, if required by R1, *e.g.*, [34]. Also note that pseudonyms traditionally are used to hide personal data, such as identifiers of users, but in a general scope we consider pseudonyms as place-holders for arbitrary properties. A pseudonym $r_s(f)$ is appropriate, if

- $r_s(f)$ respects the syntax constraints of the *Rule Evaluator* wrt. f
- $f =_p f' \Rightarrow r_s(f) =_p r_s(f')$ holds if R1 requires that f must be testable for equal content or prefix to an event property f' ; note that both $r_s(f)$ and $r_s(f')$ are computed by the process actor

- $f \neq_p f', s \neq s' \Rightarrow r_s(f) \neq_p r_s(f'), r_s(f) \neq_p r_{s'}(f')$ holds generally, *i.e.*, $r_s()$ is collision-resistant, such that no unrelated events are correlated by accident
- $f = c \Rightarrow r_s(f) = r_s(c)$ holds if R2 requires $r_s(f)$ to be testable for equal content of a clear-text value c ; note that $r_s(f)$ is computed by the process actor, who also provides s in the supporting data, such that the process owner can compute $r_s(c)$
- $f \neq c \Rightarrow r_s(f) \neq r_s(c)$ holds generally, *i.e.*, $r_s()$ is collision-resistant (see above)

Note that R2 can be required independently from R1 wrt. to f , such that R2 may be required in addition to R1. The process actor only needs to provide s in the supporting data, if R2 holds. Also note that a database lookup for a given $r_s(f)$ requires the process owner to compute $r_s(c)$ for all c visited in the database, until a match is found.

In order to reduce the inferences an attacker can make on the transitive closure of a given pseudonym, it is desirable to use $r_s()$ in a way, such that additionally

- $f =_p f' \Rightarrow r_s(f) \neq_p r_{s'}(f'), s \neq s'$ holds if R1 does not require that f must be testable for equal content or prefix to an event property f' ; note that the process owner then does not need to and therefore is incapacitated to decide whether $f =_p f'$ or $f \neq_p f'$

We assume that all process actors a priori know the fraud detection rules of the process owner, such that all agents know, when R1 and/or R2 are required. This assumption can be met by proper coordination of the configuration of all parties.

All process actors that pseudonymize a given f must choose s in a coordinated way, if R1 is required. Then, the process owners can correlate events originating from distinct process actors by means of $r_s(f)$. If there is no coordination wrt. s , the following error can occur: $r_s(f) \neq_p r_{s'}(f'), s \neq s'$, despite $f =_p f'$, resulting in failure to correlate related events.

We have proposed concrete lossless reductions based on pseudonymization, which are respecting the requirements R1 and R2 [9]. The aspects for the corresponding covered data are summarized in Section 7.1.3 and 7.2. We have also proposed lossy reductions for event timestamp properties, which are respecting requirements R3 and R4. This recent result is described in some more detail in Section 7.3.

7.1.3 Specific Requirements for Covered Data

In contrast to the open data the covered data is basically independent from the rules evaluated by the process owner(s). This results from the fact that the covered data is not used by the *Rule Evaluator*. The covered data is used for information reconstruction when a purpose alert is triggered by the *Rule Evaluator*. This obviously results in fewer specific constraints for the design of lossless reductions.

It is necessary that the process actors generate covered data in a coordinated way. Note the analogy to the situation wrt. to the parameter s in Section 7.1.2.

7.2 Lossless Information Reduction with Covered Data

The process owner normally employs a *Rule Evaluator* to merely evaluate events with pseudonymized properties with respect to fraud rules. Only if a (*threshold alert* occurs, *i.e.*, a fraud suspicion has been detected, the covered data can be used for *information reconstruction*, *i.e.*, the original event data can be reconstructed. In the *purpose bound alert mode* the process owner can employ the *reconstructed event data* to establish accountability for legal purposes, such as damage prevention and litigation. In our previous work on pseudonymization the pseudonyms that replace identifying properties in the event data are chosen randomly while respecting the requirements for linkability for fraud detection[10].

For the covered data, the approach leverages Shamir’s threshold scheme for cryptographic secret sharing [28]: The fraud suspicions, also called *red flags*, for fraud detection are modeled as thresholds of secret sharing schemes. The covered data contains the encrypted identifying event properties that are replaced by the pseudonyms in the open data, and it contains shares of the respective decryption keys. As a result, the disclosure of the encrypted identifying event properties is enforced cryptographically, such that decryption is possible if and only if the pseudonyms are involved in a sufficient suspicion of fraud (*technical purpose binding*), *i.e.*, the number of shares associated with the pseudonyms exceeds the threshold in the model of the fraud suspicion. Note that it may be necessary to provide the ability to recover the decryption keys independently of a priori defined models of fraud suspicion in order to investigate fraud scenarios that have not (yet) been modeled. In that case, the grounds for decryption must be scrutinized by one or more trusted parties (*organizational purpose binding*). Involving these parties can be enforced cryptographically, *e.g.*, using threshold cryptosystems [11, 7].

7.3 Lossy Information Reductions for Timestamps

An important step of collaborative fraud detection is to identify correlated events. Time is often a necessary indicator for correlation, *i.e.*, one event happened (shortly) before the other. When collected as proposed in Section 2, events usually have a time property, *i.e.*, a timestamp. In order to be able to compare two timestamps from different sources in a distributed system, these systems require synchronized clocks. The wide-scale adoption of the Network Time Protocol (NTP) [20] offers such synchronization for systems connected to the Internet.

Our example protocol pseudonymizes timestamps for use of a central party in the first architecture (see Section 7). The central party – let’s call her Trudy – should learn as little information as possible about the timestamps, but it should still be able to perform the functionality of timestamp correlation. Assume two events which have occurred at times t and t' , respectively. Trudy’s task is to compute

$$|t - t'| < \delta$$

where δ is a threshold not to be exceeded.

Let Alice and Bob be two process actors and Trudy be the process owner or a third actor conducting fraud detection. Trudy can compare their timestamps for them using the algorithm described in the following section. The security guarantees of the algorithm are

1. Trudy cannot learn the value of any timestamp, hence the information reduction is lossy.
2. Trudy can compute the distance between any two timestamps, if that distance is below or equal to the threshold δ .
3. Trudy cannot compute the distance between two timestamps, if that distance is above or equal to 2δ .

The notion of distance can be extended to the 2-dimensional case [16] which has applications in location-based services.

7.3.1 Architecture and Algorithm

Alice and Bob jointly choose a shared secret s which ensures that the cryptographic functions are sufficiently hard to guess for Trudy. Let $MAC(\cdot, s)$ denote a message authentication code with the key s . Alice and Bob also agree on the threshold value δ which is the maximum of distance comparisons. Note that δ limits the information Trudy can learn, but also limits the computations Trudy can perform. Furthermore, Alice and Bob jointly choose a random number r between $0 \leq r < \delta$.

For each of their timestamps t Alice and Bob perform the following steps:

1. Compute the lower grid point $l = \delta \cdot \lfloor \frac{t-r}{\delta} \rfloor + r$.
2. Compute the upper grid point $u = \delta \cdot \lceil \frac{t-r+1}{\delta} \rceil + r$.
3. Compute the distance m between t and l as $m = t - l$.
4. Compute the distance v between t and u as $v = t - u$.
5. Send the timestamp tuple $\mathbf{t} = \langle MAC(l, s), m, MAC(u, s), v \rangle$ to Trudy.

For simplicity we do not differentiate between l and u and their hashed counterparts and refer to both as grid points.

Trudy computes the distance $d = |t - t'|$ of two timestamps t and t' from the timestamp (tuples) $\mathbf{t} = \langle g_1, h_1, g_2, h_2 \rangle$ and $\mathbf{t}' = \langle g'_1, h'_1, g'_2, h'_2 \rangle$ as follows:

Case 1: $g_i \neq g'_j \forall i, j: d > \delta$

Case 2: $\exists g_c = g_i = g'_j: d = |h_i - h'_j|$

Imagine the timestamps on a ray from left to right. The grid points divide the ray into equal-sized sections. Alice and Bob compute for each timestamp the two grid points closest to the timestamp: l is the lower one and u is the upper one. In addition the distance from the grid points to the timestamp is sent to Trudy in plain-text, *i.e.*, the least significant bits are leaked, but the exact value of those bits is still blinded by r . Fig. 7 shows two timestamps t_1 and t_2 (as dots on the ray) with distance $d < \delta$ which have a common grid point g_c (grid points are shown as line markers on the

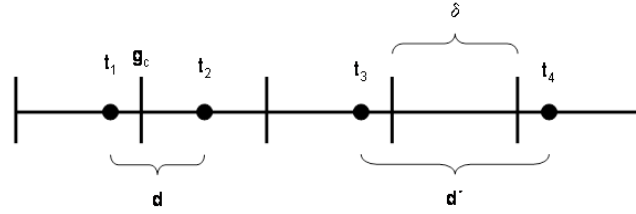


Fig. 7 Distances of 4 timestamps

ray) and two timestamps t_3 and t_4 with distance $d' > \delta$ which have no common grid point.

7.3.2 Limitations

A problem of our timestamp pseudonymization algorithm is that the privacy it provides is limited and only holds in the case of two timestamps. We describe an attack on larger sets of timestamps where multiple timestamps are close and form clusters that allow sorting all pseudonyms within a cluster.

Assume Trudy has a black-box device telling her for any two timestamp tuples their distance d , if $d \leq \delta$ or indicates otherwise. Such a device is a stricter form of our algorithm, which actually allows to compute the difference (and not just the distance) and also may allow the computation of some differences $d > \delta$ (but $d < 2\delta$). In summary, everything an attacker can do with such a black-box device, Trudy can do in our pseudonymization algorithm. Given this device and a data set T of tuples t_1, \dots, t_n , the attacker can sort the timestamps aligning them on a linear ray. He selects two tuples \mathbf{t} and \mathbf{t}' from T by repeatedly querying the black-box device, until $|t - t'| = d \leq \delta$. He then searches the remaining set of timestamps for an instance t'' (again by repeatedly querying the black-box device), such that $|t - t''| = d' \leq \delta$. Now, he queries the device for $d'' = |t' - t''|$. We assume that $t < t'$. The attacker will learn this from the difference, but the distance hides the direction, yet it is one additional bit of information. We could have achieved the same in our difference computation by flipping just one coin and accordingly multiply each difference with 1 or -1 before sending it to Trudy. If $d'' \leq \delta$ which he will learn from the device, then if $d = d' - d''$, he concludes that $t < t' < t''$ or, if $d' = d - d''$, then he concludes that $t < t'' < t'$. If $t' - t'' > \delta$, he concludes that $t'' < t < t'$ and that $d'' = d + d'$, i.e., he has computed the distance $d'' > \delta$ by inference from two other distances $d < \delta$ and $d' < \delta$. Given enough data points the attacker sorts all timestamps along the time ray.

The problem becomes harsher in our algorithm, because we use grid points. The attacker only needs to align the grid points on the ray and the timestamps will follow, but our analysis shows that the sorting is unavoidable by any solution to the problem that abstracts the functionality the way we do.

7.3.3 Evaluation

The larger the distances we can compute, the larger the utility for fraud detection, but also the less privacy our algorithm provides. This subsection aims at estimating the limit on the privacy provided by our algorithm in terms of the parameter δ and the arrival rate of new events.

We can model the arrival of events (and corresponding log entries with timestamps) by a constant arrival rate λ , *e.g.*, $5 \frac{1}{min}$. The distribution of the time difference between two consecutive events is then exponential with parameter λ and mean $\mu = \frac{1}{\lambda}$ (12s in the example). The distribution remains exponential in the case of multiple collaborating sources, since the distribution of a random variable $Y = \min(X_1, \dots, X_N)$ is also exponentially distributed, if all X_i are exponentially distributed.

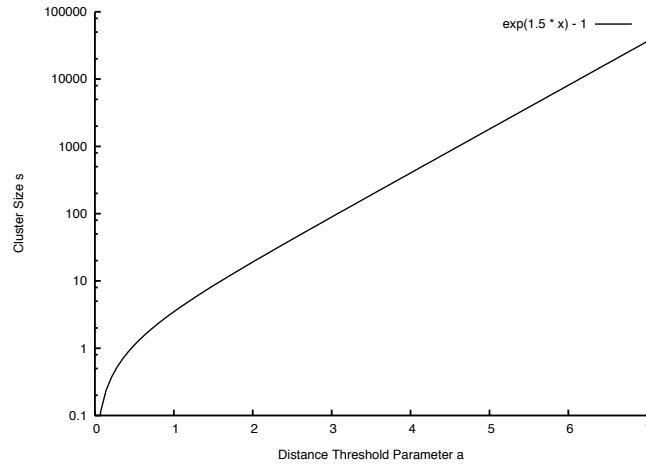


Fig. 8 Expected cluster size

We can express the threshold δ in terms of μ : $\delta = a\mu$ (*e.g.*, $\delta = 24s, a = 2$). Then using the cumulative density function of the exponential distribution we can compute the probability that an event occurs within time δ or less. We must model the interval $]\delta, 2\delta[$ where the probability that the distance $\delta + i$ ($0 < i < \delta$) between two timestamps can be computed is $1 - \frac{i}{\delta}$ (proof omitted), so we consider the average of distances below or equal to $\frac{3}{2}\delta = \frac{3}{2}a\mu$ as computable. The probability p that a distance d of an event to its predecessor is computable (*i.e.*, $d \leq \frac{3}{2}\delta$) is then:

$$p = 1 - e^{-\frac{3}{2}a}$$

We call a consecutive series of timestamps t_1, \dots, t_n where the distance between two consecutive timestamps t_i and t_{i+1} is less than or equal to $\frac{3}{2}\delta$ a cluster. Due to

the attack described in the previous subsection the distances between all timestamps in a cluster can be computed by inference. A cluster is broken every time an event occurs with a distance $d > \frac{3}{2}$ to its predecessor and therefore the expected size s of a cluster is

$$s = (1 - p) \sum_{i=0}^{\infty} i p^i = \frac{p}{1 - p} = e^{\frac{3}{2}a} - 1$$

In our example ($a = 2$) the expected cluster size is $s \approx 19$ timestamps. Fig. 8 displays the expected cluster size for the threshold parameter a .

8 Legal Improvements by Pseudonymizing Event Data

We now revisit our analysis from Section 6 and determine the improvement afforded by the technical solutions proposed in Section 7. Generally the statements made in Section 6 about the legal requirements for state-of-the-art FDS still hold for the FDS operating on pseudonymized data, but as we will see, some of these requirements are now technically supported instead of relying on an organizational approach. The advantages of fraud detection systems that work on pseudonymized event data shall now be evaluated.

Therefore the relevant features of the method described in Section 7.2 will be summarized again from a legal point of view. Afterwards the impact on the adherence of identified basic aspects of data privacy law will be discussed.

8.1 Technical Description

The pseudonymizing technique introduced in Section 7.2 has the effect that the identifying elements of the personal information stored by the FDS are replaced by a randomly chosen pseudonym. The mappings between pseudonym and the original identifiers are encrypted with a key using a secure cryptographic technique before persisting them. Without this key the identifying data elements can only be decrypted by an attack on the cryptographic method which usually takes a disproportional amount of time.

The key is fragmented by a method that guarantees that the original key can be only efficiently computed, if all components of the key are known. (1) The key is fragmented into two components which are distributed to the data privacy official and the FDS administrator and (2) the key is fragmented and bound to the steps of a known suspicious behavioral pattern. Thus, the key can only be computed efficiently in two ways:

1. The FDS administrator and the data privacy official collaborate by putting their key pairs together (*organizational purpose-binding*).

2. All steps which belong to a known highly suspicious behavioral pattern have been executed by one user (*technical purpose binding*).

The aim is to ensure, that the pseudonyms can only be exposed, if there is a strong suspicion or evidence that the person concerned committed fraud. In the following section it shall be evaluated in which way such a technique can improve the compliance of an FDS respecting to some of the predefined principles of data privacy.

8.2 Privacy Relevance of Pseudonymized Event Data

At first, like in the more general evaluation of FDS, the question has to be answered, if preliminary pseudonymized event data is still personal data from the legal point of view. Many national regulations, for example the German BDSG define pseudonymized data per se as non personal data (cf. [12] §3 (6)). Pseudonymizing is defined as “*replacing identifying data elements by a label to prevent or significantly hinder the disclosure of the person concerned identity.*” It could be concluded that if the pseudonymized data would not meet the definition of personal data, the data privacy law would no longer be applicable. A problem arises from the legal scope of the controller who draws the line at the border of the company. As long as the pseudonymized event data, the user’s identity and the mapping between both parts lie in the sphere of influence of one company the person concerned is still identifiable and therefore the data is still data privacy relevant. Thus the decomposed personal data is still legally qualified as personal data.

As just stated the reason why the pseudonymized data is still treated as personal data lies in the legal scope of the controller which per definition is the company. Based on this argumentation there is possibly an exception, if the technical and organizational architecture is created as follows:

1. Collection of event data and the fraud detection process are located in different organizations
2. The original event data is technically guaranteed to be never stored or dropped afterwards.
3. The key for the FDS administrator is automatically transferred to another company and guaranteed not to be saved locally.

In result the process actor that collects the event data – more precisely the data privacy official keeps just one part of the key for the organizational purpose binding. The other part lies in the hand of the process owner, or better, a third company that is conducting the fraud detection as a service, *i.e.*, executing the *Rule Evaluator* (cf. Figure 5). The pseudonymized event data is now from the perspective of both organizations non personal data per definition, because any attempt to identify the user requires a successful attack on the cryptographic method that has been used.

In this special case the parts of the event data that can be disclosed with proportional effort is reduced to the portion that indeed form a manifest suspicion against the person concerned.

8.3 Strengthening the Data Privacy Official

In a normal FDS systems purpose binding can only be implemented by organizational provisions alone. In other words it depends on the privacy awareness of a number of people who potentially have access to the event data, to ensure a privacy compliant operation. The data privacy official in many national legal acts is defined as the person, who shall control the adherence to data privacy in an organization. As such he has at least to absolve training in this field of law (cf. i.e. [12] §4f). Practically he often has only a watchman's position, and is rarely involved actively into data privacy relevant processes like fraud detection, what leads to a situation, that these activities are often out of his attention.

By adopting the pseudonymizing extension of a FDS this role changes from a passive to an active one. The pseudonymizer shall ensure technically that the information can only be disclosed under certain conditions and by certain positions inside an organization. How effective this technically supported restriction is achieved depends on the trustworthiness of especially one person – the data privacy official itself, who takes a very important role in this system. The mode of his involvement depends on the way of disclosure, technical or organizational purpose binding. The organizational purpose binding restricts the disclosure of pseudonyms efficiently to cases in which FDS administrator and the data privacy official collaborate. So the data privacy official directly scrutinizes suspicion that needs be reached to disclose a certain pseudonym. In the technical purpose binding version the suspicion that allows disclosure depends on the technical definition of fraud that underlies the behavioral patterns that allow automated disclosure of a pseudonym. If the definition of the patterns the exposure of pseudonyms is strictly bound to the occurrence of an undesirable behavior of the “person concerned” this technical solution also guarantees an increase of data privacy awareness. Thus it must be ensured by organizational or technical means that the FDS administrator also collaborates with the data privacy official for defining the behavioral patterns.

In summary the adherence to the purpose binding principle can be significantly improved by a pseudonymizing extension because the decision about disclosure of personal data is concentrated in the position of the data privacy official.

8.4 Disclosure With Legal Permission

In the case of FDS in general the collecting, storing or processing of a personal data will rarely be legitimated by the given consent of a person concerned. Thereby the

disclosure is strictly bound to cases, where this technique meets the condition that the personal right of the person concerned is outweighed by the controller's interest in protecting himself and his property. For these cases the principles contain an explicit permission based on the underlying national law. Thus to be legitimate the data usage must be restricted to such purposes, where this exception is applicable. By concentrating the decision whether a certain pseudonym shall be disclosed or not, in the position of the data protection official, who is or should be an expert in this field of law, this principle is technically supported. Ideally and as a further extension for the organizational purpose binding the system should require a brief documentation about the facts that lead to his decision. In any case, if the system is adopted in an appropriate way the amount of personal data that is effectively revealed will be smaller than in a state of the art FDS.

In conclusion a system integrated guarantee for purpose binding strengthens the control over the collected data and the legitimation of its usage.

8.5 Data and System Security

While in a FDS without this extension the event data itself must be protected against misuse the privacy-respecting version leads to a situation where the pseudonymized event data is not critical any more. Thus the improvement to the purpose binding principle that can be achieved by pseudonymizing techniques depends on appropriate safeguards that hinder physical access to the machine and the pseudonymizer software. If the technique is not bypassed and the mapping information is inaccessible by a third person the original personal data can no longer be used for other purposes than fraud detection. In comparison to a state-of-the-art FDS the adherence to data privacy principles demands, that personal data is held in a secure environment, but the safeguards that have to be implemented are mainly effective against outsiders. A protection against insiders of the controllers institution can only be guaranteed by organizational barriers. The pseudonymizing technique focuses on insiders. An outsider who tries to steal information usually tries to break into the account of an insider. Thereby, if this method minimizes the circle of internal users that have access to personal data, this effectively hinders outsiders too.

The presented pseudonymizing technique reduces the circle of persons inside the controller's institution that have access to the original event data. Hence, the technique also builds up an additional barrier for outsiders.

9 Conclusion

We have shown a general approach for monitoring modern SOA systems for fraud evidence and determined the legal ramifications. Fraud detection is subject to privacy law, implying obligations such as the limited use of collected evidence, infor-

mation of the affected individuals, and arguing the effectiveness and necessity of the planned FDS. For easing the introduction of FDS in practice we considered technical requirements for privacy mechanisms and proposed two concrete mechanisms for pseudonymizing event properties, one specialized for timestamps. We analysed the technical trade-offs of the latter and determined the improvement of the legal situation for the first.

The result is that the technical solution would significantly help the controller to guarantee, that a FDS adheres to the basic principles of data privacy. The strict purpose binding is technically guaranteed or at least supported. The level of data security has increased by implementing an additional barrier. From the legal point of view, especially from the perspective of the principle of proportionality, it can be concluded, that the availability of such a technique even constitutes an obligation to use it.

Acknowledgements

The research leading to these results has received funding from the German Federal Ministry of Economy and Technology under promotional reference 01MQ07012 (project THESEUS/TEXO) and the European Communitys Seventh Framework Programme (FP7/2007-2013) under grant agreement FP7-216917 (project MASTER). The authors take the responsibility for their contribution.

References

1. Agrawal, R., Kiernan, J., Srikant, R., Xu, Y.: Hippocratic databases. In: VLDB, pp. 143–154. Morgan Kaufmann (2002)
2. Atallah, M.J., Bykova, M., Li, J., Frikken, K.B., Topkara, M.: Private collaborative forecasting and benchmarking. In: V. Atluri, P.F. Syverson, S.D.C. di Vimercati (eds.) Proceedings of the ACM Workshop on Privacy in the Electronic Society, pp. 103–114. ACM (2004)
3. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: Proceedings of the 20th ACM Symposium on Theory of Computing, pp. 1–10. ACM (1988)
4. Bizer, J.: Sieben goldene Regeln des Datenschutzes. *Datenschutz und Datensicherheit* **31**(5), 350–356 (2007)
5. Botan, I., Kossmann, D., Fischer, P.M., Kraska, T., Florescu, D., Tamosevicius, R.: Extending XQuery with window functions. In: VLDB '07: Proceedings of the 33rd international conference on Very Large Data Bases, pp. 75–86. VLDB Endowment (2007)
6. Decker, G., Kopp, O., Barros, A.: An Introduction to Service Choreographies. *Information Technology* **50**(2), 122–127 (2008)
7. Desmedt, Y., Frankel, Y.: Threshold cryptosystems. In: G. Brassard (ed.) Proceedings of the Conference on Advances in Cryptology (CRYPTO'89), no. 435 in Lecture Notes in Computer Science, pp. 307–315. Springer, Santa Barbara, California (1989)
8. 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 move-

- ment of such data. Official Journal L 281 (1995). http://europa.eu.int/eur-lex/en/lif/dat/1995/en_395L0046.html
9. Flegel, U.: Pseudonymizing Unix log files. In: G. Davida, Y. Frankel, O. Rees (eds.) Proceedings of the Infrastructure Security Conference (InfraSec2002), no. 2437 in Lecture Notes in Computer Science, pp. 162–179. Springer, Bristol, United Kingdom (2002)
 10. Flegel, U.: Privacy-Respecting Intrusion Detection, *Advances in Information Security*, vol. 35. Springer, New York (2007)
 11. Gemmel, P.S.: An introduction to threshold cryptography. *Cryptobytes* 2(3), 7–12 (1997)
 12. Federal data protection act. In: German Federal Law Gezette, p. 2954 ff. (1990). <http://www.datenschutz-berlin.de/gesetze/bdsg/bdsgeng.htm>
 13. Criminal code. In: German Federal Law Gezette, p. 945 ff. (1998). <http://www.iuscomp.org/gla/statutes/StGB.htm>
 14. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: Proceedings of the 19th ACM Conference on Theory of Computing, pp. 218–229. ACM (1987)
 15. Karastoyanova, D., Khalaf, R., Schroth, R., Paluszek, M., Leymann, F.: BPEL Event Model. Technical Report Computer Science 2006/10, University of Stuttgart, Faculty of Computer Science, Electrical Engineering, and Information Technology, Germany, University of Stuttgart, Institute of Architecture of Application Systems (2006)
 16. Kerschbaum, F.: Distance-preserving pseudonymization for timestamps and spatial data. In: P. Ning, T. Yu (eds.) WPES, pp. 68–71. ACM (2007)
 17. Kopp, O., van Lessen, T., Nitzsche, J.: The Need for a Choreography-aware Service Bus. In: YR-SOC 2008, pp. 28–34. Online (2008)
 18. Lee, A.J., Tabriz, P., Borisov, N.: A privacy-preserving interdomain audit framework. In: Proceedings of the 5th ACM workshop on Privacy in electronic society, pp. 99–108. ACM, New York, NY, USA (2006). DOI <http://doi.acm.org/10.1145/1179601.1179620>
 19. Lincoln, P., Porras, P., Shmatikov, V.: Privacy-preserving sharing and correlation of security alerts. In: Proceedings of the 13th USENIX Security Symposium, pp. 239–254. San Diego, California, USA (2004)
 20. Mills, D.: Network time protocol (version 3) specification, implementation (1992)
 21. OASIS: Web Services Security Policy Language (WS-SecurityPolicy) (2005). URL <http://specs.xmlsoap.org/ws/2005/07/securitypolicy/ws-securitypolicy.pdf>
 22. OASIS: Web Services Business Process Execution Language Version 2.0 (2007)
 23. OASIS: Web Services Reliable Messaging Policy Assertion (WS-RM Policy) (2008). URL <http://docs.oasis-open.org/ws-rx/wsrmp/200702>
 24. OECD: Guidelines on the protection of privacy and transborder flows of personal data. http://www.oecd.org/document/18/0,3343,en_2649_34255_1815186_1_1_1_1,00.html, (2009-07-01) (1980)
 25. OMG: Business process modelling notation (BPMN) specification version 1.2 (2006)
 26. Parekh, J.J., Wang, K., Stolfo, S.J.: Privacy-preserving payload-based correlation for accurate malicious traffic detection. In: Proceedings of the 2006 SIGCOMM workshop on Large-scale attack defense, pp. 99–106. ACM, New York, NY, USA (2006). DOI <http://doi.acm.org/10.1145/1162666.1162667>
 27. United States House of Representatives 93d Congress, n.S.: US privacy act of 1974. <http://www.usdoj.gov/opcl/1974privacyact-overview.htm> (2009-07-01)
 28. Shamir, A.: How to share a secret. *Communications of the ACM* 22, 612–613 (1979)
 29. W3C: OWL-S: Semantic Markup for Web Services (2004). URL <http://www.w3.org/Submission/OWL-S/>
 30. W3C: Web Service Modeling Ontology (WSMO) (2005). URL <http://www.w3.org/TR/wsd120/>
 31. W3C: Web Services Policy 1.2 - Framework (WS-Policy) (2006). URL <http://www.w3.org/Submission/WS-Policy/>
 32. W3C: Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language (2007). URL <http://www.w3.org/Submission/WSMO/>

33. Waters, B.R., Balfanz, D., Durfee, G., Smetters, D.K.: Building an encrypted and searchable audit log. In: Proceedings of the 11th Annual Network and Distributed System Security Symposium (2004)
34. Xu, J., Fan, J., Ammar, M., Moon, S.B.: Prefix-preserving IP address anonymization: Measurement-based security evaluation and a new cryptography-based scheme. In: Proceedings of the 10th IEEE International Conference on Network Protocols (ICNP), pp. 280–289 (2002)
35. Yao, A.C.C.: Protocols for secure computations (extended abstract). In: Proceedings of the annual IEEE Symposium on Foundations of Computer Science, pp. 160–164. IEEE (1982)