Requirements for Secure Logging of Decentralized Cross-Organizational Workflow Executions

Andreas Wombacher¹, Roel Wieringa¹, Wim Jonker¹, Predrag Knežević², and Stanislav Pokraev³

¹ University of Twente, 7500 AE Enschede, The Netherlands
² Fraunhofer Gesellschaft, Integrated Publication and Information Systems Institute, 64293 Darmstadt, Germany
³ Telematica Institute, Enschede
knezovic@ipsi.fhg.de
Stanislav.Pokraev@elin.nl

Abstract. The control of actions performed by parties involved in a decentralized cross-organizational workflow is done by several independent workflow engines. Due to the lack of a centralized coordination control, an auditing is required which supports a reliable and secure detection of malicious actions performed by these parties. In this paper we identify several issues which have to be resolved for such a secure logging system. Further, security requirements for a decentralized data store are investigated and evaluated with regard to decentralized data stores.

1 Introduction

A multi-lateral collaboration representing a cross-organizational workflow, is based on communication between several parties each providing its own local workflow. However, several interaction structures of such workflows can be differentiated. The most different once are centralized and decentralized workflows. A centralized workflow consists of a single coordinator, who derives and checks the actions to be performed next. In a decentralized cross-organizational workflow such a centralized coordinator is missing and all parties have to trust the other parties to perform correct actions, that is, actions which are in accordance with the cross-organizational workflow. If an action is performed which (i) does not fit to the receiving party’s workflow state or (ii) an action derived from a party’s workflow state is not accepted by the receiving party, then an inconsistency of the cross-organizational workflow state is detected in case the cross-organizational workflow is considered to be consistent.

However, there are two main strategies to cope with inconsistent states in cross-organizational workflows: the first option is to prevent these states to happen and the second option is to provide means to detect these inconsistent states and to determine the fraudulent party (detection approach). Prevention has been addressed in different

¹ An action which is not performed results in later actions, which do not fit to the cross-organizational workflow and therefore must not be considered explicitly.
² Consistency here means the deadlock-freeness of the cross-organizational workflow.

approaches e.g. under the label of fair exchange of goods in multi-lateral collaborations. However, these approaches require a lot of shared knowledge of the cross-organizational workflow and its state to provide the required properties. Alternatively, non-repudiation protocols can be used to resolve conflicts in a bilateral exchange, although this does not provide a solution for the multi-lateral case due to a lack of a global ordering criteria of the bilateral exchanges. Further, all these approaches introduce a lot of communication overhead, which in most of the cases is not needed due to the partners act honestly. Therefore we follow the detection approach supporting a logging information to determine actors and malicious actions, which are unsupported by the cross-organizational workflow. This decision has to be derived based on the logged local state information of the parties involved and being relevant to this instance of the cross-organizational workflow execution. The overall aim is to provide a logging mechanism where local state information is used to reconstruct the cross-organizational workflow state and to determine the party performing a malicious action. As a consequence of this, integrity and non-repudiation of logging data has to be ensured. Further, since logging data provides a detailed insight into the mission critical information of a party data privacy has to be granted.

The aim of this paper is to identify the issues and security requirements of auditing cross-organizational workflows and discuss potential logging methods and their compliance with the identified security requirements. In particular, we illustrate that the cross-organizational workflow state construction problem is independent of the used storage method. Further, we elaborate that centralized logging is hard to realize in concrete applications and that Distributed Hash Tables with probabilistic data availability guarantees cover the specified security requirements.

The paper starts with a brief introduction of a scenario (Section 2) used for explaining the state representation and communication issues (Section 3). In Section 4 the security requirements for a decentralized data store are derived. In Section 5 the derived security requirements are compared with decentralized storage solutions. Finally the paper concludes with related work, a summary and future work.

2 Scenario

Auditing cross-organizational workflows is discussed by means of an order process example consisting of an accounting department authorizing the order after authenticating the buyer submitting the order and a logistics department providing the shipping including a parcel tracking.

The cross-organizational workflow representing this scenario as depicted in Figure 1 is represented in Finite State Automation (FSA) [5] notation [6]. States are represented as circles, transitions represent message exchanges denoted as arcs where sender, recipient and message name are annotated to an arc. The successful termination state of an FSA is indicated by a final state denoted by a circle with a thick line.

3 A classical result of distributed systems is that bilateral knowledge is insufficient for deriving global knowledge due to loss of information. Global knowledge can only be derived if additional information is provided. [4]

4 More complex models like Workflow Nets (WF-Net) [11] could also have been used.
The process starts with buyer and accounting department authenticating each other, while both orders of authentication are supported using B#A#auth and A#B#auth messages. After the authentication the buyer sends either the purchase order (B#A#PO message) or a change order request (B#A#change_order message) to the accounting. The buyer request is the basis for the accounting to send an order (A#B#order message) to the logistics department. In case the order is a new order, the logistics requests money from the accounting by sending an invoice (L#A#invoice message).

The cross-organizational workflow depicted in Figure 1 represents the combined public workflows of the three parties involved. In particular, the public workflow describes the externally observable message exchanges the party is involved in during the execution of the cross-organizational workflow. The public workflows of the three parties are depicted in Figure 2.

In this example only three parties are involved in an instance of the cross-organizational workflow. However, it is usual that the different parties are participating in several instances of a workflow and are involved in several cross-organizational workflows, thus, there is a high number of parties forming a network. A characteristic of this network is that it is changing over time, that is, new parties are joining, while others are leaving. To provide a logging of the cross-organizational workflow a decentralized representation of the cross-organizational workflow state and a decentralized data store are required.

3 State Representation

The aim of the state representation is to be able to derive the state of the cross-organizational workflow based on the locally logged state information. Since the cross-organizational
workflow is coordinated by the exchange of messages, the only state information which can be interpreted by all parties is the sequence of exchanged messages. The challenge is to reconstruct the message sequence of the cross-organizational workflow from the logged public workflow message sequence information.

3.1 Communication Model

The way to derive cross-organizational state information from public one dependents on the communication type: synchronous communication, where the recipient must pick up the message before the sender can continue the processing, or asynchronous communication, where the sender does not need to wait for reception of the message before continuing. Due to this definition, the order of sending messages and the one of receiving messages may quite vary in asynchronous communication model, while they are quite the same in the synchronous communication model. Since the synchronous communication model contains more constraints it is easier to handle and will be used for further discussion. In particular, message sequences supporting a synchronous communication model can be represented as Finite State Automata (FSA) [5].

3.2 Example

Let’s assume first there are no malicious actions contained in the public workflow’s message sequences. Then the construction of the cross-organizational workflow’s message sequences is based on combining the public workflow’s message sequences by keeping the local order of the public sequences and using the co-occurrence of sending and receiving messages to further reduce the potential number of combinations. Let’s consider the following execution sequences derived from the public workflows depicted in Figure 3 where a message is represented by its sender $S$, its recipient $R$, and the message name $msg$ as $S\#R\#msg$:

$$B: \quad A\#B\#auth - B\#A\#auth - B\#A\#PO$$
$$A: \quad A\#B\#auth - B\#A\#auth - B\#A\#PO - A\#L\#order$$
$$L: \quad A\#L\#order$$

Combining the message sequences supported by $B$ and $L$ results in the following potential sequences:

$$A\#L\#order - A\#B\#auth - B\#A\#auth - B\#A\#PO$$
$$A\#B\#auth - A\#L\#order - B\#A\#auth - B\#A\#PO$$
$$A\#B\#auth - B\#A\#auth - A\#L\#order - B\#A\#PO$$
$$A\#B\#auth - B\#A\#auth - B\#A\#PO - A\#L\#order$$

Considering also the sequence reported by $A$ and combining it with the above derived potential sequences constructed from the sequences supported by $B$ and $L$ results in the single sequence:

$$A\#B\#auth - B\#A\#auth - B\#A\#PO - A\#L\#order$$

which represents the single possible state of the cross-organizational workflow. The combination of the public message sequences not necessarily results in a single sequence,
but may result in a set of sequences in case the process is not completed and further actions are still possible. However, a complete execution of a cross-organizational workflow always results in a single combined execution sequence.

3.3 Concurrent Communication

There are rare cases where the order of sending and receiving messages do not correspond as illustrated on behalf of the following example. The buyer and the accounting department can concurrently exchange the authentication messages, thus, the messages may cross each other at the communication channel. The following execution sequences can be derived from the public workflows, which individually conform to the cross-organizational workflow.

\[
\begin{align*}
A : & \quad A \# B \# auth - B \# A \# auth \\
B : & \quad B \# A \# auth - A \# B \# auth
\end{align*}
\]

With regard to the cross-organizational workflow, these two sequences can not be combined to a single one as described in Section 3.2 because the order of the messages are contradicting each other and a combined order of the two partial orders can not be achieved. As a consequence, two potential message sequence have to be considered, that is, \( A \# B \# auth - B \# A \# auth \) and \( B \# A \# auth - A \# B \# auth \) where a precedence of the ordering of one party is given against the ordering maintained by the other party. To avoid the handling of these precedences the following issue has to be resolved:

**Issue 1.** (Exclusive Communication Channels) The used synchronous communication model must guarantee that the bilateral communication channel is used exclusive, that is, can only be used by a single party at a time.

In case this issue is not resolved, the resulting state representation problem is comparable to the one observed under the asynchronous communication model.

3.4 Malicious Actions

In the following we consider parties to log malicious actions under the prerequisite that the cross-organizational workflow is consistent, that is, does not contain deadlocks. The following cases involving malicious actions are observable for a single message exchange:

**A single party logs malicious actions,** while the other one logs truthfully. It can be detected that two different messages have been logged, although it can not differentiated which party cheated. It could either be the sender, who logged a message he hasn’t sent to the recipient, who is logging truthfully, or the sender logging truthfully while the recipient logs a malicious action. It get’s even worse if both parties cheat.

**Both parties log malicious actions.** In case the parties are logging different messages, again the difference can be detected although it can not be decided who or whether at least one acted truthfully. In case the two parties agree on which malicious action to log (the two parties conspire), then the malicious logging remains undetected in a first step. Since each party can log any message, the representation of a message exchange in
the cross-organizational workflow depends on the information of the sent and received messages. The Hamming distance \( d \) specifies the difference between two different codings or code words, where the distance provides information about the error detection (maximum \( d - 1 \) errors can be detected) and error correction (maximum \( \frac{d-1}{2} \) errors can be corrected) capabilities [6]. With regard to the reconstruction of the global state, two logging entries form up an entry for the global state, therefore the Hamming Distance is two supporting the detection of one error and no error correction.

We observe this behavior in the first case involving malicious actions, where only one party cheats. Here we can detected the error, but can not decide who caused the error to correct the cross-organizational state information. As a consequence, the limitation to one error detection and no error correction is quite strong and is not appropriate for the problem of auditing cross-organizational workflows. The research challenge here is how to increase the Hamming distance of the log data or how to use additional context information to support error correction and higher order error detection.

**Issue 2. (State Representation)** The data logged by the different parties has to be configurable with regard to the number of resolvable malicious actions in accordance to the actual security requirements of the application scenario.

### 3.5 Conflict Resolution

Before further digging into the technical issues of the logging and the corresponding security requirements for storing log data, we will discuss the approach of conflict resolution using the log data. The process starts with a party complaining about state inconsistency observed at its local state representation, that is, receiving a message which does not fit to the current state or a party refusing to accept a sent message. In either case, an authority is included, which is trusted by all parties like e.g. court. In front of the authority, each party presents his evidence on illustrating that he/she always acted truthfully. The authority checks the proof for integrity and combines the proof's to resolve the conflict, that is, identifies the party performing malicious actions. Finally, the authority specifies the compensation to be provided by the misbehaving parties to the remaining parties.

One major issue here is to provide reliable proofs derived from the logging information. A standard approach to maintain logging information is using a trusted third party, that is, a single party which is trusted by all parties involved in the cross-organizational workflow. An example of such an approach is e.g. [7] or [8], but there also the coordination of the cross-organizational workflow (at least the general one) is performed by a centralized instance as opposed to a decentralized coordination as discussed in this paper. Further, the centralized solution suffers from the fact that all involved parties have to agree on a single trusted third party. In particular, [9] states that decentralization engenders trust and centralized regulation destroys it, which favors a decentralized monitoring of state information. Therefore, we focus in the following on decentralized storage models for logging information and the discussion of the security requirements. However, the above discussed issues of exclusive communication channels (see Issue 1) and state representation (see Issue 2) are independent of the used storage model for logging data.
4 Security Properties of Data

Independent of the format and the details about the actual data being logged there are several security requirements which have to be considered using a decentralized auditing of cross-organizational workflows: integrity of public workflows and data, privacy of content, originator or storing party, and data availability.

4.1 Integrity of Public Workflow Models

The basis for comparing the log information with the actual cross-organizational workflow are the public workflows being the basis for the cross-organizational one. It is important that the public workflows are not altered after the consistency of the cross-organizational workflow has been confirmed. Thus, the following issue arises:

Issue 3. (Integrity of Public Workflows) The integrity of the public workflows used for consistency checking has to be guaranteed.

A potential approach to this is to log the public workflows using the same data store as for logging message exchanges. For example, each party logs its own public workflow and the ones of its trading partners using the same mechanisms as for logging data.

4.2 Integrity of Stored Data

The usage of a decentralized data store for logging data implies that log information is handled by several parties until it is finally stored. All parties involved in the storing or retrieval procedure may tamper the integrity of the data passed by. In particular, the data can be modified, data can not be forwarded in an appropriate way, that is, withdrawing data from the storage system, or false data can be introduced by replacing the original data, as e.g. done in replay attacks. The latter two address transactional properties of the store and retrieve operations of the used data store.

The idea is to have no need to secure the store and retrieval operations, because the originator and the storing party remain unknown/private. Therefore, no party in the middle performing a part of the operation knows whose data it is currently handling therefore does not provide any information to perform specific threats. Obviously, a party may block or change data passing by, but this is detectable by the originator of the operation. Such a party can quite easily be identified due to determinism of the used operations. Therefore the integrity requirements on data drill down to integrity of data content:

Issue 4. (Integrity of Data) The decentralized data store has to support the integrity of data content and transactional properties.

3 In the following we assume reliable communication channels, since unreliable communication channels introduces several orthogonal problems which are addressed e.g. by the distributed systems community (see e.g. [4]).
4.3 Privacy

Due to the usage of a decentralized data store resulting in storing content at different parties, which are unknown to the party performing the store operation, the privacy of the data content has to be protected. In particular, it has to be ensured that the content could neither be read by the party routing the content nor by the party storing the content. Further, privacy about the originator of the store operation as well as privacy about the actual storing party is required. The first one prevents that a storing party alters the content or doesn’t log the content to foist a malicious action on the originating party initiated by the storing party itself. The second one prevents the party performing the store operation to conspire with the storing party to finally alter data, add or remove data. Thus, the privacy requirements can be summarized as follows:

Issue 5. (Privacy Requirements) The decentralized data store has to provide privacy of data during routing and storing.

4.4 Availability of Data

Since the decentralized data store relies on the parties storing locally the logging information, the disappearance of a party results in unavailable data. Although, the loss of data either temporarily or permanent is not acceptable in our scenario. However, a guarantee of data availability can not be provided due to the nature of decentralized systems, but the remaining risk can be estimated and the data store can be adapted in accordance. Thus, we require probabilistic guarantees on the data availability.

Due to long running transactions there is no option to specify when a log information will never be required afterwards, therefore an explicit deletion of data is required, although it has to be prevented that logs can be deleted before the completion of the process by all involved parties. Thus, everybody has to agree that the business process has been completed and that the corresponding logging information of that business case can be deleted. In particular, this requirement covers two orthogonal requirements: the first one addresses a consensus making of the involved parties on deleting the log data, while the second one is performing the delete operation in a decentralized data store. We summarize these observations as the requirements on the data store:

Issue 6. (Availability) The decentralized data store has to support availability of data in fluctuating networks with probabilistic guarantees.

and an issue on the organization of the cross-organizational workflow initialization and termination phase:

Issue 7. (Consensus on Deletion) The logging system has to support deletion of log data only after a consensus of all parties involved in the collaboration to delete the data has been reached.

5 Monitoring Solutions

Based on the above investigation of security requirements on the decentralized data store, a brief overview of potential solutions is provided.
5.1 Decentralized File Sharing

Current decentralized storage systems can be roughly divided into the following categories: file-sharing and decentralized storage approaches. The first file-sharing approaches have been Napster followed by Gnutella, KaZaA and eDonkey [10], where every peer makes its files available to the community. They are content-sharing applications, a file-sharing subgroup. Another approach to file-sharing are content-storing approaches, which enable file access for the given community. In particular, every peer offers its disk space that is used by other peers for storing their files. Examples of that category are FreeNet, GNUNet, Past or OceanStore [10].

As a consequence of a thorough investigation of decentralized file sharing, which can not presented here due to a lack of space, it can be derived that content-sharing approaches are inappropriate since they do not provide the required integrity of the logged data. Although, the content-storing approaches provide this capability as well as some support for high data availability exists partially, the privacy requirement of the storing party is not provided.

5.2 Distributed Hash Table

Further research in decentralized systems proposed Distributed Hash Tables (DHT) as the next generation low-level decentralized storage approach. DHTs are quite mature and many implementations like Pastry, Tapestry, P-Grid, CAN and Chord [10] are available.

Distributed Hash Tables (DHT) are low-level structured decentralized systems that provide a consistent way of routing information to the final destination, can handle the changes in topologies and, have an API similar to the hash table data structure. Thus, log data (representing the content object) can be stored in a DHT by using a key (e.g. generated by a hash function).

Due to the peer-to-peer communication the originator of the content and the final storing peer remain unknown to the peers in general. There are some options to derive partial knowledge of especially the storing peer derived from the routing tables maintained locally, but this option is limited to a fraction of the key space only. Thus, the privacy requirement (see Issue 5) seems to be covered in most of the cases by this approach.

Unfortunately, a DHT layer as available right now does not guarantee the availability of data it manages. Whenever a peer goes offline, locally stored data become inaccessible. However, there are approaches recently under development, where higher data availability can be reached by enabling redundancy in two ways: replicating data many times, or using erasure coding to code data and dividing them into many blocks. Although erasure coding provides in general lower storage costs [11], it has has been argued in [12] that the resulting system design get’s more complicated. At a first glance the aim is to start with a more simple approach and applying optimizations after the remaining issues have been resolved. Therefore, we choose replication as a way to achieve data availability in a DHT as required for the auditing (see Issue 5). A potential solution providing probabilistic guarantees has been proposed in [13]. Therefore, the easiest case is to have a DHT with insertions only, but providing an option to delete entries e.g.
by maintaining a revocation list or using a similar approach. As a consequence, the integrity requirement (see Issue 4) can easily be achieved.

6 Related Work

In contrast to the approach for modeling cross-organizational workflows using a synchronous communication model, cross-organizational workflow models based on asynchronous communication have been proposed e.g. by v.d.Aalst [15] or Martens [16]. However, these workflow models do not support decentralized decision making due to the complexity of the selected formal model.

Alternative approaches of reconstructing global states based on local information are known from distributed computing, where a group of parties wants to know the system state. The challenge here is to provide sufficient state information to all parties without influencing the overall system too much by the additionally introduced communication. A class of solutions is known as snapshot protocols, like e.g. [17], where state information is reduced to safe points representing the end points of a concrete transaction to be used for synchronization. A similar problem is the multi-party private computation problem [18] addressing the issue of computing a function value without revealing the private parameters. Again, here all parties involved in the calculation can start deriving the function value at any point in time. The difference to the one addressed in this paper is that the cross-organizational workflow state needs only to be reconstructed in case of a complaint as opposed to a continues provisioning of this service.

7 Conclusion and Future Work

The paper identifies issues related to secure logging of decentralized cross-organizational workflows and underlying decentralized data storage. While the latter ones are investigated with regard to content-sharing, content-storing and Distributed Hash Tables, the issues related to the used communication model, the state representation and the number of resolvable malicious actions, the integrity of the public workflows, and the deletion of logging data based on a consensus of all parties have not been addressed in this paper. In particular, these issues will be investigated in future work as a basis for a secure logging system. Please be aware that the issues raised so far are related to static workflows, that is, workflows not changing over time. Obviously, dynamic workflows introduce an additional complexity to the monitoring as discussed in this paper.

References


