Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 236 (2009) 33–46
www.elsevier.com/locate/entcs

Secure Web Service Workflow Execution

Carsten Rudolph[1] Nicolai Kuntze[2] Zaharina Velikova[3]

Fraunhofer Institute for Secure Information Technology (SIT), Darmstadt, Germany

Abstract

In this paper we identify specific security requirements for distributed workflows and provide a decentralized workflow execution mechanism that ensures their satisfaction. With our composition concept we ensure that each web service can access only the information which is needed for the correct execution of the invoked operations and we provide an execution proof of the fulfilled assignments. Our approach relies on a data structure, called *process slip*, which is passed among the web services participating in the composition.

Keywords: decentralized workflow execution, security requirement, electronic process slip

1 Introduction

A Workflow Management Systems (WFMS) is often used to support the automated execution of business processes. Nowadays the World Wide Web provides new opportunities of performing such business processes, namely by deploying different web services. A standard for specifying such workflow processes is the Web Services Business Process Execution Language (WSBPEL) [2], or BPEL in short. A web service workflow can be defined as a set of interacting web services or a *web service composition*, in which it is determined which web services participate in the process, the order of their interactions and which data is transferred during the process. Web service compositions are used to automate the coordination between participating "partners" thereby increasing the efficiency of the whole process. There exist two different types of interaction between the single web services in a workflow:

- *service orchestration (centralized)* refers to those workflows, in which there exists one central service that receives the client requests, makes the required data transformations and invokes the component web services.

[1] Email: carsten.rudolph@sit.fraunhofer.de
[2] Email: nicolai.kuntze@sit.fraunhofer.de
[3] Email: zaharina.velikova@sit.fraunhofer.de

doi:10.1016/j.entcs.2009.03.013

- *service choreography (decentralized)* refers to the workflows in which there are multiple engines, each executing a composite web service specification (a small part of the original composite web service specification but complete in itself) at distributed locations. The engines communicate directly with each other (rather than through a central coordinator) to transfer data and control when necessary in an asynchronous manner.

In the centralized system model one or more workflow engines, where each of them is able to interpret the process definition, interact with workflow participants and, where required, invoke the use of IT tools and applications. On one hand centralized systems provide centralized monitoring and auditing, simpler synchronization mechanisms and overall design simplicity. But on the other hand, as stated in [26], the centralized approach of web service compositions suffers from some drawbacks. Centralized service architectures are designed as classical client/server applications in which the server provides most of the functionality of the system while the computational potential at the client side is barely used. This can result in inefficient systems when many parallel instances need to be executed. Systems with client/server architecture are normally vulnerable to server failures. They have difficulties with system configuration and actions requiring modification and update of the centralized server, which is very inconvenient and inefficient. From a security perspective a central server introduces a single point of failure where a number of different partners share one system which is under the control of one single partner. In addition to these availability issues this leads to organizational implications related to the trustworthiness of the overall concept.

In contrast to the centralized approach, in a decentralized workflow each partner can be aware of the actual state of the workflow and its involvement in the workflow. The decentralized WFMS should be able to distribute the tasks to the appropriate partners, and ensure specified task dependencies by sending the tasks to the predetermined partners only when all prerequisite conditions are satisfied. However, decentralized execution of inter-organizational workflows may raise a number of security issues including integrity, non-repudiation and confidentiality.

In this paper we provide a decentralized workflow execution mechanism that ensures the correctness of the control flow and the satisfaction of main security requirements. With our composition approach we ensure that each web service can access only the information which is needed for the correct execution of the invoked operations and we provide an execution proof of the fulfilled assignments. At the end of the workflow the integrity and the authenticity of the provided execution proof can be verified by a central verification unit. The main idea of our approach is that the communication and data transfer between the web services participating in the composition is based on a particular data structure, called *process slip*.

In the following sections we present our concept of the process slip. Section 2 presents the technological foundation and briefly discusses related concepts. The concept of distributed workflows and their specific security issues are explored in Section 3. The structure of our container is presented in Section 4. In Section 5 we describe the integration of the process slip in a decentralized workflow and show

how the security properties, given in Section 3, are realized by the integration of our container. This section also contains a brief description of the concept of a *security stub* and explains in details our workflow execution mechanism. In the conclusion we discuss planned further research in this area.

2 Related Work

Related work in the field of process security for distributed workflows is scarce. On the level of document security the TransiDoc Project [22] Piechalski and Schmidt proposed a scheme to extend the concept of transformations of paper documents to their digital counterparts. Thus, each change to a document (e.g. a translation) can be digitally signed and non-repudiation of processes is achieved.

Web services based on WSDL, SOAP, and UDDI are developed to a standard in the communication from machine to machine [1]. The WS-* family of protocols cover a variety of middleware related topics like messaging [16], transactions [11,10], and security [17]. Best practice patterns and tools for the application of security and trust mechanisms already exist for these protocols [24]. In complex environments process definition languages like BPEL [2] are used in combination with additional concepts as defined for example by the Enterprise Service Bus [4]. This allows for flexible solutions with respect to integration of new services and changes in the processes supported by the system design.

Workflows defined by BPEL are abstractions of business processes and have inherent security requirements with respect to their order of execution and the data they contain. Vagts [25] proposes a decentralized framework for workflow execution that ensures exception safety and considers security issues. Secure execution orders are also considered by Biskup et al. [3] supporting the case of a decentralized system by proposing a container structure with authentication mechanisms for data access. A similar approach is shown by Charfi and Mezini [5]. They introduce a framework for providing middleware support in BPEL engines based on a process container, which intercepts the execution of BPEL processes and calls dedicated middleware services to add support for security, persistence and reliable messaging. This is done by using an aspect oriented approach similar to AspectJ [9].

In scenarios with central authority of a dedicated workflow management system existing WS-* protocols can offer a certain level of security on the link level. Bilal et al. [12] show how to describe non-repudiation protocols in BPEL. In particular, they propose and verify non-repudiation protocol using Petri Nets for chain-linked business transactions and show that they may be specified in BPEL.

Besides the technological basis for the provisioning of security features the definition of security requirements and policy definition for each BPEL workflow has to be considered. Work exists in the automated creation of BPEL from choreography languages like WS-CDL [13] or UMM [8]. However, existing choreography languages are missing features to define non-repudiation or complex end to end confidentiality. Nevertheless, high level definition languages are a promising approach to create consisting security policies for workflow definitions.

3 Security Issues and Distributed Workflows

A wide variety of security requirements can occur for workflows and even more for distributed workflows. Usually these requirements are either identified based on static trust relations on the organizational or IT infrastructure level, or for single services within the workflow. However, in particular in the case of decentralized workflows it is not trivial to derive the correct combination of security mechanisms that satisfies all requirements of the different partners involved in the workflow. Therefore, it is necessary to precisely define security requirements on the level of the workflow itself. A full formal framework for the specification of security requirements for workflows is out of the scope of this paper, but we can refer to the generic framework for security requirements by Gürgens et al. [7] which is used in the EU FP7 project SERENITY [4] for the formal specification of workflow security requirements in the context of security services.

Workflow security requirements can be, for example, concerned with the authenticity of an entity performing a particular service in a workflow, integrity or confidentiality of data that is transported between entities involved in the workflow, or the enforcement of particular (distributed) sequences of actions (or services) in the workflow. A combination of security mechanisms for single services as well as for the overall workflow can be required to satisfy all these different security properties. The main contribution of this paper is one particular security mechanism, namely the *electronic process slip* for decentralized workflows. In such a workflow, different partners can be service providers but also take control for parts of the workflow. There is no clear distinction between client, service provider and workflow engine. This situation increases the complexity of security requirements. In the following paragraphs the discussion of security requirements for distributed workflows concentrates on those requirements that can be satisfied by electronic process slips. We distinguish four classes of properties: authenticity, non-repudiation, confidentiality and enforcement of workflow sequences.

Authenticity

In general, authenticity of a particular action is satisfied for one partner P in a workflow if P can deduce that this action has occurred from its knowledge about the global behaviour of the workflow system (including all partners and also including possible malicious behaviour) and the view of the current behaviour that the action has occurred. Stronger authenticity requirements can restrict the occurrence of the action to be authentic to the current instance of the workflow or even to a particular phase of the current instance of the workflow. In a centrally controlled workflow, a workflow engine or in service-oriented architectures an enterprise service bus are in a position to control the workflow and reliably log and report all actions in the workflow. Furthermore, central control also obviously allows a workflow engine to enforce particular sequences or workflow behaviour. Thus, in centralized workflows authenticity can usually be reduced to authenticity of single service executions. In

[4] http://www.serenity-project.org/

distributed decentralized workflows all players need to contribute to the control and enforcement of authenticity.

Non-repudiation

Non-repudiation is strongly related to authenticity but in addition requires that one partner can prove to other partners that a particular action or sequence of actions has occurred. Thus, the partner executing those action cannot repudiate this execution. Again, a trusted central entity could simply collect evidence (e.g. digital signatures) and in case of dispute provide this evidence for dispute resolution. In a decentralized workflows partner have to collect evidence and might even have to rely on other partners for the enforcement of non-repudiation requirements. If security is based on such a mutual trust between partners in a distributed workflow, overall trust requirements have to be considered in the design of the workflow and the security policies.

Confidentiality

A large variety of information can be required to be confidential in a workflow. This information can include data transferred or computed within the workflow, identity information of the partners involved, order of execution of workflow actions, parts of the workflow specification, security policies, cryptographic keys, and audit trails or evidence collected for non-repudiation. The electronic process slip introduced in Section 4 can provide security mechanisms for a large part of this variety of requirements. It can also be used as a basis to reason about the requirements in order to find conflicting requirements or contradictions. Formalization of confidentiality is often based on non-interference and information flow properties formalizing the requirement that the occurrence of some actions cannot interfere with the behaviour of those partners not allowed to gain knowledge about these actions. Non-interference properties can be used to formalize confidentiality of workflow data towards external attackers. However, non-interference properties are not suitable for properties within a workflow where a partner might know about the occurrence of some or all workflow actions but might not be allowed to know the values of all parameters in the workflow data. The notion of *parameter confidentiality* [6] is more suitable to formalise confidentiality requirements for workflows.

Enforcement of sequences

Security of a workflow very often depends on the order of actions in the workflow. A particular action can depend on a number of other actions occurred before or a particular binding phase can only be finished if all goals of the involved partners are satisfied. Again, a trusted central entity can enforce these properties and reduce these rather complex requirements to requirements for single actions or services within the workflow. It should be noted that this reduction is not always as easy as in the case of authenticity of particular actions. It might require a combination of mechanisms for confidentiality, authenticity and non-repudiation.

The lack of central control in distributed workflows increases the complexity of the problem. It might be even impossible to achieve the most efficient decentralization if control has to be given to an untrusted entity and if it is necessary to actually prevent a deviation from the workflow specification. Additional communication between partners might be required to release confidential data that is necessary to continue with the workflow. A realization is easier if it is satisfactory to detect violations of the requirement after they have occurred. Many of these different combinations of security requirements can be realized by the electronic process slip introduced in the following section.

4 A Process Slip

Traditional paper based workflows use signed reports carrying the relevant information. These reports are produced wherever it is required to grant the owner the access to certain services. This token based concept can also be applied to digital business processes by establishing a trustworthy data structure as a technical basis for the workflow.

By removing a central authority it is required to distribute the control of the workflow among the managers which are involved into the single (sub)workflows. Additionally we define a control token traveling from manager to manager, which is a digital form of a report description of the execution of a workflow process. It transfers all kinds of data between the single managers and stores their digital signatures as proofs for the executed tasks. Moreover, the webservice workflow uses the process slip to provide the necessary security benefits to these execution records such as restricted access, integrity and non-repudiation in order to prevent further unauthorized access of any kind. The process slip must be generated prior to the initiation of each instance of the workflow. This must be done either by the initiator of the workflow or by a service invoked by the initiator.

Our process slip structure contains four subcontainers (see Figure 1): *Data*, *Audit Data*, *Security Policies* and *Workflow Description*.

Fig. 1. Data components of a process slip

- **Data** - This subcontainer stores the input data transferred between the consecutive steps of the process. Therefore, these data are completely variable from

step to step. Additional security may be introduced by adding encryption and integrity means to achieve minimal need to know and preserve integrity. This leads to a structured container format supporting addressing of the data to certain steps resp. participants of the workflow. According to the needed security level some (or all) data can be encrypted and signed for the corresponding recipient to ensure confidentiality and integrity of the data. Access control to the encrypted data is regulated by using tickets, a Public Key infrastructure (PKI), identity management systems, or pre-shared keys for example. As an alternative to embed the data into the subcontainer it is also possible to include a redirection to an alternative source providing the data. This could be implemented by using a ticket scheme. Access control in this case can also be restricted using a method as presented above.

- **Audit Data** - In the audit trail data subcontainer each involved manager should write according to the workflow or security definition log data for the recently performed tasks. Each service is only allowed to add process information and documentation data relevant for the service's assignments and their output. In particular, a web service partner stores in the audit subcontainer for each fulfilled task the following information:
  · `requestPartner`, specifying the URL of the service from which the request was received,
  · `requestOperation`, giving the request operation to be invoked,
  · `requestAssignment`, storing the label(s) of the fulfilled assignments.
  These documentation data is encrypted for special recipients such as a central verification unit or partner services which according to the underlying security policies have as requirement for their task the currently fulfilled assignment. Additionally each report entry in this section must be signed by its author to guarantee the integrity of the written information. By adding an authenticated log of each step to the final result it is later on possible to track down the process and assign responsibility. This provides non-repudiation to the audit data entries.
- **Security Policy** - Security policies are situated in a separate data structure which is logically linked to the workflow definition. They specify security boundaries for each step or sequence of steps in the workflow such as the permitted activities that a given partner can apply on a specified data and the execution order of the assignments in the workflow. Each manager has to be able to interpret and enforce these policies accordingly as there is no central authority which grants for it. To ensure that each service will have access only to the policy rules needed to execute the assigned tasks and fill in the information, needed for the next partners, the initiator could encrypt the rules for the corresponding authorized service. By encrypting the rules each partner service is limited in his knowledge to the absolute minimum of knowledge regarding the workflow he is working in. Additional data structures are required to support the creation of audit data in this case.
- **Workflow Description** - The whole process is controlled by a workflow description in the container. This static data structure is not changed during the

execution of the workflow and can either be a full description of the workflow or a reference to a location where the definition is kept. According to the needed security level the partner services may have only access to the description of their own activities. This implies that during the generation of the container some (or all) parts of the workflow description must be encrypted for the corresponding services, so that they are able to invoke the corresponding actions but do not have access to the activities of the other partners. Thus authentication and integrity may be added by XML Signature, XML Security, or WS-Security. In case of a reference it should be noted that the integrity means are created nonetheless. If the reference is changed or not available the workflow has to perform a roll back.

Whenever a web service participating in the composition receives a process slip container it performs the following tasks:

(i) first, it verifies that the sender is an authorized partner;

(ii) second, it extracts from the slip the audit data of the sender and its security policy in order to check whether all requirements for its current assignments are met;

(iii) it decrypts the needed input data if there exists such in the Data subcontainer of the slip;

(iv) it invokes the corresponding operations and fulfills his assignments;

(v) when all operations are terminated, it modifies correspondingly the received process slip;

(vi) finally, it sends the modified slip container to partner(s) associated with the next control structure(s).

5 Realization of Global Workflow Security Policies

In this section we will describe the integration of the process slip container in a decentralized workflow and the realization of the described security properties by the integration of our container. First in 5.1 we will specify the notation used to present the structure and the functionality of our workflow execution concept. In subsection 5.2 we briefly explain the concept of a security stub utilized to handle the process slip container. In the last subsection we give a detailed description of the proposed workflow execution.

5.1 Workflow Model

A workflow is defined as the *automation of a business process*, during which documents, information or tasks are passed from one participant to another, according to a set of procedural rules [18]. It can be represented as partially ordered set of interrelated assignments (A) that lead to accomplishing a specific predefined goal. The order of the assignment execution can be managed by specifying interdependencies between the input and output event or data of each assignment. In this way we can achieve chains of activities which specify the exact order of the different

assignments in the scope of the corresponding workflow. Each partner, engaged in the process, is assigned to a particular subset of assignments.

In order to be able to give an abstract representation of a distributed workflow we first introduce the following entities. Let:

- $P = \{p_1, p_2, \ldots, p_n\}$ be the set of the partners, engaged in the workflow.
- $D = \{d_1, d_2, \ldots, d_s\}$ be the data set shared by the partners of a workflow.
- $A = \{a_1, a_2, \ldots, a_m\} = \{(in_1, out_1), (in_2, out_2), \ldots, (in_m, out_m)\}$ be a set of assignments, where each assignment is specified by its input and output data ($A \subseteq D \times D$).
- $ACT = \{act_1, act_2, \ldots, act_k\}$ be a set of k activities which define the possible data access privileges.
- $POL = \{pol_1, pol_2, \ldots, pol_l\}$ be a set of l security policies.

Furthermore, we specify the following relations between these entities, which actually define the content of the secure policies:

- $assign : P \rightarrow A_k = \{a_1, a_2, \ldots, a_p\} \subset A$ specifies the assigned tasks to a partner p_k;
- $access : P \times D \rightarrow \{ACT\}_{i=1,\ldots,i}$ which gives the permitted activities that a given partner can perform on a given data;

One possibility to ensure that our framework fulfills the corresponding security properties (see Section 3) is to use a PKI, which involves generation of public and private keys and public key certificates for each service, deployed during the workflow. Since the process of key and certificate generation and distribution is out of the scope of this work we assume the existence of a trusted CA, that can generate keys and certificates for the workflow participants and the existence of a service, responsible for their secure distribution among the partners. If new parties participate in interactions during the handling process, we assume that their key pairs and certificates are also correspondingly generated and distributed. Therefore we can assume that prior to the execution of a workflow each participant p_i receives a public/private key pair $\left(p_i^{pub}, p_i^{priv}\right)$ for asymmetric encryption and digital signatures and a public key certificate, digitally signed by the corresponding CA. For simplicity we assume that the same keypair is used for encryption and digital signatures. For practical realizations one would distinguish signature and encryption keys.

We denote with W the abstract representation of a distributed workflow defined by

$$W = \left\{iid_W, (p_i)_{i\in[1,n]}, (A_i)_{i\in[1,n]}, \mathcal{C}\right\},$$

where iid_W is a unique string, identifying an instance of the workflow, p_i is a partner web service, A_i is the set of its assignments in the workflow and $\mathcal{C}$ is the set of the inter dependencies between the separate assignments.

5.2 The Security Stub

The structure of the process slip container (see Section 4) is not a standard one and to require that each partner service, deployed in the composition, implements extra functions to parse and use it, is a very strong requirement. Therefore, in order to assure that each web service, deployed in the workflow, will be able to handle the process slip container properly and to use its functionality, we apply a generic approach by defining a *Security Stub* (see Figure 2). This stub wraps around the core functionality of the web service and provides all the extra functions needed to handle our process slip. The main task of the *Security Stub* is to parse the audit and security data, to extract and enforce the corresponding security policies. During this process it is also required that the stub verifies all relevant security means to testify the integrity, authenticity and authorization properly. If all these tasks are performed successfully the core service is invoked and with the results of the service invocation a new process slip audit and data block are created according to the security and workflow policies.

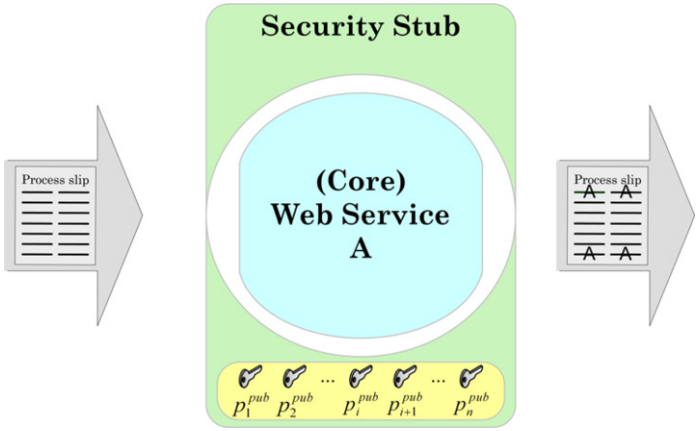


Fig. 2. Security stub

5.3 Workflow Execution

In this section we will describe the main steps involved in our custom workflow execution. As we already outlined, the generation and the distribution of the public/private key pairs and the identity certificates of the partners is not a matter of this work. Therefore we assume that there exists a trusted CA and the keys and the certificates have been generated and securely distributed among all possible partners. We also assume that the workflow definition already exists and the partners which are supposed to participate in the workflow execution are fixed [5].

The first step is the initialization of the process slip components. This is performed by the initiator of the workflow p_1. We can split this process in the following steps:

[5] If the partners are not fixed the initiator of the workflow must perform a service location procedure (e.g. UDDI search) during the workflow initiation phase.

- **Generating the Security Policies** The security policies must correspond the security requirements of the different tasks and the workflow definition. They must also authorize all activities, which the partners should be able to perform on the different sections of the process slip (e.g. some services may be granted a *read* or *write* access to the *Audit Data* section but no one can be allowed to delete anything from this section). To prevent an unauthorized access to the service's policies the initiator may encrypt each policy with the public key of the corresponding service.
- **Initialization of the *Data* and the *Audit Data* Sections**: Since the workflow is still in initialization phase, there exists no actual and audit data to be shared among the partner services. Therefore the *Data* and the *Audit Data* sections are only initialized.

Once the initialization phase is complete, the workflow execution proceeds with the delegation of the execution to the service responsible for the first set of tasks. Let assume that we are in the i-th step of the workflow execution, where the next set of tasks $A_i = assign(p_i)$ must be performed by p_i and p_{i-1} has just finished its assigned tasks A_{i-1} and has sent the process slip to p_i. Then the workflow execution follows with the following scenario:

(i) The security stub of p_i extracts the policy pol_i from the *Security Policies* section of the slip.

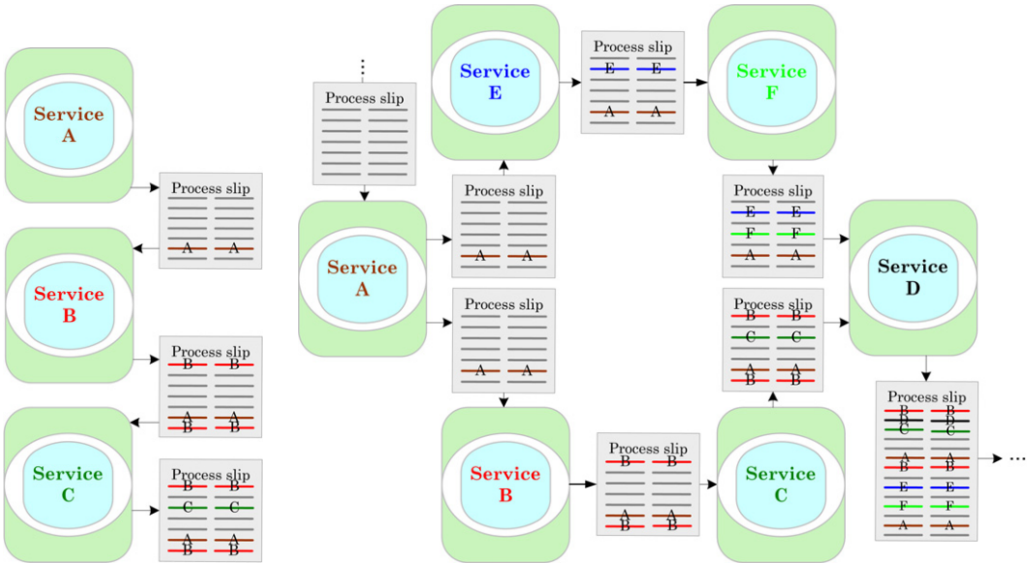
(ii) After that the data $\{d_i\}_{i\in[1,t]}$, located in the *Data* section of the container and encrypted for p_i, is decrypted and the core web service of p_i is invoked.

(iii) After p_i has completed the assigned tasks A_i the security stub of p_i fills in the *Audit Data* section, according to the specified rules in pol_i. It must first encrypt the task report entries for the central verification unit. Then the security stub of p_i must sign the *Audit Data* section by inserting a digital signature of the *Audit Data* section. The signature is on a message digest of the whole content of the section and on a time stamp, which is proving creation time of the signature. If there exists output data $D_i = \{d_i\}_{i\in[1,k]}$ of the currently completed tasks which must be confidential for a partner p_j, then it encrypts the corresponding data with p_j^{pub} and inserts the data and a digital signature (on the inserted data and on a time stamp, proving the time of the insertion) in the *Data* section.

(iv) Once all previous activities have been completed the process slip is sent to the next service partner, which continues the execution of the workflow.

If in the workflow there exist service partners which cannot be trusted or which join the workflow for the first time then they must authenticate themselves in front of the initiator or to each service with which they communicate. In order to do this they send their public key certificate to the service they want to communicate with. In order to verify that a received certificate is not revoked the security stub relies on some communication with external security services. These are Public Key Infrastructure services which are used to verify certificate validity by querying the

certificate authority using the Online Certificate Status Protocol (OCSP) [15]. In order to provide this functionality we introduce in our workflow model the Trusted Third Party (TTP) service, which implements the functionality of the verifying CA. Therefore, once the request from such a partner p_{i-1} is received the security stub of the receiver p_i first sends to the TTP an OCSP request that contains a fingerprint of p_{i-1}'s public key. Then the TTP sends a signed OCSP response back to p_i and p_i verifies the TTP's response (each partner in the process has access to the TTP's public key and TTP is a trusted service) and ensures that the request has come from an authorized partner.



(a) Sequence process slip composition

(b) Process slip composition with parallel workflows

Fig. 3. Composition of the process slip

During the process execution there exist two basic variants of execution scenarios for the composition of the process slip. The basic variant is serial execution. In this case every service works on the results of the previous service. Each service inserts its own entries in the *Audit Data* section and signs the whole section. Figure 3(a) describes this approach. It depicts the sequence composition of a process slip, where the services A, B and C insert their entries one after another. In this way the resulting structure of signatures resembles an onion structure [14] with several layers.

Figure 3(b) illustrates the second case of parallel workflows. In this case at a certain point in time the workflow forks and has to be merged at a later step. In this case the "join" service (depicted by service D in Figure 3(b)), which is in charge for the merge, receives all results from the sub-workflows and creates one signature around the whole data package. It is to be noted that in case of parallel workflow branches the "join" service should wait for the input process slips of all parallel branches so that no information is lost during the merging process. Therefore, here

it is important to establish an adequate workflow control to ensure that all sub-workflows meet again in one service to be merged. More detailed information about methods for merging business processes can be found in [23] .

6 Conclusion

Distribution of workflow execution offers advantages for the key characteristics of WFMSs availability and information governance due to the removal of the central entity which is in charge for the workflow execution. Moreover, by applying our concept non-repudiation can be added as a new feature to the functional library of workflow based systems.

The security treatment (e.g. allow access to data or encrypt data for specific partners etc.) in our approach is determined by security policies. Thus the selection of proper specification of security policy rules and their systematic generation is crucial for the fulfillment of the specified security requirements. Therefore the process of generation of security policies, which will be able to ensure a proper level of security in the common case, still needs some further research.

Automated processes between machines are increasingly required by the industry to meet the trend towards highly flexible environments. In this machine to machine (M2M) case each entity performs actions in the mandate of the owner. Combining the presented concept with hardware based security signature concepts, stating the integrity of each entity, opens the way for non-repudiation in M2M processes.

Till now little effort has been dedicated to verification of the modeled business processes. For example, there is no support to detect possible deadlocks, to detect parts of the process that are not viable or to verify specific security requirements. Using asynchronous product automata (APA) and the simple homomorphism verification tool (SHVT) [21,19,20], developed in the Fraunhofer SIT, we intend to formally specify and verify the presented approach, and thus prove that the execution of such workflow instances is secure with respect to the identified security requirements.

References

[1] Alonso, G., F. Casati, H. Kuno and V. Machiraju, "Web Services: Concepts, Architecture and Applications," Springer Verlag, 2004.

[2] Alves, A., A. Arkin, S. Askary, C. Barreto, B. Bloch, F. Curbera, M. Ford, Y. Goland, N. Kartha, C. K. Liu, S. Thatte, P. Yendluri, A. Yiu, A. Guízar, R. Khalaf, D. König, M. M. V. Mehta and D. van der Rijn, *Web services business process execution language version 2.0* (2007).
URL http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf

[3] Biskup, J., B. Carminati, E. Ferrari, F. Muller and S. Wortmann, *Towards secure execution orders for compositeweb services*, in: *ICWS'07: Proceedings of the IEEE International Conference on Web Services*, pp. 489–496.

[4] Chappell, D. A., "Enterprise service bus," O'Reilly Media, Inc., 2004.

[5] Charfi, A. and M. Mezini, *An aspect-based process container for BPEL*, in: *AOMD'05: Proceedings of the 1st ACM workshop on Aspect Oriented Middleware Development* (2005).

[6] Gürgens, S., P. Ochsenschläger and C. Rudolph, *Abstractions preserving parameter confidentiality*, in: *ESORICS'05: Proceedings of the European Symposium On Research in Computer Security*, 2005, pp. 418–437.

[7] Gürgens, S., P. Ochsenschläger and C. Rudolph, *On a formal framework for security properties*, International Computer Standards & Interface Journal (CSI), Special issue on formal methods, techniques and tools for secure and reliable applications (2005).

[8] Hofreiter, B. and C. Huemer, *Transforming UMM Business Collaboration Models to BPEL*, in: R. Meersman, Z. Tari and A. Corsaro, editors, *OTM Workshops*, Lecture Notes in Computer Science **3292** (2004), pp. 507–519.

[9] Laddad, R., "AspectJ in Action: Practical Aspect-Oriented Programming," Manning Publications Co., Greenwich, CT, USA, 2003.

[10] Langworthy, D., editor, "WS-BusinessActivity," 2004.

[11] Langworthy, D., editor, "WS-Transaction," 2004.

[12] M Bilal, J. P. T., M. Thomas and S. Abraham, *Fair BPEL Processes Transaction using Non-Repudiation Protocols*, in: *SCC'05: Proceedings of the 2nd IEEE International Conference on Services Computing* (2005), pp. 337–342.

[13] Mendling, J. and M. Hafner, *From Inter-organizational Workflows to Process Execution: Generating BPEL from WS-CDL*, in: R. Meersman, Z. Tari, P. Herrero, G. Méndez, L. Cavedon, D. Martin, A. Hinze, G. Buchanan, M. S. Pérez, V. Robles, J. Humble, A. Albani, J. L. G. Dietz, H. Panetto, M. Scannapieco, T. A. Halpin, P. Spyns, J. M. Zaha, E. Zimányi, E. Stefanakis, T. S. Dillon, L. Feng, M. Jarrar, J. Lehmann, A. de Moor, E. Duval and L. Aroyo, editors, *OTM Workshops*, Lecture Notes in Computer Science **3762** (2005), pp. 506–515.

[14] Montagut, F. and R. Molva, *Enforcing integrity of execution in distributed workflow management systems*, in: *SCC'07: Proceedings of the 4th IEEE International Conference on Services Computing*, 2007.

[15] Myers, M., R. Ankney, A. Malpani, S. Galperin and C. Adams, *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP*, RFC 2560 (Proposed Standard) (1999).
URL http://www.ietf.org/rfc/rfc2560.txt

[16] OASIS, *WS-Reliability* (2004).

[17] OASIS, *WS-Security version 1.0* (2004).

[18] Object Management Group, Inc., *Workflow management facility specification, v1.2* (2000).
URL http://www.omg.org/docs/formal/00-05-02.pdf

[19] Ochsenschläger, P., J. Repp and R. Rieke, *Abstraction and composition: a verification method for co-operating systems*, J. of Experimental and Theoretical Artificial Intelligence **12** (2000), pp. 447–459.
URL http://www.sit.fhg.de/english/META/meta_publications/doc/flairs-2000c.pdf

[20] Ochsenschläger, P., J. Repp and R. Rieke, *The SH-Verification Tool*, in: J. Etheredge and B. Manaris, editors, *FLAIRS'00: Proceedings of the 13th International FLorida Artificial Intelligence Research Society Conference* (2000), pp. 18–22.

[21] Ochsenschläger, P. and R. Rieke, *Abstraction Based Verification of a Parameterised Policy Controlled System*, in: *MMM-ACNS'07: Proceedings of the 4th International Conference on Mathematical Methods, Models and Architectures for Computer Network Security*, CCIS **1** (2007).

[22] Piechalski, J. and A. U. Schmidt, *Authorised Translations of Electronic Documents*, in: H. S. Venter, J. H. P. Eloff, L. Labuschagne and M. M. Eloff, editors, *ISSA'06: Peer-reviewed Proceedings of the From Insight to Foresight Conference* (2006).
URL
http://www.math.uni-frankfurt.de/~aschmidt/docs/Authorised_Translations_ISSA06_final.pdf

[23] Sun, S., A. Kumar and J. Yen, *Merging workflows: a new perspective on connecting business processes*, Decis. Support Syst. **42** (2006), pp. 844–858.

[24] Tatsubori, M., T. Imamura and Y. Nakamura, *Best-Practice Patterns and Tool Support for Configuring Secure Web Services Messaging*, in: *ICWS'04: Proceedings of the IEEE International Conference on Web Services* (2004), p. 244.

[25] Vagts, H.-H., "Control Flow Enforcement in Workflows in the Presence of Exceptions," Master's thesis, TU Darmstadt (2007).
URL http://www.sec.informatik.tu-darmstadt.de/pages/dipl/docs/finished/vagts_diplom.pdf

[26] Yan, J., Y. Yang and G. K. Raikundalia, *SwinDeW-A p2p-Based Decentralized Workflow Management System*, IEEE Transactions on Systems, Man, and Cybernetics Part A - Systems and Humans (2006), pp. 922–935.