Bridging Security and Fault Management within Distributed Workflow Management Systems

Frederic Montagut, *Member, IEEE,* Refik Molva, *Member, IEEE,*

Abstract— **As opposed to centralized workflow management systems, the distributed execution of workflows can not rely on a trusted centralized point of coordination. As a result, basic security features including compliance of the overall sequence of workflow operations with the pre-defined workflow execution plan or traceability become critical issues that are yet to be addressed. Besides, the detection of security inconsistencies during the execution of a workflow usually implies the complete failure of the workflow although it may be possible in some situations to recover from the latter. In this paper, we present security solutions supporting the secure execution of distributed workflows. These mechanisms capitalize on onion encryption techniques and security policy models in order to assure the integrity of the distributed execution of workflows, to prevent business partners from being involved in a workflow instance forged by a malicious peer and to provide business partners' identity traceability for sensitive workflow instances. Moreover, we specify how these security mechanisms can be combined with a transactional coordination framework in order to recover from faults that may be caught during their execution. The defined solutions can easily be integrated into distributed workflow management systems as our design is strongly coupled with the runtime specification of decentralized workflows.**

Index Terms— **Decentralized workflows, Security, Fault management**

I. Introduction

Distributed workflow management systems [1], [2] eliminate the need for a centralized coordinator that can be a performance bottleneck in some business scenarios. Because of this flexibility, the execution of workflows in the decentralized setting raises new security requirements as opposed to usual centralized workflow management systems. Distributed workflow systems can not indeed rely on a trusted centralized coordination mechanism to manage the most basic execution primitives such as message routing between business partners. As a result, basic security features such as integrity of workflow execution assuring the compliance of the overall sequence of operations with the pre-defined workflow execution plan are no longer guaranteed. In addition, tracing back the identity of the business partners involved in a distributed workflow instance becomes an issue without a trusted centralized coordination mechanism selecting workflow participants. Yet, existing decentralized workflow management systems do not incorporate the appropriate mechanisms to meet the new security requirements in addition to the ones identified in the centralized setting. Even though some recent research efforts in the field of distributed workflow security have indeed been focusing on issues related to the management of rights in business partner assignment or detecting conflicts of interest [3],[4],[5], basic security issues related to the security of the overall workflow execution such as integrity and evidence of execution have not yet been addressed. Besides, faults that may occur during the execution of security mechanisms are often considered unrecoverable and lead to the complete failure of a workflow instance while in some failure scenarios some simple recovery mechanisms can be specified to resume a workflow execution that has failed.

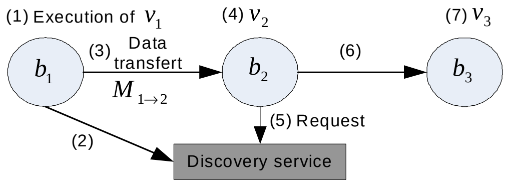


Fig. 1. Pervasive workflow runtime

In this paper, we present security solutions supporting the secure execution of distributed workflows. These mechanisms capitalize on onion encryption techniques [6] and security policy models in order to assure the integrity of the distributed execution of workflows, to prevent business partners from being involved in a workflow instance forged by a malicious peer and to provide business partners' identity traceability for sensitive workflow instances. The design of the suggested mechanisms is strongly coupled with the runtime specification of decentralized workflow management systems which eases their integration into existing distributed workflow management solutions. Moreover, we specify how these security mechanisms can be combined with a transactional coordination framework in order to recover from faults that may be caught during the execution of these security solutions so that the detection of security inconsistencies no longer imply the complete failure of a workflow execution.

The remainder of the paper is organized as follows. In section II, we give an overview of the distributed workflow management system and the coordination protocol that will be used throughout the paper to illustrate our approach. Section III outlines the security requirements associated with the execution of workflows in the decentralized setting. In section IV our solution is specified while in section V the runtime specification of the secure distributed workflow execution is presented. Section VI presents the security analysis of the proposed mechanisms. In section VII we present how the security solutions we propose in this paper can be integrated within the transactional protocol presented in section II. Finally section IX discusses related work and section X presents the conclusion.

II. Workflow model and fault management system

The workflow management system used to support our approach was designed in [2]. This model called pervasive workflow

Manuscript received ; revised . This work has been partially sponsored by EU IST Directorate General as a part of FP6 IST project R4eGov and by SAP Labs France S.A.S.

F. Montagut is with SAP Labs France, 805 av Dr. Donat, 06250 Mougins, France

R. Molva is with Institut Eurecom, 2229 Route des Cretes, 06904 Sophia-Antipolis, France

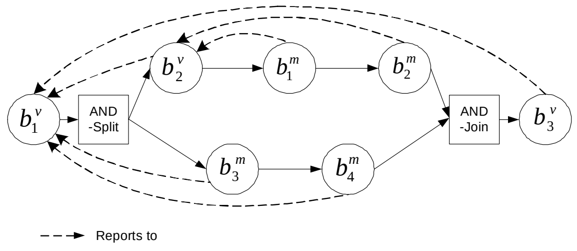


Fig. 2. Protocol actors

supports the execution of business processes in environments without infrastructure and features a distributed architecture characterized by two objectives:

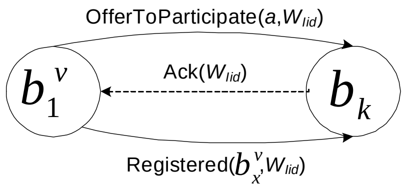
- fully decentralized: the workflow management task is carried out by a set of business partners in order to cope with the lack of dedicated infrastructure
- dynamic assignment of business partners to workflow tasks: the actors can be discovered at runtime

Having designed an abstract representation of the workflow whereby business partners are not yet assigned to tasks, a partner launches the execution and executes a first set of tasks. Then the initiator searches for a partner able to perform the next set of tasks. Once the discovery phase is complete, a workflow message including all data is sent by the workflow initiator to the newly discovered partner and the workflow execution further proceeds with the execution of the next set of tasks and a new discovery procedure. The sequence composed of the discovery procedure, the transfer of data and the execution of a set of tasks is iterated till the final set of tasks. In this decentralized setting, the data transmitted amongst partners include all workflow data. We note W the abstract representation of a distributed workflow defined by $W = \{(v_i)_{i\in[1,n]}, \delta\}$ where v_i denotes a vertex which is a set of workflow tasks that are performed by a business partner from the receipt of workflow data till the transfer of data to the next partner and δ is the set of execution dependencies between those vertices. We note $(M_{i\to j_p})_{p\in[1,z_i]}$ the set of workflow messages issued by b_i to the z_i partners assigned to the vertices $(v_{j_p})_{p\in[1,z_i]}$ executed right after the completion of v_i. The instance of W wherein business partners have been assigned to vertices is denoted $W_b = \{W_{iid}, (b_i)_{i\in[1,n]}\}$ where W_{iid} is a string called workflow instance identifier. This model is depicted in figure 1. In this paper, we only focus on a subset of execution dependencies or workflow patterns namely, SEQUENCE, AND-SPLIT, AND-JOIN, OR-SPLIT and OR-JOIN. In order to adequately support the execution of critical workflow instances in this decentralized setting, we proposed in a previous work [7] a transactional coordination protocol whose main features are summarized hereafter.

A. Fault management

The coordination protocol we designed meets the two main requirements relevant to the execution of critical workflow instances in a dynamic and distributed setting:

- **Relaxed atomicity**: atomicity of the workflow execution can be relaxed as intermediate results produced by the workflow may be kept despite the failure of one partner. The specification process of transactional requirements associated with workflows has to be flexible enough to support coordination scenarios more complex than the coordination rule “all or nothing” specified for the two phase commit protocol [8].



OfferToParticipate(a,W_{iid})
Ack(W_{iid})
Registered(b_x^v,W_{iid})
b_1^v
b_k

Fig. 3. Business partner registration

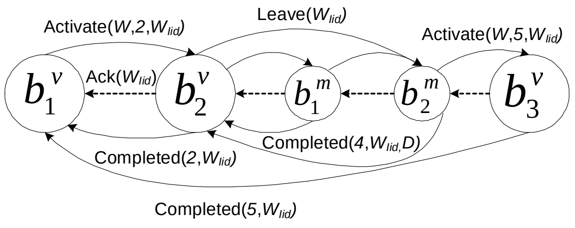


Fig. 4. Normal execution

- **Dynamic assignment of business partners**: the workflow execution is dynamic in that the workflow partners offering different characteristics can be assigned to tasks depending on the resources available at runtime. Business partners' characteristics have thus to be combined or composed in a way such that the transactional requirements specified for the workflow are met.

The execution of this protocol is managed in a centralized way by the workflow initiator b_1^v. The role of the coordinator consists in issuing coordination decisions based on the state of the workflow execution. The coordination is assured in a hierarchical way as some business partners $(b_k^v)_{k\in[1,j]}$ play the role of subcoordinators and report directly to b_1^v whereas the remaining partners $(b_k^m)_{k\in[1,l]}$ report to the business partner b_x^v most recently executed [1]. This reporting strategy is depicted in figure 2.

The execution of this protocol takes place in two phases. The first phase consists of the discovery and registration of all the partners that will be involved in the critical instance in order to combine the properties offered by available business partners [9] in a way such that the workflow execution does not lead to any inconsistent outcomes. The discovery process through which business partners that can be assigned to tasks are identified is performed by the initiator of the workflow by means of a registration handshake depicted in figure 3. The coordinator b_1^v contacts a business partner asking it whether it agrees to commit to execute the operation a of the workflow whose identifier is W_{iid}. Once the newly assigned business partner's coordinator is known, b_1^v sends the information.

Once all involved business partners are known, the workflow execution can start supported by the coordination protocol. Business partners are sequentially activated based on the workflow specification. A sample for normal execution of a workflow is depicted in figure 4. The $Activate(W, k, W_{iid}, D)$ message is a workflow message defined in [4], it especially contains the workflow specification W, the requested task k to be executed, the workflow data D modified during the execution and the workflow identifier W_{iid}. Within the workflow execution local acknowledgments $Ack(W_{iid})$ are used. Each business partner reports its status to the workflow initiator and once its execution is

[1]business partner b_k^v that is located on the same branch of the workflow as these b_k^m business partners and that has most recently completed its execution.

complete it can leave the execution. The $Completed(k, W_{iid}, D)$ message sent by a business partner includes a backup copy of the data modified by the business partner that can be reused later on for the recovery procedure in case of failure. In the presence of recoverable faults, business partners can be indeed replaced using this backup message.

III. Security requirements

As opposed to centralized workflow management systems the distributed execution of workflows raises security constraints due to the lack of a dedicated infrastructure assuring the management and control of the workflow execution. As a result, security features such as compliance of the workflow execution with the pre-defined plan are no longer assured. We group the security requirements we identified for distributed workflow management systems into three main categories: authorization, proofs of execution and data protection.

A. Authorization

The main security requirement for a workflow management system is to ensure that only authorized business partners are assigned to workflow tasks during an instance. In the decentralized setting, the assignment of workflow tasks is managed by partners themselves relying on a service discovery mechanism. In this case, the business partner assignment procedure enforces a matchmaking procedure whereby business partners' security credentials are matched against security requirements for tasks.

B. Execution proofs and traceability

A decentralized workflow management system does not offer any guarantee regarding the compliance of actual execution of workflow tasks with the pre-defined execution plan. Without any trusted coordinator to refer to, the business partner b_i assigned to the vertex v_i needs to be able to verify that the vertices scheduled to be executed beforehand were actually executed according to the workflow plan. This is a crucial requirement to prevent any malicious peer from forging a workflow instance.

In our workflow execution model, candidate business partners are selected at runtime based on their compliance with a security policy. Partners' involvement in a business process can thus remain anonymous as their identity is not assessed in the partner selection process. In some critical business scenarios however, disclosing partners' identity may be required so that in case of dispute or conflict on the outcome of a sensitive task the stakeholders can be identified. In this case, the revocation of business partners' anonymity should only be feasible for some authorized party in charge of arbitrating conflicts, preserving the anonymity of identity traces is thus necessary.

C. Workflow data protection

In the case of decentralized workflow execution, the set of workflow data denoted $D = (d_k)_{k \in [1,j]}$ is transferred from one business partner to another. This raises major requirements for workflow data security in terms of integrity, confidentiality and access control as follows:

- **Data confidentiality**: for each vertex v_i, the business partner b_i assigned to v_i should only be authorized to read a subset D_i^r of D

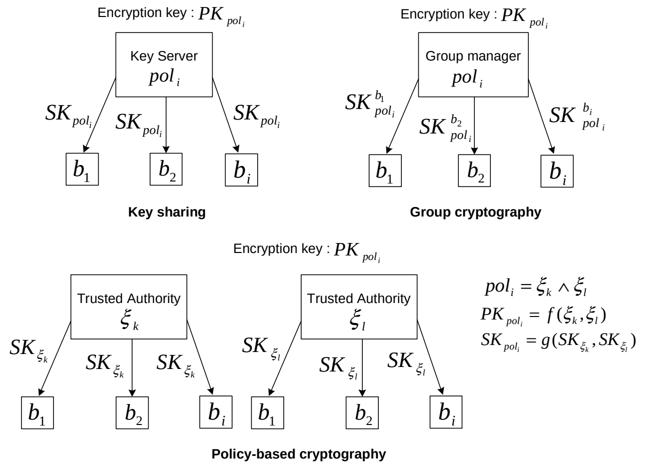


Fig. 6. Policy private key distribution schemes

- **Data integrity**: for each vertex v_i, the business partner b_i assigned to v_i should only be authorized to modify a subset D_i^w of D_i^r
- **Access control**: the subsets D_i^r and D_i^w associated with each vertex v_i should be determined based on the security policy of the workflow

The solution we developed towards meeting these security requirements is presented in the next section.

IV. The solution

In this section the mechanisms we designed in order to meet the security requirements we identified for distributed workflow systems are specified. The solution is mainly described in terms of the key management, data protection, execution proofs and communication protocol.

A. Key management

Two types of key pairs are introduced in our approach. Each vertex v_i is first associated with a policy pol_i defining the set of credentials a candidate partner needs to satisfy in order to be assigned to v_i. The policy pol_i is mapped to a key pair (PK_{pol_i}, SK_{pol_i}) where SK_{pol_i} is the policy private key and PK_{pol_i} the policy public key. Thus satisfying the policy pol_i means knowing the private key SK_{pol_i}, the inverse may however not be true depending on the policy private key distribution scheme as explained later on in section VI. The policy private key SK_{pol_i} can indeed be distributed by different means amongst which we distinguish three main types depicted in figure 6:

1) Key sharing: a policy pol_i is associated with a single policy private key that is shared amongst principals satisfying pol_i. A simple key server KS_{pol_i} associated with pol_i can be used to distribute the policy private key SK_{pol_i} based on the compliance of business partners with the policy pol_i. In this case, the business partners satisfying pol_i thus share the same policy private key SK_{pol_i} associated with the encryption key PK_{pol_i}.

2) Policy-based cryptography: a policy pol_i is expressed in a conjunctive-disjunctive form specifying the combinations of credentials ξ_k a principal is required to satisfy to be compliant with the policy: $pol_i = \wedge_{i=1}^{m} [\vee_{j=1}^{m_i} [\wedge_{k=1}^{m_{i,j}} \xi_{i,j,k}]]$ where $\wedge$ represents a conjunction (AND) and $\vee$ a disjunction (OR). A cryptographic scheme [10] is used to map credentials to keys denoted credential keys SK_{ξ_k} that can be combined to

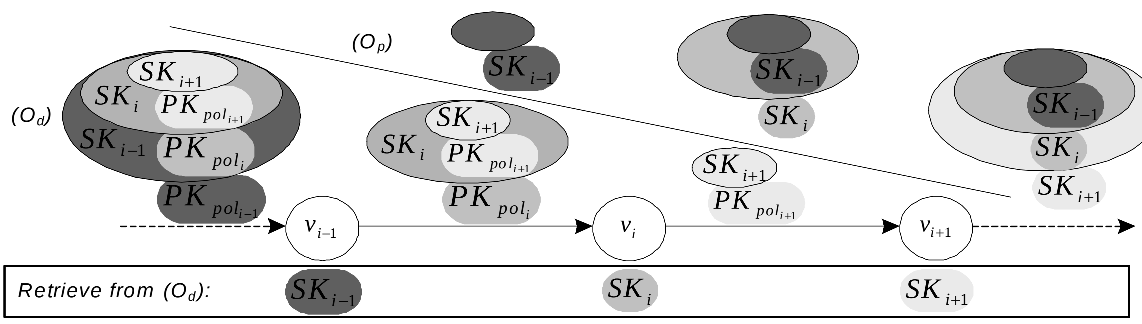


Fig. 5. Key management

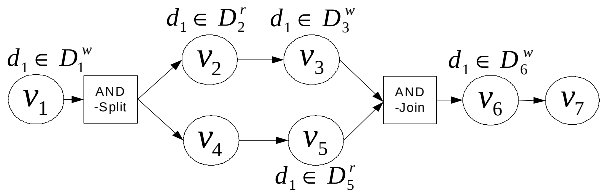


Fig. 7. Workflow example

encrypt, decrypt and sign messages based on a given policy. Some trusted authorities TA are in charge of distributing credential keys to requesters when the latter satisfies some assertions (that can be expressed in a conjunctive-disjunctive form e.g. (jobtittle=director)$\wedge$(company=xcorp)). This scheme provides direct mapping between a policy and some key material and thus eases policy management as opposed to key sharing. No anonymity-preserving traceability solution is however offered as principals satisfying a given assertion may possess the same credential key.

3) Group cryptography: a policy pol_i is mapped to a group structure in which a group manager distributes different policy private keys to group members satisfying the policy pol_i. A single encryption key PK_{pol_i} is used to communicate with group members who however use their personal private key to decrypt and sign messages. This mechanism offers an identity traceability feature as only the group manager can retrieve the identity of a group member using a signature issued by the latter [11]. The policy private key of the business partner b_k is denoted $SK_{pol_i}^{b_k}$. We note GM_{pol_i} the group manager of the group whose members satisfy the policy pol_i. The management of policy key pairs is as complex as for the key server solution since a group structure is required for each specified policy.

Second, we introduce vertex key pairs $(PK_i, SK_i)_{i\in[1,n]}$ to protect the access to workflow data. We suggest a key distribution scheme wherein a business partner b_i whose identity is *a priori* unknown retrieves the vertex private key SK_i upon his assignment to the vertex v_i. Onion encryption techniques with policy public keys PK_{pol_i} are used to distribute vertex private keys. Furthermore, execution proofs have to be issued along with the workflow execution in order to ensure the compliance of the execution with the pre-defined plan. To that effect, we also leverage onion encryption techniques in order to build an onion structure with vertex private keys to assure the integrity of the workflow execution. The suggested key distribution scheme (O_d) and the execution proof mechanism (O_p) are depicted in figure 5 and specified later on in the paper.

In the sequel of the paper, $\mathcal{M}$ denotes the message space, $\mathcal{C}$ the ciphertext space and $\mathcal{K}$ the key space. Using a key $K \in \mathcal{K}$ on

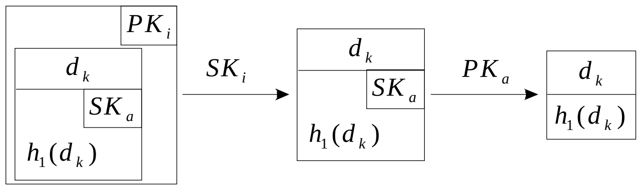


Fig. 8. Access to workflow data

a message $m \in \mathcal{M}$ is noted $\{m\}_K$ (e.g. encryption with a public key, signature with a private key) and h, h_1, h_2 denote one-way hash functions.

B. Data protection

The role of a business partner b_i assigned to a vertex v_i consists in processing the workflow data that are granted read-only and read-write access during the execution of v_i. We define a specific structure depicted in figure 8 called data block to protect workflow data accordingly. Each data block consists of two fields:

- the actual data: d_k
- a signature: $sign_a(d_k) = \{h(d_k)\}_{SK_a}$

We note $B_k^a = (d_k, sign_a(d_k))$ the data block including the data segment d_k that has last been modified during the execution of v_a. The data block B_k^a is also associated with a set of signatures denoted H_k^a that is computed by b_a assigned to v_a.

$$H_k^a = \{\{h(\{B_k^a\}_{PK_l})\}_{SK_a} | l \in R_k^a\}$$

where R_k^a is the set defined as follows. Let $l \in [1, n]$.

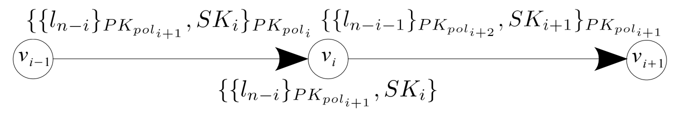
$$l \in R_k^a \Leftrightarrow l \text{ satisfies } \begin{cases} d_k \in D_l^r \\ v_l \text{ is executed after } v_a \\ v_l \text{ is not executed after } v_{p(a,l,k)} \end{cases}$$

where $v_{p(a,l,k)}$ denotes the first vertex executed after v_a such that $d_k \in D_{p(a,l,k)}^w$ and that is located on the same branch of the workflow as v_a and v_l. For instance, consider the example of figure 7 whereby d_1 is in D_1^w, D_2^r, D_3^w, D_5^r and D_6^w, $v_{(1,2,1)} = v_3$, $R_1^1 = \{2, 3, 5, 6\}$ and $R_1^3 = \{6\}$.

When the business partner b_i receives the data block B_k^a three cases can occur:

1) b_i is granted read access on d_k: B_k^a is encrypted with PK_i and b_i decrypts the structure using SK_i in order to get access to d_k and $sign_a(d_k)$. b_i is then able to verify the integrity of d_k using PK_a, i.e. that d_k was last modified after the execution of v_a.

2) b_i is granted write access on d_k: b_i can (in addition to what is possible in the first case since write access implies read access) update the value of d_k and compute $sign_i(d_k)$ yielding a new data block B_k^i and a new set H_k^i.



$\{\{l_{n-i}\}_{PK_{pol_{i+1}}}, SK_i\}_{PK_{pol_i}}$
$\{\{l_{n-i-1}\}_{PK_{pol_{i+2}}}, SK_{i+1}\}_{PK_{pol_{i+1}}}$
v_{i-1}
v_i
v_{i+1}
$\{\{l_{n-i}\}_{PK_{pol_{i+1}}}, SK_i\}$

Fig. 9. SEQUENCE pattern

3) b_i has no right on d_k: b_i receives B_k^a encrypted with PK_m (in this case v_m is executed after v_i) and he can only verify the integrity of $\{B_k^a\}_{PK_m}$ by matching $h(\{B_k^a\}_{PK_m})$ against the value contained in H_k^a.

The integrity and confidentiality of data access thus relies on the fact that the private key SK_i is made available to b_i only, prior to the execution of v_i. The corresponding distribution mechanism is presented in the next section.

C. Vertex private key distribution mechanism

The objective of the vertex private key distribution mechanism is to ensure that only the business partner b_i assigned to v_i at runtime and whose identity is *a priori* unknown can access the vertex private key SK_i. To that effect, the workflow structure in terms of execution patterns is mapped with an onion structure O_d so that at each step of the execution a layer of O_d is peeled off using SK_{pol_i} and SK_i is revealed.

Definition 4-1 (**Onion Structure**). Let X a set. An onion O is a multilayered structure composed of a set of n subsets of X $(l_k)_{k\in[1,n]}$, such that $\forall k \in [1,n]$ $l_k \subseteq l_{k+1}$. The elements of $(l_k)_{k\in[1,n]}$ are called layers of O, in particular, l_1 and l_n are the lowest and upper layers of O, respectively. We note $l_p(O)$ the layer p of an onion O.

Definition 4-2 (**Onion wrapping**). Let $A = (a_k)_{k\in[1,j]}$ and $B = (b_k)_{k\in[1,l]}$ two onion structures, A is said to be wrapped by B, when $\exists k \in [1,l]$ such that $a_j \subseteq b_k$.

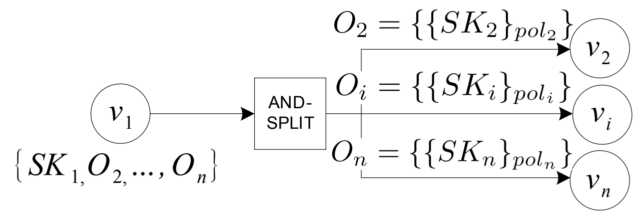
We first present how vertex private keys are distributed to partners with respect to various workflow patterns including SEQUENCE, AND-SPLIT, AND-JOIN, OR-SPLIT and OR-JOIN before describing how those are combined in the execution of a complete workflow.

1) SEQUENCE workflow pattern: Vertex private keys are sequentially distributed to business partners. In this case, an onion structure assuring the distribution of vertex private keys is sequentially peeled off by business partners. Considering a sequence of n vertices $(v_i)_{i\in[1,n]}$, the business partner b_1 assigned to the vertex v_1 initiates the workflow execution with the onion structure O defined as follows.

$$O : \begin{cases} l_1 = \{SK_n\} \\ l_i = \left\{\{l_{i-1}\}_{PK_{pol_{n-i+2}}}, SK_{n-i+1}\right\} \text{ for } i \in [2,n] \\ l_{n+1} = \left\{\{l_n\}_{PK_{pol_1}}\right\} \end{cases}$$

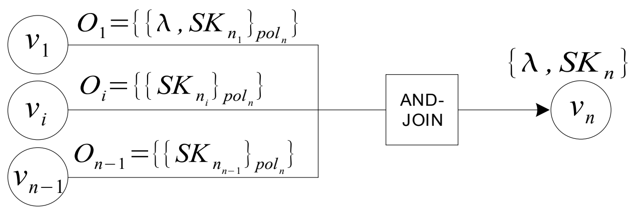
The onion layers are iteratively wrapped to match the SEQUENCE pattern and the workflow execution proceeds as depicted in figure 9. For $i \in [2, n-1]$ the business partner b_i assigned to the vertex v_i receives $\{l_{n-i+1}(O)\}_{PK_{pol_i}}$, peels one layer off by decrypting it using SK_{pol_i}, reads $l_{n-i+1}(O)$ to retrieve SK_i and sends $\{l_{n-i}(O)\}_{PK_{pol_{i+1}}}$ to b_{i+1}.

2) AND-SPLIT workflow pattern: In the case of the AND-SPLIT pattern, the business partners $(b_i)_{i\in[2,n]}$ assigned to the vertices $(v_i)_{i\in[2,n]}$ are contacted concurrently by b_1 assigned to the vertex v_1. In this case, $n-1$ vertex private keys should be delivered to $(b_i)_{i\in[2,n]}$ and the upper layer of the onion O_1 available to b_1 therefore wraps SK_1 and $n-1$ onions $(O_i)_{i\in[2,n]}$ to be sent to $(b_i)_{i\in[2,n]}$ as depicted in figure 10.



$O_2 = \{\{SK_2\}_{pol_2}\}$
v_2
v_1
AND-SPLIT
$O_i = \{\{SK_i\}_{pol_i}\}$
v_i
$O_n = \{\{SK_n\}_{pol_n}\}$
v_n
$\{SK_1, O_2, \ldots, O_n\}$

Fig. 10. AND-SPLIT pattern



v_1
$O_1 = \{\{\lambda, SK_{n_1}\}_{pol_n}\}$
v_i
$O_i = \{\{SK_{n_i}\}_{pol_n}\}$
v_{n-1}
$O_{n-1} = \{\{SK_{n_{n-1}}\}_{pol_n}\}$
AND-JOIN
$\{\lambda, SK_n\}$
v_n

Fig. 11. AND-JOIN pattern

$$O_1 = \{SK_1, O_2, O_3, .., O_n\}$$
$$O_i = \left\{\{SK_i\}_{PK_{pol_i}}\right\} \text{ for } i \in [2,n]$$

3) AND-JOIN workflow pattern: Since there is a single workflow initiator, the AND-JOIN pattern is preceded in the workflow by an AND-SPLIT pattern. In this case, the vertex v_n is executed by the business partner b_n if and only if the latter receives $n-1$ messages as depicted in figure 11. In order to meet this requirement, the vertex private key SK_n is divided into $n-1$ parts and defined by

$$SK_n = SK_{n_1} \oplus SK_{n_2} \oplus ... \oplus SK_{n_{n-1}}$$

The key SK_{n_i} is simply included in the onion O_i sent by b_i to b_n. Besides, in order to avoid redundancy, the onion structure λ associated with the sequel of the workflow execution right after v_n is only included in one of the onions received by b_n. Each $(b_i)_{i\in[1,n-1]}$ therefore sends the onion O_i defined as follows to b_n.

$$O_1 = \{\{\lambda, SK_{n_1}\}_{PK_{pol_n}}\}$$
$$O_i = \left\{\{SK_{n_i}\}_{PK_{pol_n}}\right\} \text{ for } i \in [2, n-1]$$

4) OR-SPLIT workflow pattern: This is an exclusive choice and the business partner b_1 assigned to the vertex v_1 only needs to send one message.

$$O_1 = \{SK_1, O_2, O_3, .., O_n\}$$
$$O_i = \left\{\{SK_i\}_{PK_{pol_i}}\right\} \text{ for } i \in [2,n]$$

The onion O_1 is available to the business partner b_1. This is the same structure as the AND-SPLIT pattern, yet the latter only sends the appropriate onion O_i to the business partner assigned to the vertex v_i depending on the result of the condition associated with the OR-SPLIT pattern.

5) OR-JOIN workflow pattern: Since there is a single workflow initiator, the OR-JOIN is preceded in the workflow by an OR-SPLIT pattern. The business partner assigned to v_n receives in any cases a single message thus a single vertex private key is required that is sent by one of the business partners $(b_i)_{[1,n-1]}$ depending on the choice made at the previous OR-SPLIT in the workflow. the business partner b_n thus receives in any cases the onion O defined as follows.

$$O = \left\{\{\lambda, SK_n\}_{PK_{pol_n}}\right\}$$

where λ is an onion structure associated with the sequel of the workflow execution right after v_n.

6) Complete key distribution scheme: The procedure towards building an onion structure corresponding to the workflow structure can be implemented using for instance a breath first search algorithm starting from the last vertex of the workflow and wherein the workflow graph is read backward. This is rather straightforward and the procedure is only sketched throughout an example. Let's consider the workflow depicted in figure 7. The onion O_d enabling the vertex private key distribution during the execution of the workflow is defined as follows.

$$O_d = \{\{SK_1, \underbrace{\{SK_2, \{SK_3, \{SK_{6_1}, \overbrace{\{SK_7\}_{PK_{pol_7}}}^{\text{Sequel after} v_6}}_{\text{First AND-SPLIT branch}}$$

$$\underbrace{\}_{PK_{pol_6}}\}_{PK_{pol_3}}\}_{PK_{pol_2}}}_{\text{First AND-SPLIT branch}}, \underbrace{\{SK_4, \{SK_5, \{SK_{6_2}}_{\text{Second AND-SPLIT branch}}$$

$$\underbrace{\}_{PK_{pol_6}}\}_{PK_{pol_5}}\}_{PK_{pol_4}}}_{\text{Second AND-SPLIT branch}}\}_{PK_{pol_1}}\}$$

The onions associated with the two branches forming the AND-SPLIT pattern are wrapped by the layer corresponding to the vertex v_1. Only the first AND-SPLIT branch includes the sequel of the workflow after v_6. The overall structure of the onion is of course based on the SEQUENCE pattern.

D. Execution proofs and traceability

Along with the workflow execution, an onion structure O_{p_i} is built at each execution step i with vertex private keys in order to allow business partners to verify the integrity of the workflow execution and optionally to gather anonymity-preserving traces when traceability is required during the execution of a workflow. Based on the properties we introduced in section IV-A, group cryptography is the only mechanism that meets the needs of the policy private key distribution when identity traceability is needed. In that case, we define for a workflow instance, the workflow arbitrator role that is assumed by a trusted third party.

Definition 4-3 (**Workflow arbitrator**). The workflow arbitrator, denoted W_{ar}, is a trusted third party able to disclose business partners' identity in case of dispute. The workflow arbitrator is contacted to revoke the anonymity of some business partners only in case of dispute, this is an optimistic mechanism.

The onion structure O_p is initialized by the business partner b_1 assigned to v_1 who computes $O_{p_1} = \big\{\{h(P_W)\}_{SK_{pol_1}}\big\}$ where P_W is called workflow policy and is defined as follows.

Definition 4-4 (**Workflow Policy**). The workflow specification S_W denotes the set S_W defined by $S_W = \{W, (J_i^r, J_i^w, pol_i)_{i\in[1,n]}, h\}$ where

$$J_i^r = \{k \in [1, j] | d_k \in D_i^r\} \text{ and } J_i^w = \{k \in [1, j] | d_k \in D_i^w\}$$

The sets J_i^r and J_i^w basically specify for each vertex the set of data that are granted read-only and read-write access, respectively. S_W is defined at workflow design phase.

The workflow policy P_W denotes the set defined by:

$$P_W = S_W \cup \{W_{iid}, W_{ar}, h_1, h_2\} \cup \{PK_i | i \in [1, n]\}$$

P_W is a public parameter computed by the workflow initiator b_1 and that is available to the business partners involved in the execution of W.

The onion structure O_p is initialized this way so that it cannot be replayed as it is defined for a specific instance of a workflow specification. If traceability is required during the execution of some business processes, the signatures of business partners with policy private keys are collected during the building process of O_p so that anonymity can be later on revoked in case of dispute. Group encryption is used in this case to distribute policy private keys and the business partner b_1 is in charge of contacting a trusted third party when the workflow is instantiated, sending it $(h(P_W), P_W)$ to play the role of workflow arbitrator for the instance.

At the step i of the workflow execution, b_i receives $O_{p_{i-1}}$ and encrypts its upper layer with SK_i to build an onion O_{p_i} which he sends to b_{i+1} upon completion of v_i. If traceability is required, b_i signs $\{O_{p_{i-1}}, \{h(P_W)\}_{SK_{pol_i}^{b_i}}\}$ with SK_i instead. Considering a set $(v_i)_{[1,n]}$ of vertices executed in sequence assigned to the business partners $(b_i)_{[1,n]}$ and assuming that traceability is needed (i.e. group cryptography is used) we get:

$$O_{p_1} = \big\{\{h(P_W)\}_{SK_{pol_1}^{b_1}}\big\}$$
$$O_{p_2} = \big\{\{O_{p_1}, \{h(P_W)\}_{SK_{pol_2}^{b_2}}\}_{SK_2}\big\}$$
$$O_{p_i} = \big\{\{O_{p_{i-1}}, \{h(P_W)\}_{SK_{pol_i}^{b_i}}\}_{SK_i}\big\} \text{ for } i \in [3, n]$$

The building process of O_{p_i} is based on workflow execution patterns ; yet since it is built at runtime contrary to the onion O_d, this is straightforward:

- There is no specific rule for OR-SPLIT and OR-JOIN patterns as during the workflow execution, this will result in the execution of a single branch (exclusive choice),
- When encountering an AND-SPLIT pattern, the same structure O_{p_i} is concurrently sent while in case of an AND-JOIN, the $n-1$ onions received by a partner b_n are wrapped by a single structure:

$$O_{p_n} = \big\{\{O_{p_1}, O_{p_2}, .., O_{p_{n-1}}, \{h(P_W)\}_{SK_{pol_n}^{b_n}}\}_{SK_n}\big\}$$

.

Considering the example depicted in figure 7 and assuming traceability is not required, at the end of the workflow execution the onion O_p is defined as follows.

$$O_p = \{\{\{\underbrace{\{\{\{h(P_W)\}_{SK_{pol_1}}\}_{SK_2}\}_{SK_3}}_{\text{First AND-SPLIT branch}},$$
$$\underbrace{\{\{\{h(P_W)\}_{SK_{pol_1}}\}_{SK_4}\}_{SK_5}}_{\text{Second AND-SPLIT branch}}\}_{SK_6}\}_{SK_7}\}$$

$\{h(P_W)\}_{SK_{pol_1}}$ is sent by b_1 assigned to v_1 to both b_2 and b_4 assigned to v_2 and v_4, respectively. The onion structure associated with the two branches forming the AND-SPLIT pattern thus includes $\{h(P_W)\}_{SK_{pol_1}}$ twice. The overall structure is mapped to the SEQUENCE pattern and layers are iteratively wrapped by a new layer as the workflow instance proceeds further.

In order to verify that the workflow execution is compliant with the pre-defined plan when he starts the execution of the vertex v_i, the business partner b_i assigned to v_i just peels off the layers of $O_{p_{i-1}}$ using the vertex public keys of the vertices previously executed based on S_W. Doing so he retrieves the value $\{h(P_W)\}_{SK_{pol_1}}$ that should be equal to the one he can compute given P_W, if the workflow execution has been so far executed according to the plan. In case traceability is required by the execution, b_i also verifies the signatures of the business partners assigned to the vertices $(v_{j_p})_{p\in[1,k_i]}$ executed right before him i.e.

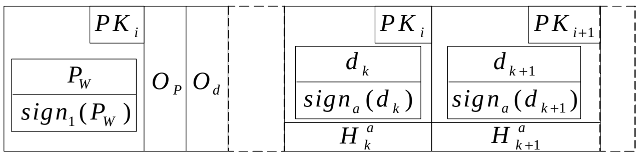


Fig. 12. Workflow message structure

b_i verifies $\{h(P_W)\}_{SK^{b_p}_{pol_p}}$ for all $p \in [1, k_i]$. If b_i detects that a signature is missing he contacts the workflow arbitrator W_{ar} to declare the workflow instance inconsistent. In fact, business partners are in charge of contacting the workflow arbitrator when a signature is not valid and those who do not declare corrupted signatures are held responsible in place of partners whose signature is missing. In case of conflict on the outcome of some workflow tasks, the onion O_p is sent to the workflow arbitrator who is able to retrieve the signatures with policy private key of the stakeholders using P_W and with the help of some group managers the corresponding identities.

E. Vertex key pair generation

Vertex key pairs have to be defined for a single instance of a workflow specification in order to avoid replay attacks. To that effect, we propose to capitalize on ID-based encryption techniques [12] in the specification of the set $(PK_i, SK_i)_{i\in[1,n]}$. For all $i \in [1, n]$ (PK_i, SK_i) is defined by:

$$\begin{cases} PK_i = h_1(W_{iid} \oplus S_W \oplus v_i) \\ SK_i = sh_2(PK_i) \end{cases}$$

where $s \in \mathbb{Z}_q^*$ for a prime q. s is called master key and is held by the vertex private key generator [12] who is in our case the workflow initiator. The signature scheme proposed in [13] can be used to compute the ID-based signatures required by the mechanisms we proposed. The public parameters such as the system public key (usually called P_{pub}) should be included in P_W. This vertex key pair specification has a double advantage. First vertex key pairs cannot be reused during any other workflow instance and second vertex public keys can be directly retrieved from W and W_{iid} when verifying the integrity of workflow data or peeling off the onion O_p.

F. Communication protocol

In order to support a coherent execution of the mechanisms presented so far, workflow messages exchanged between business partners consist of the set of information that is depicted in figure 12.

- **Workflow data**: the set $(d_k)_{k\in[1,j]}$ of workflow data is transported between business partners and each piece of data satisfy the data block specification. A single message may include several copies of the same data block structure that are encrypted with different vertex public keys based on the execution plan. This can be the case with AND-SPLIT patterns. Besides, workflow data can be stored in two different ways depending on the requirements for the execution. Either we keep the iterations of data resulting from each modification in workflow messages till the end of the execution or we simply replace data content upon completion of a vertex. The bandwidth requirements are higher in the first case since the size of messages increases as the workflow execution proceeds further.
- P_W: P_W is required to retrieve vertex and policy public keys and specifies the workflow execution plan.
- O_d and O_p: the two onion structures O_d and O_p are also included in the message.

Upon receipt of the message depicted in figure 12 a business partner b_i assigned to v_i retrieves first the vertex private key from O_d. He then checks that P_W is genuine i.e. that it was initialized by the business partner initiator of the workflow assigned to v_1. He is later on able to verify the compliance of the workflow execution with the plan using O_p and the integrity of workflow data. Finally he can process workflow data.

V. Secure execution of decentralized workflows

In this section we specify how the mechanisms presented so far in this paper are combined to support the secure execution of a workflow in the decentralized setting. After an overview of the execution steps, the secure workflow execution is described in terms of the workflow initiation and runtime specifications.

A. Execution process overview

Integrating security mechanisms to enforce the security requirements identified for the decentralized execution of workflows requires a process strongly coupled with both workflow design and runtime specifications. At the workflow design phase, the workflow specification S_W is defined in order to specify for each vertex the sets of data that are accessible in read and write access and the credentials required by potential business partners to be assigned to workflow vertices. At workflow initiation phase, the workflow policy P_W is specified and the onion O_d is built. The workflow initiator builds then the first set of workflow messages to be sent to the next partners involved. This message generation process consists of the initialization of the data blocks and that of the onion O_p.

At runtime, a business partner b_i chosen to execute a vertex v_i receives a set of workflow messages. Those messages are processed to retrieve SK_i from the onion O_d and to access workflow data. Once the vertex execution is complete b_i builds a set of workflow messages to be dispatched to the next partners involved in the execution. In this message building process, the data and the onion O_p are updated.

The set of functional operations composing the workflow initiation and runtime specifications is precisely specified later on in this section. In what follows N^i_k denotes the set defined as follows. Let $l \in [1, n]$

$$l \in [1, n] \Leftrightarrow l \text{ satisfies } \begin{cases} d_k \in D^r_l \\ v_l \text{ is executed right after } v_i \end{cases}$$

Consider the example of figure 7: d_1 is accessed during the execution of the vertices v_1, v_2 and v_5 thus $N^1_1 = \{2, 5\}$.

B. Workflow initiation

The workflow is initiated by the business partner b_1 assigned to the vertex v_1 who issues the first set of workflow messages $(M_{1\to j_p})_{p\in[1,z_1]}$. The workflow initiation mainly consists of the following steps.

Step 1 : Workflow policy specification: generate $(PK_i, SK_i)_{i\in[1,n]}$ and assign W_{ar}

Step 2 : Initialization of the onion O_d

Step 3 : Data block initialization: compute $\forall k \in [1,j]$ $sign_1(d_k)$
Step 4 : Compute N_k^1 and R_k^1 $\forall k \in [1,j]$
Step 5 : Data block encryption: compute $\forall k \in [1,j], \forall l \in N_k^1$ $\{B_k^1\}_{PK_l}$
Step 6 : Data block hash sets: compute $\forall k \in [1,j], \forall l \in R_k^1$ $\{h(\{B_k^1\}_{PK_l})\}_{SK_1}$
Step 7 : Initialization of the onion O_p: compute O_{p_1}
Step 8 : Message generation based on W and $(N_k^1)_{k\in[1,j]}$

The steps one and two are presented in sections IV-E and IV-C, respectively. The workflow messages are generated with respect to the specification defined in figure 12 and sent to the next business partners involved. This includes the initialization of the onion O_p and that of data blocks which are encrypted with appropriate vertex public keys.

C. *Workflow message processing*

A business partner b_i being assigned to a vertex v_i proceeds as follows upon receipt of the set of workflow messages $(M_{j_p \to i})_{p\in[1,k_i]}$ sent by the k_i business partners assigned to the vertices $(v_{j_p})_{p\in[1,k_i]}$ executed right before v_i.

Step 1 : Retrieve SK_i from O_d
Step 2 : Data block decryption with SK_i based on J_i^r
Step 3 : Execution proof verification: peel off the onion O_p
Step 4 : Data integrity check based on W and P_W
Step 5 : Vertex execution
Step 6 : Compute N_k^i $\forall k \in J_i^r$ and R_k^i $\forall k \in J_i^w$
Step 7 : Data block update: compute $\forall k \in J_i^w$ $sign_i(d_k)$ and update d_k content
Step 8 : Data block encryption: compute $\forall k \in J_i^r, \forall l \in N_k^i$ $\{B_k^i\}_{PK_l}$
Step 9 : Data block hash sets: compute $\forall k \in J_i^w, \forall l \in R_k^i$ $\{h(\{B_k^i\}_{PK_l})\}_{SK_i}$
Step 10 : Onion O_p update: compute O_{p_i}
Step 11 : Message generation based on W and $(N_k^i)_{k\in[1,j]}$

After having retrieved SK_1 from O_d, b_i verifies the integrity of workflow data and that the execution of the workflow up to his vertex is consistent with the onion O_p. Workflow data are then processed during the execution of v_i and data blocks are updated and encrypted upon completion. Finally b_i computes O_{p_i} and issues the set of workflow messages $(M_{i\to j_p})_{p\in[1,z_i]}$ to the intended business partners.

VI. Security analysis

The parameters that are relevant to the security properties offered by the mechanisms presented in this paper are mainly twofold. First, there are several alternatives with respect to the management of the key pair (PK_{pol_i}, SK_{pol_i}), including simple key distribution based on the policy compliance, group key management or policy-based cryptography, on which the security properties verified by our solution depend. In fact, the main difference between the three policy private key distribution schemes we identified comes from the number of business partners sharing the same policy private key. As a matter of fact, the more partners share a given private key the easier it is for some unauthorized peer to get this private key and get access to protected data. Besides, the trustworthiness of business partners can not be controlled, especially when it comes to sharing workflow data with unauthorized peers once the vertex private key has been retrieved. In this context, the mechanisms presented in this paper verify some properties that do not depend on the underpinning policy private key distribution scheme while some other do. In the security evaluation of our solution, we make two assumptions:

- **Security of policy keys**: the public key encryption scheme used in the specification of the policy key pair (PK_{pol_i}, SK_{pol_i}) is semantically secure against a chosen ciphertext attack and the associated signature scheme achieves signature unforgeability.
- **Security of vertex keys**: the public key encryption scheme used in the specification of the vertex key pair (PK_i, SK_i) is semantically secure against a chosen ciphertext attack and the associated signature scheme achieves signature unforgeability.

The theorems presented in this section have been proven in a previous work [14] and are only reminded.

A. *Inherent security properties*

Theorem 6-1 (**Integrity of execution**). *Vertex private keys are retrieved by business partners knowing policy private keys associated with the policies specified in the workflow.*

Assuming in addition that business partners do not share vertex private keys, the integrity of the distributed workflow execution is assured i.e. workflow data are accessed and modified based on the pre-defined plan specified by means of the sets J_i^r and J_i^w.

Theorem 6-2 (**Resilience to instance forging**). *Upon receipt of a workflow message, a business partner is sure that a set of business partners knowing policy private keys associated with the policies specified in the workflow have been assigned to the vertices executed so far provided that he trusts the business partners satisfying the policy pol_1.*

Theorem 6-3 (**Data Integrity**). *Assuming that business partners do not share vertex private keys they retrieve from the onion O_d, our solution achieves the following data integrity properties:*

- *Data truncation and insertion resilience: any business partner can detect the deletion or the insertion of a piece of data in a workflow message*
- *Data content integrity: any business partner can detect the integrity violation of a data block content in a workflow message*

These three security properties are sufficient to enable a coherent and secure execution of distributed workflows provided that business partners are trustworthy and do not share their policy or vertex private keys. The latter assumption is in fact hard to assess when sensitive information are manipulated during the workflow. We therefore introduced the traceability mechanism to meet the requirements of sensitive workflow executions.

B. *Revocation of a business partner anonymity*

The main flaw of the basic security mechanisms we outlined is that the involvement of business partners in a workflow can remain anonymous thus preventing the detection of potential malicious peers who somehow got access to some policy private keys. To overcome this limitation when required, traceability with group cryptography has to be used during the execution of a business process. In this case the anonymity revocation mechanism provided with group cryptography can be seen as a penalty for business partners thus preventing potential malicious behaviors such as vertex private key sharing with unauthorized peers. Besides, policy private keys distributed by a group manager are intended for individual use which makes key leakage highly unlikely.

The following theorems hold when the policy private key distribution scheme is based on group encryption techniques and traceability is required in the execution of workflows. As corollary of this assumption, we assume that vertex private keys are not shared with unauthorized peers, theorem 6-3 is thus verified.

Theorem 6-4 (**Integrity of execution**). *The integrity of the distributed workflow execution is ensured or the workflow instance is declared inconsistent by the selected workflow arbitrator. Integrity of the distributed workflow execution consists in this case in performing the following tasks:*

- *workflow data are accessed and modified based on the predefined plan specified by means of the sets J_i^r and J_i^w ;*
- *signatures with policy private key are stored by the business partners involved in the workflow execution.*

C. Discussion

As mentioned in the security analysis, group cryptography associated with anonymity revocation provides a full-fledged solution that meets the requirements of sensitive workflow instances. The other policy private key distribution schemes can be in fact used when the workflow execution is not sensitive or the partners satisfying the policies required by the workflow are deemed trustworthy. Our solution can still be optimized to avoid the replication of workflow messages. A business partner may indeed send the same workflow message several times to different partners satisfying the same security policy resulting in concurrent executions of a given workflow instance. Multiple instances can be detected by the workflow arbitrator when traceability is required or a solution based on a stateful service discovery mechanism can be also envisioned to solve this problem.

VII. Integration within the coordination protocol

In this section we discuss the possible integration of the security mechanisms specified in this paper within the transactional protocol presented in section II. We thus assume in what follows that the term workflow instance refers to the execution of a distributed workflow that is supported by the transactional protocol presented in section II and that implements the security mechanisms outlined in this paper.

There are various types of security faults that can be raised during the execution of the security mechanisms we have just specified and that need to be handled by the coordination protocol so that a workflow instance can recover when these security faults are caught. In the fashion of typical execution faults a failure recovery strategy has to be defined for security faults. In a first approach, one could regard security faults as failures to execute and directly integrate them into the transactional protocol execution. This approach would however lead to consider a strict atomicity within a workflow instance as the security mechanisms we specified are executed by all the business partners involved in a workflow instance thus making the latter prone to failures. This approach is as a result not suited to meet the requirements that are relevant to assuring consistency of workflow instances with respect to relaxed atomicity constraints. We thus choose to define security-fault handling mechanisms that rely on the workflow arbitrator and thus group cryptography techniques introduced in section IV-D, in order to manage the recovery procedure when security faults occur. In this case, the failure recovery strategy associated with the complete failure of a workflow instance due to a security fault consists in penalizing the business partner that caused the fault. The specification of possible penalties that can be issued to business partners is out of the scope of this work but one could think to legal compensations.

The security-fault handling mechanisms that we designed are integrated into the ones defined in the context of transactional failures in order to first recover from security faults that do not lead to the complete failure of a workflow instance without canceling the workflow instance as well as penalize business partners that may cause security faults from which it is not possible to recover. This approach not only enables the detection of security faults but also their recovery whenever possible while preserving relaxed atomicity constraints required by the execution of workflows. Security solutions are indeed often only considered a means to detect inconsistencies within the execution of applications but the way to properly handle these inconsistencies is left aside. On the contrary, we consider in this work these two aspects, security inconsistencies do not indeed always imply complete failure of a workflow execution.

We first describe the different types of security faults that can be raised during the security mechanism execution before specifying the recovery mechanisms designed to handle the latter.

A. Security faults

There are six main types of security faults that can be encountered during the execution of a distributed workflow instance implementing the security mechanisms we presented, as follows.

1) Data decryption fault: a business partner is not able to decrypt a piece of data that he is allowed to access based on the workflow specification during the execution of the vertex to which he is assigned. The data decryption fault can be raised at anytime during a workflow instance and is forwarded to the coordination protocol as it keeps a business partner from carrying out a vertex execution.

2) Data integrity fault: : a business partner detects that a piece of data has been altered in the workflow message he has just received. The data integrity fault can be raised at anytime during a workflow instance and is forwarded to the coordination protocol as it keeps a business partner from carrying out a vertex execution.

3) Vertex private key retrieval fault: a business partner can not retrieve a vertex private key from the onion O_d. The vertex private key retrieval fault can be raised at anytime during a workflow instance and is forwarded to the coordination protocol as it keeps a business partner from carrying out a vertex execution.

4) Proof of execution fault: a business partner can not assess the validity of the onion O_p i.e. that all signatures are valid. The proof of execution fault can be raised at anytime during a workflow instance and is forwarded to the coordination protocol as it keeps a business partner from carrying out a vertex execution.

5) Discovery fault: : no candidate business partner satisfying the policy associated with a vertex can be found. The discovery fault can only be raised during the replacement of business partners since the transactional protocol execution requires the assignment of all business partners to workflow vertices prior to the workflow instantiation. This fault is in fact equivalent to the failure of the associated vertex and of course is critical enough to be forwarded to the coordination protocol, we however consider it a transactional failure rather than a security fault.

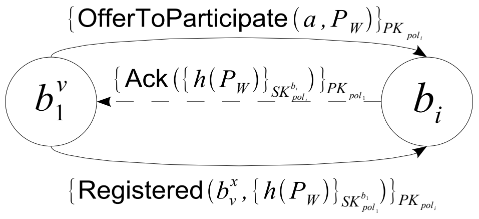

$\{\text{OfferToParticipate}(a, P_W)\}_{PK_{pol_i}}$
$\{\text{Ack}(\{h(P_W)\}_{SK^{b_i}_{pol_i}})\}_{PK_{pol_1}}$
b^v_1
b_i
$\{\text{Registered}(b^x_v, \{h(P_W)\}_{SK^{b_1}_{pol_1}})\}_{PK_{pol_i}}$

Fig. 13. Business partner registration when security mechanisms are used

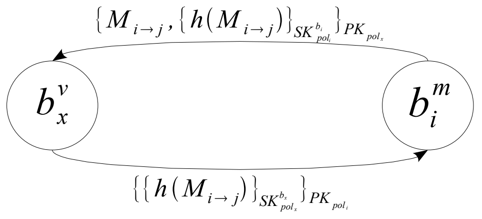

$\{M_{i \to j}, \{h(M_{i \to j})\}_{SK^{b_i}_{pol_i}}\}_{PK_{pol_x}}$
b^v_x
b^m_i
$\{\{h(M_{i \to j})\}_{SK^{b_x}_{pol_x}}\}_{PK_{pol_i}}$

Fig. 14. Workflow message backup when security mechanisms are used

6) Encryption, decryption or signature fault: an error occurred during the encryption, the decryption or the signature process of some data that is not due to an invalid signature or a key material issue. The "Encryption, decryption or signature fault" only refers to computational or accidental faults that may occur during the operation execution and we consider that these operations are retriable. This fault is therefore not forwarded to the coordination protocol.

These various types of faults are basically derived from the sequence of operations specified in section V-C and that is executed by all business partners involved in a distributed workflow instance. The four types of security faults that are forwarded to the coordination protocol, can be in some cases handled without having to declare the complete failure of a workflow instance if these faults occurred accidentally. The workflow execution can indeed recover using a simple backup of corrupted workflow messages since the operations that can raise these faults are executed prior to any workflow data processing. When however it is not possible to recover from them using basic fault handling mechanisms, the workflow arbitrator has to be contacted. Our goal towards designing appropriate security-fault handling mechanisms therefore consists in the enforcement of the following property:

(P_3) Should a security fault from which it is not possible to recover occur, the identity of the business partner that made an error or behaved maliciously can be traced back

The design of the corresponding security-fault handling mechanism that is presented next thus assumes that the policy private key distribution scheme implemented is group cryptography so that business partners' identities can be easily traced back using the mechanisms specified in this paper.

B. Business partner registration

During the course of the normal business partner registration phase that takes place within the transactional protocol execution, we add a security handshake wherein the business partner selected to execute a given vertex transmits to the critical zone initiator his signature with policy private key of the workflow policy. This handshake can be seen as a commitment that the business partner will behave correctly during the workflow instance. The business partner registration is depicted in figure 13. The critical zone initiator contacts a candidate business partner asking him whether he agrees to commit to execute the operation a of the workflow whose workflow policy is P_W. In case he accepts and this means that he trusts the business partner initiator of the critical zone who satisfies the policy pol_1, the candidate business partner acknowledges with a signature on the workflow policy with his policy private key $SK^{b_i}_{pol_i}$. Once the newly assigned business partner's coordinator for the transactional protocol execution is known, b^v_1 sends the information and acknowledges the registration with a signature on P_W with his policy private key. The following registration mechanism assumes as for the basic security mechanisms that the selected business partner trusts business partners satisfying the policy associated with the first vertex of a critical zone.

C. Workflow message backup process

The workflow data backup procedure implemented by the basic transactional protocol is also modified in order to handle the security faults we identified. In this case, we store the workflow message that has been issued by a business partner of type b^m_k to the next business partner involved in the workflow instance. The workflow message backup copy is still handled by the business partner b^v_x most recently executed. The workflow message backup procedure is depicted in figure 14. The business partner b^m_i is in charge of assessing the validity of the message that he sends to the business partner b^v_x most recently executed, this is why he signs the message he sends with his policy private key $SK^{b_i}_{pol_i}$. The business partner b^v_x acknowledges the receipt of the workflow message signing it with his policy private key $SK^{b_x}_{pol_x}$. Of course at each step of the procedure both business partners verify the validity of the signature provided by the other and may ask for a new transmission if they do not match.

D. Recovering from security-faults

Security faults which occur accidentally, such as these that result from transmission errors and the like, can be simply recovered using backup workflow messages that are stored by the business partners of type b^v_x and that should be valid. It is indeed the responsibility of the business partner that backed up the workflow message that should be retransmitted to ensure the validity of the latter. This statement is actually enforced by the signature provided by business partners during the message backup procedure. If backup workflow messages are valid, there exists a restoration point in the workflow execution such that the execution is still consistent from both transactional and security perspectives so that the worfklow execution can be restarted from that point if required. If the backup workflow message is however not valid, the recovery procedure fails and the workflow instance arbitrator is contacted to mediate the case as specified in the next section.

E. Handling security-faults when the recovery procedure fails

When security faults can not be recovered after several retransmissions of a backup workflow message, the mediation of the workflow instance arbitrator is required. The reasons why the recovery procedure fails are in fact not limited to malicious behaviors and our procedure to handle recovery failures does not depend on these various reasons as our primary goal is the anonymity revocation of the business partner that caused the fault. There are mainly four situations that require the mediation of the workflow arbitrator:

1) Failure to recover from a "Data decryption fault": based on the signatures and workflow messages sent by the two business partners that were involved in the backup procedure of the corrupted workflow message, the workflow arbitrator determines whether the business partner of type b_x^v that stored this message has caused the corruption or whether the other business partner backed up a message wherein a piece of data was not properly encrypted with the appropriate vertex public key. Based on the outcome of the mediation, the workflow arbitrator can either ask for a new generation of the corrupted workflow message with workflow data correctly encrypted or declare the workflow instance inconsistent and penalize the business partner who caused the fault.

2) Failure to recover from a "Vertex private key retrieval fault": based on the signatures and workflow messages sent by the two business partners that were involved in the backup procedure of the corrupted workflow message, the workflow arbitrator determines whether the business partner of type b_x^v that stored this message has caused the corruption, whether the other business partner backed up a message that was corrupted or whether the workflow initiator generated an onion O_d that was corrupted in the first place. Based on the outcome of the mediation, the workflow arbitrator can either ask for a retransmission of the corrupted workflow message should a valid copy of it be available, ask the workflow initiator to generate a new onion O_d corresponding to the current state of the workflow execution (i.e. the upper layer of this onion is associated with the vertex during the execution of which the fault was raised) or declare the workflow instance inconsistent and penalize the business partner that caused the fault.

3) Failure to recover from a "Data integrity fault": based on the signatures and workflow messages sent by the two business partners that were involved in the backup procedure of the corrupted workflow message, the workflow arbitrator determines whether the business partner of type b_x^v that stored this message has caused the corruption or whether the other business partner backed up a message that was corrupted in the first place. Based on the outcome of the mediation, the workflow arbitrator can either ask for a retransmission of the corrupted workflow message should a valid copy of it be available or declare the workflow instance inconsistent and penalize the business partner that caused the fault.

4) Failure to recover from a "Proof of execution fault": based on the signatures and workflow messages sent by the two business partners that were involved in the backup procedure of the corrupted workflow message, the workflow arbitrator determines whether the business partner of type b_x^v that stored this message has caused the corruption or whether the other business partner backed up a message that was corrupted in the first place. Based on the outcome of the mediation, the workflow arbitrator can either ask for a retransmission of the corrupted workflow message should a valid copy of it be available or declare the workflow instance inconsistent and penalize the business partner that caused the fault. In case the fault is the result of a signature missing in the onion O_p, the business partner that did not store it can be easily identified since the workflow initiator gathers all signatures prior to the workflow instantiation.

It should be noted that these situations wherein workflow message retransmission is not sufficient to recover are in fact highly unlikely to occur since the integration within the business partner registration process of a signature retrieval mechanism enabling anonymity revocation can be seen as a penalty for business partners that may cause security faults.

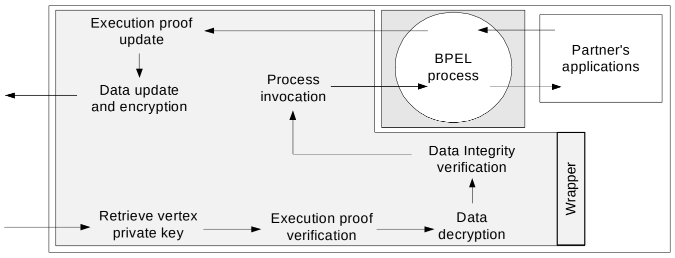

Execution proof update
Data update and encryption
Process invocation
BPEL process
Partner's applications
Data Integrity verification
Wrapper
Retrieve vertex private key
Execution proof verification
Data decryption

Fig. 15. Integration of the security mechanisms within a workflow system

VIII. Implementation

We developed a security library in order to implement the mechanisms presented in this paper. The implementation supports the following encryption algorithms:

- **IBE**: we use the Java Cryptography Extension (JCE) provider presented in [15] that is an implementation of the ID-based cryptosystem specified in [12],
- **RSA-ECB**: we use the Bouncy Castle [16] JCE provider that implements the RSA-ECB algorithm.

The RSA-ECB encryption algorithm is only used for demonstration purposes as the IBE implementation is resource-consuming ; the RSA-CBC scheme would be more suitable for deployment but is unfortunately not implemented by the JCE provider we have used. The security library has been integrated into the implementation of the pervasive workflow system introduced in section based on the sequence of operations specified in the sections V-B and V-C. The basic principles of the pervasive workflow engine implementation leveraging the security mechanisms we presented are depicted in figure 15. Due to the distributed nature of the pervasive workflow system, the pervasive workflow engine is deployed on each business partner involved in a workflow instance. Its implementation is composed of the following modules:

- **Engine wrapper**: this module implements the pervasive workflow business logic executed by a business partner as specified in section II.
- **BPEL engine**: the pervasive workflow implementation leverages the BPEL [17] workflow description language, a BPEL workflow engine is therefore used as an interpreter for pervasive workflow specifications.
- **Partners's applications**: the set of applications and services available on a business partner's device. They are contacted by the BPEL process.

The complete security mechanism execution is handled by the engine wrapper that forwards decrypted data only to the BPEL workflow engine. Once the results are sent back by the BPEL engine to the engine wrapper, the latter processes data and builds the workflow messages that are sent to the next business partners involved in the distributed workflow execution.

As specified in section VII the security mechanisms have been integrated into a coordination framework whose implementation principles are depicted in figure 16. The coordination stack is composed of the following components:

- **Transactional coordinator**: this component is supported by the critical workflow initiator. On the one hand it implements

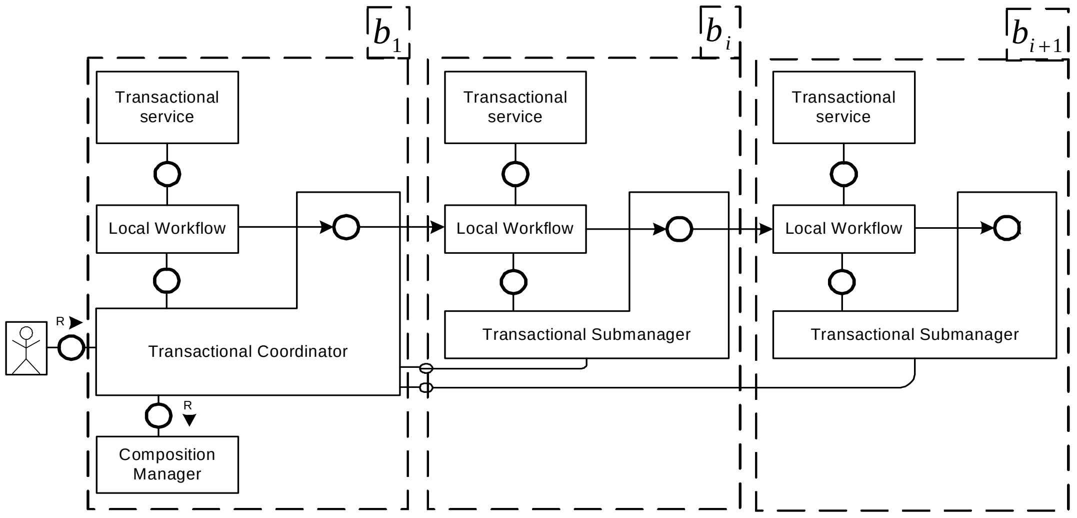


Fig. 16. Fault management framework

the business partner assignment procedure as part of the composition manager module and on the other hand it is in charge of assuring the coordinator role of the transactional protocol.

- **Transactional submanager**: this component is deployed on the other partners and is in charge of forwarding coordination messages from the local workflow engine to the appropriate subcoordinator or coordinator and conversely.

IX. Related work

Security of cross-organizational workflows in both centralized and decentralized settings has been an active research field over the past years [18], [19] mainly focusing on access control, separation of duty and conflict of interests issues. However, in the decentralized setting issues related to the integrity of workflow execution and workflow instance forging, which are presented in this paper have been left aside. This section discusses related work in the access control, separation of duty and conflict of interests research fields, before presenting related work in the area of onion encryption.

A. Separation of duty and conflict of interests

The management of conflict of interests within a workflow execution consists in enforcing that workflow data which are sensitive for a business partner can not be accessed by business partners that are competitors (in the marketing sense) of the latter within the workflow instance. In the centralized setting whereby workflow management and control tasks rely on a trusted coordinator, solutions can be found to manage workflow data accordingly, however in the decentralized setting wherein all workflow data are transferred between partners this is a challenging issue.

In [4] and [3] mechanisms are proposed for the management of conflict of interests [20] during the distributed execution of workflows. These pieces of work specify solutions in the design of access control policies to prevent business partners from accessing data that are not part of their classes of interest. These approaches do not address the issue of policy enforcement with respect to integrity of execution in fully decentralized workflow management systems yet they appear to complement our security mechanism design. The access control policy models suggested in [4] and [3] can indeed be used to augment our work especially in the specification of the sets J_i^r and J_i^w at workflow design time so that the mechanisms we designed to enforce access control on workflow data integrate solutions for the management of conflict of interests.

The separation of duty principle within a workflow execution consists in ensuring that two or more distinct business partners should be involved in the completion of a set of related workflow tasks [21]. As for the management of conflict of interests, existing solutions such as the one presented in [22] can be used to augment our work at workflow design time in the specification of more fine-grained workflow policies associated with vertices.

B. Access control within workflow management systems

The enforcement of access control policies within workflow management systems has been a quite active research field over the past years especially in the centralized setting, only few contributions do however tackle this issue in the decentralized setting.

Most approaches rely on the Role Based Access Control model [23], [24] to implement access control policies within the execution of a workflow [25]. In the centralized setting, many access control infrastructures have been as well proposed for business processes executed in the Service Oriented Architecture paradigm [26], [24], [27].

In [28], [5] centralized infrastructures are proposed which protect resources that should be accessed during the execution of a workflow based on credentials provided by business partners involved in the workflow instance. In [29] a similar approach based on XACML [30] is presented to enforce access control policies so that BPEL activities are executed by authorized users during the execution of BPEL processes. Despite solid contributions, these approaches do not however meet the requirements introduced in the decentralized setting as they rely on a centralized component to issue access control decisions. They are as a matter fact designed to protect local resources rather than workflow data that are transferred between business partners without a centralized point of coordination in charge of business partner assignment and access control policy enforcement.

Some pieces of work have also tackled security issues within distributed collaborative applications. In [31] a solution enforcing

RBAC policies is presented to protect the access to peers part of a peer-to-peer community. In this approach peers are able to make access control decisions autonomously without relying on an external policy decision point. Finally in [32] an access control model is also proposed that however do not include the notion of role and thus is not appropriate to meet the requirements we identified for distributed workflows.

C. Onion encryption techniques

Onion encryption techniques have been introduced in [6] and are widely used to enforce anonymity in network routing protocols [33] or mobile agents [34]. In the approach presented in this paper, we apply onion encryption techniques within a new application domain that is workflow-based applications and that introduces new requirements. We map onion structures with workflow execution patterns in order to build proofs of execution and enforce access control on workflow data. As a result, more complex business scenarios are supported by our solution than usual onion routing solutions. Furthermore, combined with policy encryption techniques, our solution provides a secure runtime environment for the execution of fully decentralized workflows supporting runtime assignment of business partners, a feature which had not been tackled so far. Existing solutions based on onion ring encryption developed in other application domains can not indeed be directly used to meet security requirements specified within decentralized workflow management systems.

X. Conclusion

We presented mechanisms towards meeting the security requirements raised by the execution of workflows in the decentralized setting. Our solution, capitalizing on onion encryption techniques and security policy models, protects the access to workflow data with respect to the pre-defined workflow execution plan and provides proofs of execution to business partners. In addition, those mechanisms combined with group cryptography provide business partners' identity traceability for sensitive workflow instances and can easily be integrated into the runtime specification of decentralized workflows. Our approach is suitable for any business scenarios in which business roles can be mapped to security policies that can be associated with key pairs. It can thus be easily integrated into existing security policy models such as the chinese wall [20] security model.

We also showed how these security mechanisms can be integrated into a transactional coordination framework. The goal of this approach is to make sure that detecting security faults does not necessarily mean the complete failure of the process execution. We presented the corresponding security-fault mechanisms relying on the workflow arbitrator to handle critical situations.

Finally, an implementation work based on Web services technologies and identity-based encryption techniques has been pursued as a proof of concept of our theoretical work.

References

[1] D. Barbara, S. Mehrotra, and M. Rusinkiewicz, "Incas: Managing dynamic workflows in distributed environments," *Journal of Database Management*, vol. 7, no. 1, 1996.

[2] F. Montagut and R. Molva, "Enabling pervasive execution of workflows," in *Proceedings of the 1st IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing, CollaborateCom*, 2005.

[3] V. Atluri, S. A. Chun, and P. Mazzoleni, "A chinese wall security model for decentralized workflow systems," in *CCS '01: Proceedings of the 8th ACM conference on Computer and Communications Security*, 2001, pp. 48–57.

[4] S. Chou, A. Liu, and C. Wu, "Preventing information leakage within workflows that execute among competing organizations," *J. Syst. Softw.*, vol. 75, no. 1-2, pp. 109–123, 2005.

[5] M. H. Kang, J. S. Park, and J. N. Froscher, "Access control mechanisms for inter-organizational workflow," in *SACMAT '01: Proceedings of the sixth ACM symposium on Access control models and technologies*, 2001, pp. 66–74.

[6] P. F. Syverson, D. M. Goldschlag, and M. G. Reed, "Anonymous connections and onion routing," in *IEEE Symposium on Security and Privacy*, USA, 1997, pp. 44–54.

[7] F. Montagut and R. Molva, "Towards transactional pervasive workflows," in *EDOC 2006, 10th IEEE International EDOC Conference "The Enterprise Computing Conference", 16-20 October 2006, Hong-Kong*, Oct 2006.

[8] "Open system interconnection- distributed transaction processing (osi-tp) model. iso is 100261."

[9] F. Montagut and R. Molva, "Augmenting Web services composition with transactional requirements," in *ICWS 2006, IEEE International Conference on Web Services, September 18-22, 2006, Chicago, USA*, Sep 2006.

[10] W. Bagga and R. Molva, "Policy-based cryptography and applications," in *FC' 2005, 9th International Conference on Financial Cryptography and Data Security, Roseau, The Commonwealth of Dominica*, Mar 2005.

[11] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik, "A practical and provably secure coalition-resistant group signature scheme," *Lecture Notes in Computer Science*, vol. 1880, pp. 255–271, 2000.

[12] D. Boneh and M. K. Franklin, "Identity-based encryption from the weil pairing." in *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, CA, USA*, 2001, pp. 213–229.

[13] K. Paterson, "Id-based signatures from pairings on elliptic curves," *Electronics Letters*, vol. 38, no. 18, pp. 1025–1026, 2002.

[14] F. Montagut and R. Molva, "Traceability and integrity of execution in distributed workflow management systems," in *ESORICS 2007, European Symposium On Research In Computer Security,Dresden, Germany, September 24 - 26, 2007*, Sep 2007.

[15] L. Owens, A. Duffy, and T. Dowling, "An identity based encryption system," in *PPPJ '04: Proceedings of the 3rd international symposium on Principles and practice of programming in Java*. Trinity College Dublin, 2004, pp. 154–159.

[16] "Bouncy Castle - http://www.bouncycastle.org/," 2007.

[17] "Business process execution language for web sevices," http://www.ibm.com/developerworks/library/ws-bpel/.

[18] C. Li and C. Pahl, "Security in the web services framework," in *ISICT '03: Proceedings of the 1st international symposium on Information and communication technologies*. Trinity College Dublin, 2003, pp. 481–486.

[19] E. Bertino, E. Ferrari, and V. Atluri, "The specification and enforcement of authorization constraints in workflow management systems," *ACM Trans. Inf. Syst. Secur.*, vol. 2, no. 1, pp. 65–104, 1999.

[20] D. F. C. Brewer and M. J. Nash, "The chinese wall security policy," in *IEEE Symposium on Security and Privacy*, 1989, pp. 206 –214.

[21] R. T. Simon and M. E. Zurko, "Separation of duty in role-based environments," in *IEEE Computer Security Foundations Workshop*, 1997, pp. 183–194.

[22] P. C. K. Hung, "From conflict of interest to separation of duties in ws-policy for web services matchmaking process," in *HICSS '04: Proceedings of the Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04) - Track 3*. Washington, DC, USA: IEEE Computer Society, 2004, p. 30066.2.

[23] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-based access control models," *IEEE Computer*, vol. 29, no. 2, pp. 38–47, 1996.

[24] R. Wonohoesodo and Z. Tari, "A role based access control for web services," in *SCC '04: Proceedings of the 2004 IEEE International Conference on Services Computing*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 49–56.

[25] G.-J. Ahn, R. Sandhu, M. H. Kang, and J. S. Park, "Injecting RBAC to secure a web-based workflow system," in *Proceedings of 5th ACM Workshop on Role-Based Access Control*, 2000, pp. 1–10.

[26] W. T. Tsai, X. Liu, and Y. Chen, "Distributed policy specification and enforcement in service-oriented business systems," in *ICEBE '05:*

Proceedings of the IEEE International Conference on e-Business Engineering. Washington, DC, USA: IEEE Computer Society, 2005, pp. 10–17.

[27] V. Atluri and W. kuang Huang, "An authorization model for workflows," in *Proceedings of the Fourth European Symposium on Research in Computer Security*, 1996, pp. 44–64.

[28] H. Koshutanski and F. Massacci, "An access control framework for business processes for web services," in *XMLSEC '03: Proceedings of the 2003 ACM workshop on XML security*. New York, NY, USA: ACM Press, 2003, pp. 15–24.

[29] E. Bertino, J. Crampton, and F. Paci, "Access control and authorization constraints for ws-bpel," in *ICWS '06: Proceedings of the IEEE International Conference on Web Services (ICWS'06)*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 275–284.

[30] "OASIS eXtensible Access Control Markup Language (XACML)," 2005.

[31] J. S. Park and J. Hwang, "Role-based access control for collaborative enterprise in peer-to-peer computing environments," in *SACMAT '03: Proceedings of the eighth ACM symposium on Access control models and technologies*. New York, NY, USA: ACM Press, 2003, pp. 93–99.

[32] P. C. K. Hung and K. Karlapalem, "A secure workflow model," in *ACSW Frontiers '03: Proceedings of the Australasian information security workshop conference on ACSW frontiers*, 2003, pp. 33–41.

[33] J. Kong and X. Hong, "Anodr: anonymous on demand routing with untraceable routes for mobile ad-hoc networks," in *MobiHoc '03: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, 2003, pp. 291–302.

[34] L. Korba, R. Song, and G. Yee, "Anonymous communications for mobile agents," in *MATA '02: Proceedings of the 4th International Workshop on Mobile Agents for Telecommunication Applications*. London, UK: Springer-Verlag, 2002, pp. 171–181.

Frederic Montagut is a PhD candidate at SAP Research in Mougins, France. Frederic obtained his engineering diploma from Telecom INT, France and the Diploma of Advanced Studies (M.Sc.) in Network and Distributed Systems from Nice University, France in 2004. He joined SAP Research in October 2004 working under the supervision of Pr Refik MOLVA, Eurecom Institute. His research interests range from distributed workflow systems, workflow coordination to workflow security.

Refik Molva is a full professor and the head of the Computer Communications Department at Eurecom Institute in Sophia Antipolis, France. His current research interests are in security protocols for self-organizing systems and privacy. He has been responsible for research projects on multicast and mobile network security, anonymity and intrusion detection. Beside security, he worked on distributed multimedia applications over high speed networks and on network interconnection. Prior to joining Eurecom, he worked as a Research Staff Member in the Zurich Research Laboratory of IBM where he was one of the key designers of the KryptoKnight system. He also worked as a security consultant in the IBM Consulting Group in 1997. Refik Molva has a Ph.D. in Computer Science from the Paul Sabatier University in Toulouse (1986) and a B.Sc. in Computer Science (1981) from Joseph Fourier University, Grenoble, France.