# Traceability and Integrity of Execution in Distributed Workflow Management Systems *

Frederic Montagut[1] and Refik Molva[2]

[1] SAP Labs France, 805 Avenue du Docteur Maurice Donat,
Font de l'Orme, 06250 Mougins, France
[2] Institut Eurecom, 2229 Route des Cretes, 06904 Sophia-Antipolis, France
frederic.montagut@sap.com, refik.molva@eurecom.fr

**Abstract.** The execution of business processes in the decentralized setting raises security requirements due to the lack of a dedicated infrastructure in charge of management and control tasks. 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. In this paper, we suggest new security mechanisms capitalizing on onion encryption and group encryption techniques in order to assure the integrity of the distributed execution of workflows and to manage traceability with respect to sensitive workflow instances. We carry out an in depth analysis of the security properties offered by these mechanisms. Our solution can easily be integrated into distributed workflow management systems as its design is strongly coupled with the runtime specification of decentralized workflows.

**Keywords:** Integrity of execution, Traceability, Decentralized workflows.

## 1 Introduction

State of the art business processes may require a decentralized support of execution [1],[2],[3] because of their dynamicity or unusual execution environments. The flexibility of a distributed workflow enactment system on the other hand comes at the expense of security due to the lack a dedicated infrastructure to perform the management and control tasks during the execution of a business process. 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 workflow instance becomes an issue without a trusted centralized coordination mechanism selecting workflow participants. As opposed to centralized workflow management systems, the distributed execution of workflows indeed raises new security requirements due to the lack of a dedicated coordinator. 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

* 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.

J. Biskup and J. Lopez (Eds.): ESORICS 2007, LNCS 4734, pp. 251–266, 2007.

been focusing on issues related to the management of rights in business partner assignment or detecting conflicts of interest [4],[5],[6], basic security issues related to the security of the overall workflow execution such as integrity and evidence of execution have not yet been addressed. We already tackled some of these problems in a previous work [7] yet the solution we proposed did not take into account the management of security policies and business partners' trustworthiness.

In this paper, we present new mechanisms supporting the secure execution of workflows in the decentralized setting. These mechanisms capitalize on onion encryption techniques [8] 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 suggested mechanisms can easily be integrated into the runtime specification of decentralized workflow management systems as illustrated in this paper using the pervasive workflow model specified in [3]. The remainder of the paper is organized as follows. Section 2 and 3 outline the pervasive workflow model and the associated security requirements, respectively. In section 4 our solution is specified while in section 5 the runtime specification of the secure distributed workflow execution is presented. Section 6 presents the security analysis of the proposed mechanisms. Finally section 7 discusses related work and section 8 presents the conclusion.

## 2 Workflow Model

The workflow management system used to support our approach was designed in [3]. This model 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 devices 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

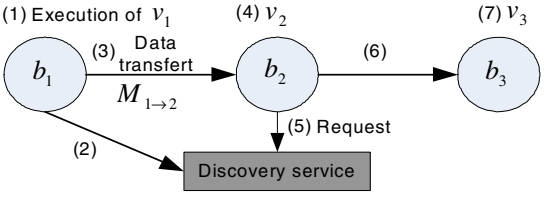


**Fig. 1.** Pervasive workflow runtime

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 $\delta$ is the set of execution dependencies between those vertices. We note $(M_{i\rightarrow j_p})_{p\in[1,z_i]}$ 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.

## 3 Security Requirements

### 3.1 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.

### 3.2 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.

### 3.3 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 $D$
- 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

# 4 The Solution

## 4.1 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 6. The policy private key $SK_{pol_i}$ can indeed be distributed by different means amongst which we distinguish three main types:

- Key sharing: a policy $pol_i$ is associated with a single private policy 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 $pol_i$. In this case, the partners satisfying $pol_i$ share the same policy private key.
- Policy-based cryptography: a policy $pol_i$ is expressed in a conjunctive-disjunctive form specifying the combinations of credentials a principal is required to satisfy to be compliant with the policy. A cryptographic scheme [9] is used to map credentials to keys denoted credential keys that can be combined to encrypt, decrypt and sign messages based on a given policy. Some trusted authorities are in charge of distributing credential keys to requesters when the latter satisfies some assertions (e.g. (jobtittle=director)∧(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.
- Group cryptography: a policy $pol_i$ is mapped to a group structure in which a group manager distributes different private policy keys to group members satisfying $pol_i$.

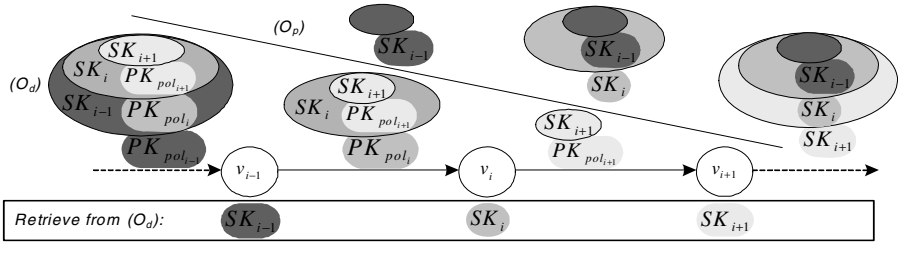


**Fig. 2.** Key management

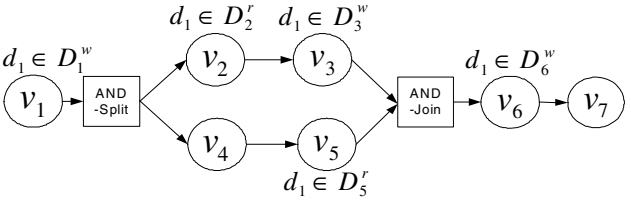


**Fig. 3.** Workflow example

A single encryption key 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 [10]. We note $GM_{pol_i}$ the group manager of the group whose members satisfy $pol_i$. The management of policy key pair 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 2 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. The encryption of a message $m \in \mathcal{M}$ with a key $K \in \mathcal{K}$ is noted $\{m\}_K$ and $h_1, h_2$ denote one-way hash functions.

### 4.2 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 4 called data block to protect workflow data accordingly. Each data block consists of two fields: the actual data $d_k$ and a signature $sign_a(d_k) = \{h_1(d_k)\}_{SK_a}$. 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 $v_a$. $H_k^a = \left\{\{h_1(\{B_k^a\}_{PK_l})\}_{SK_a} | l \in R_k^a\right\}$ where $R_k^a$ is the set defined as follows. $R_k^a = \{l \in [1,n] | (d_k \in D_l^r)$ and ($v_l$ is executed after $v_a$) and ($v_l$ is not executed after $v_{p_{(a,l,k)}}$)$\}$ 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 3 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 $R_1^3 = \{6\}$.

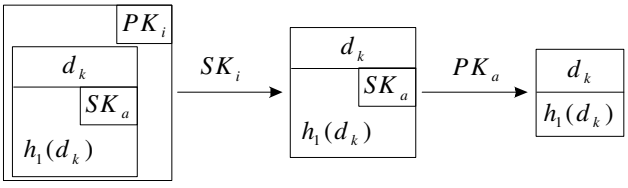


**Fig. 4.** Access to workflow data

When the business partner $b_i$ receives the data block $B_k^a$ encrypted with $PK_i$ (i.e. he is granted read access on $d_k$), he 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 $v_a$. Further, if $b_i$ is granted write access on $d_k$, he can update the value of $d_k$ and compute $sign_i(d_k)$ yielding a new data block $B_k^i$ and a new set $H_k^i$. If on the contrary $b_i$ receives $B_k^a$ encrypted with $PK_m$ (in this case $v_m$ is executed after $v_i$), $b_i$ can verify the integrity of $\{B_k^a\}_{PK_m}$ by matching $h_1(\{B_k^a\}_{PK_m})$ 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.

### 4.3 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. The complete building process is specified in [7], the main results on the distribution of vertex private keys with respect to various workflow patterns are thus only reminded in this section.

**Definition 1.** *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 2.** *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 $a_j \subseteq b_k$.*

**SEQUENCE workflow pattern.** An onion structure assuring the distribution of vertex private keys is sequentially peeled off by partners. Considering a sequence of $n$ vertices $(v_i)_{i\in[1,n]}$ $b_1$ assigned to $v_1$ initiates the workflow with the onion structure $O$.

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

For $i \in [2, n-1]$ the partner $b_i$ assigned to $v_i$ receives $\{l_{n-i+1}(O)\}_{PK_{pol_i}}$, reads $l_{n-i+1}(O)$ using $SK_{pol_i}$ to retrieve $SK_i$ and sends $\{l_{n-i}(O)\}_{PK_{pol_{i+1}}}$ to $b_{i+1}$.

**AND-SPLIT workflow pattern.** $n$ business partners are concurrently contacted by a single partner, $n$ different onions are therefore concurrently sent.

**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. When $n-1$ branches merge into a vertex $v_a$, $v_a$ is executed if and only if $n-1$ messages are received. The vertex private key $SK_a$ is thus divided into $n-1$ parts contained in $n-1$ onions to be received by $v_a$. Besides, in order to avoid redundancy, the onion structure associated with the sequel of the workflow execution right after $v_a$ is only included in one of the onions received by $v_a$.

**OR-SPLIT workflow pattern.** This is an exclusive choice, a single onion is sent depending on the result of the OR-SPLIT condition. The onions associated with the branches that can be executed are thus wrapped beforehand.

**OR-JOIN workflow pattern.** Since there is a single workflow initiator, the OR-JOIN is preceded in the workflow by an OR-SPLIT pattern. A single branch is executed depending on the choice made at the previous OR-SPLIT in the workflow, thus a single onion is sent to the vertex into which the branches merge.

**Complete key distribution scheme.** The onion $O_d$ enabling the vertex private keys distribution during the execution of the workflow depicted in figure 3 is defined as follows.

$$O_d = \{\{SK_1, \underbrace{\{SK_2, \{SK_3, \{SK_{6_1}, \overbrace{\{SK_7\}_{PK_{pol_7}}}^{\text{Sequel after } v_6}\}_{PK_{pol_6}}\}_{PK_{pol_3}}\}_{PK_{pol_2}}}_{\text{First AND-SPLIT branch}},$$
$$\underbrace{\{SK_4, \{SK_5, \{SK_{6_2}\}_{PK_{pol_6}}\}_{PK_{pol_5}}\}_{PK_{pol_4}}}_{\text{Second AND-SPLIT branch}}\}_{PK_{pol_1}}\}$$

The onions associated with the two branches forming the AND-SPLIT pattern are wrapped by the layer corresponding to $v_1$. Only the first AND-SPLIT branch includes the sequel of the workflow after $v_6$.

### 4.4 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 4.1, 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 $W_{ar}$ who 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 $O_{p_1} = \left\{\{h_1(P_W)\}_{SK_{pol_1}}\right\}$ where $P_W$ is called workflow policy and is defined as follows.

**Definition 3.** *The workflow specification* $S_W$ *denotes the set* $S_W = \{W, (J_i^r, J_i^w, pol_i)_{i\in[1,n]}, h_1\}$ *where* $J_i^r = \{k \in [1,j] | d_k \in D_i^r\}$ *and* $J_i^w = \{k \in [1,j] | d_k \in D_i^w\}$ *(*$J_i^r$ *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* $P_W = S_W \cup \{W_{iid}, W_{ar}, h_2\} \cup \{PK_i | i \in [1,n]\}$. $P_W$ *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 key and $b_1$ is in charge of contacting a trusted third party, sending it $(h_1(P_W), P_W)$ 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, $b_i$ encrypts $\{O_{p_{i-1}}, \{h_1(P_W)\}_{SK_{pol_i}}\}$ with $SK_i$ instead. Considering a set $(v_i)_{[1,n]}$ of vertices executed in sequence and assuming that traceability is needed we get:

$$
\begin{aligned}
O_{p_1} &= \left\{\{h_1(P_W)\}_{SK_{pol_1}}\right\} \\
O_{p_2} &= \left\{\{O_{p_1}, \{h_1(P_W)\}_{SK_{pol_2}}\}_{SK_2}\right\} \\
O_{p_i} &= \left\{\{O_{p_{i-1}}, \{h_1(P_W)\}_{SK_{pol_i}}\}_{SK_i}\right\} \text{ for } i \in [3,n]
\end{aligned}
$$

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. First, there is no specific rule for OR-SPLIT and OR-JOIN patterns. Second, 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: $O_{p_n} = \left\{\{O_{p_1}, O_{p_2}, .., O_{p_{n-1}}, \{h_1(P_W)\}_{SK_{pol_n}}\}_{SK_n}\right\}$. Considering the example depicted in figure 3 and assuming traceability is not required, at the end of the workflow execution the onion $O_p$ is defined as follows.

$$
O_p = \{\underbrace{\{\{\{\{\{h_1(P_W)\}_{SK_{pol_1}}\}_{SK_2}\}_{SK_3}}_{\text{First AND-SPLIT branch}}, \underbrace{\{\{\{h_1(P_W)\}_{SK_{pol_1}}\}_{SK_4}\}_{SK_5}}_{\text{Second AND-SPLIT branch}}\}_{SK_6}\}_{SK_7}\}
$$

$\{h_1(P_W)\}_{SK_{pol_1}}$ 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 $\{h_1(P_W)\}_{SK_{pol_1}}$ twice.

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 $\{h_1(P_W)\}_{SK_{pol_1}}$ 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. $b_i$ decrypts $\{h_1(P_W)\}_{SK_{pol_p}}$ for all $p \in [1,k_i]$. If $b_i$

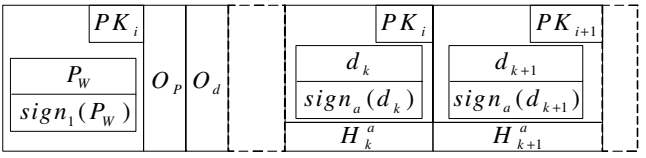


**Fig. 5.** Workflow message structure

detects that a signature is missing he contacts $W_{ar}$ 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.

### 4.5 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 [11] 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:

$$\begin{cases} PK_i = h_1(W_{iid} \oplus S_W \oplus v_i) \\ SK_i = s \times h_2(PK_i) \end{cases}$$

where $s \in \mathbb{Z}_q^*$ for a prime $q$. $s$ is called master key and is held by the vertex private key generator [11] who is in our case the workflow initiator. The signature scheme proposed in [12] 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 $O_p$.

### 4.6 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 5.

Workflow data $(d_k)_{k\in[1,j]}$ are all transported between business partners and 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.

$P_W$ is required to retrieve vertex and policy public keys and specifies the workflow execution plan.

The two onion structures $O_d$ and $O_p$ are also included in the message.

Upon receipt of the message depicted in figure 5 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 finally he can process workflow data.

## 5 Secure Execution of Decentralized Workflows

In this section we specify how the mechanisms presented so far 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.

### 5.1 Execution Process Overview

Integrating security mechanisms to enforce the security requirements of the decentralized workflow execution 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 the following $N_k^i$ denotes the set defined by $N_k^i = \{l \in [1,n] | d_k \in D_l^r$ and $v_l$ is executed right after $v_i\}$. Consider the example of figure 3: $d_1$ is accessed during the execution of the vertices $v_1$, $v_2$ and $v_5$ thus $N_1^1 = \{2,5\}$.

### 5.2 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 $(M_{1 \rightarrow j_p})_{p \in [1,z_1]}$. The workflow initiation consists of the following steps.

1. Workflow policy specification: generate $(PK_i, SK_i)_{i \in [1,n]}$ and assign $W_{ar}$
2. Initialization of the onion $O_d$
3. Data block initialization: compute $\forall k \in [1,j]$ $sign_1(d_k)$

4. Data block encryption: $\forall k \in [1,j]$ determine $N_k^1$ and compute $\forall k \in [1,j], \forall l \in N_k^1$ $\{B_k^1\}_{PK_l}$
5. Data block hash sets: $\forall k \in [1,j]$ determine $R_k^1$ and compute $\forall k \in [1,j], \forall l \in R_k^1$ $\{h_1(\{B_k^1\}_{PK_l})\}_{SK_1}$
6. Initialization of the onion $O_p$: compute $O_{p_1}$
7. Message generation based on $W$ and $(N_k^1)_{k\in[1,j]}$

The steps one and two are presented in sections 4.5 and 4.3, respectively. The workflow messages are generated with respect to the specification defined in figure 5 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.

### 5.3 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 $v_i$.

1. Retrieve $SK_i$ from $O_d$
2. Data block decryption with $SK_i$ based on $J_i^r$
3. Execution proof verification: peel off the onion $O_p$
4. Data integrity check based on $W$ and $P_W$
5. Vertex execution
6. Data block update: compute $\forall k \in J_i^w$ $sign_i(d_k)$ and update $d_k$ content
7. Data block encryption: $\forall k \in J_i^r$ determine $N_k^i$ and compute $\forall k \in J_i^r, \forall l \in N_k^i$ $\{B_k^i\}_{PK_l}$
8. Data block hash sets: $\forall k \in J_i^w$ determine $R_k^i$ and compute $\forall k \in J_i^w, \forall l \in R_k^i$ $\{h_1(\{B_k^i\}_{PK_l})\}_{SK_i}$
9. Onion $O_p$ update: compute $O_{p_i}$
10. 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.

## 6 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

### 6.1 Inherent Security Properties

**Proposition 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 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 $J_i^w$.*

*Proof.* This property is ensured by the onion $O_d$ which assures distribution of the vertex keys used for accessing workflow data based on the workflow execution plan.

Assuming that a workflow initiator builds $O_d$ based on the methodology specified in 4.3 and under the policy key security assumption, we claim that it is not feasible for an adversary $\mathcal{A}$ to extract the vertex private key $SK_i$ from $O_d$ if $\mathcal{A}$ does not know the set of policy private keys $(SK_{pol_{i_k}})_{k\in[1,l]}$ associated with the set of vertices $(v_{i_k})_{k\in[1,l]}$ executed prior to $v_i$ in $W$. This is true as the structure of $O_d$ is mapped to $W$.

**Proposition 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 $pol_1$.*

*Proof.* This property is enforced by the onion $O_p$ whose building process is based on the workflow structure and vertex private keys. As stated in the previous claim, vertex private keys can only be retrieved by business partners knowing some policy private keys. We also claim that an adversary that does not verify a policy can not forge a workflow instance, i.e. that the adversary can not produce a workflow message pertaining to a valid workflow instance.

Assuming that a workflow initiator builds $O_p$ based on the methodology specified in 4.4 and under the policy key security assumption, we claim that the onion structure $O_p$ is unforgeable. The unforgeability property relies on two further properties:

1. a genuine onion structure $O_p$ built during a previous instance of a workflow can not be replayed ;
2. an onion structure $O_p$ can not be built by an adversary that is not trusted by business partners.

The first property is enforced by the fact that an onion structure $O_p$ properly built is bound to a specific workflow policy $P_W$ and thus can not be reused during an attempt to execute a malicious workflow instance. The second property is straightforward under the policy key security assumption as the policy-based signature scheme achieves signature unforgeability. Thus an adversary can not produce a valid onion $O_{p_1} = \left\{\{h_1(P_W)\}_{SK_{pol_1}}\right\}$.

**Proposition 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*

*Proof.* The first property is ensured as the set of workflow data blocks that should be present in a workflow message is specified in $P_W$, the workflow message formatting has thus to be compliant with the workflow specification. The second property is assured by the fact that an adversary can not modify a given data block without providing a valid signature on this data block. This property relies on the unforgeability of the signature scheme used in the data block and hash set specifications.

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.

### 6.2 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 claims 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, proposition 3 is thus verified.

**Proposition 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 pre-defined 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.*

*Proof.* Anonymity revocation is here a means to force business partners to behave properly during the execution of a workflow. If any malicious business partner is involved, he will not store his signature and we claim that the workflow instance will no longer be a valid one. The mechanism we proposed for anonymity revocation is as we mentioned optimistic and four scenarios can actually occur:

- A business partner detects that a signature is missing during the course of the workflow execution
- Each business partner stored his signature
- A set of business partners did not store their signature while some other partners did not declare the missing signatures to the workflow arbitrator
- A set of business partners assigned to vertices contiguously executed till the end of the workflow did not store their signature

In the first case, the workflow instance will be declared inconsistent by the workflow arbitrator. In the second case, trustworthy business partners have been involved in the workflow and their identity can be easily traced back by the workflow arbitrator. In the third case which is in fact highly unlikely to occur, the business partners who have not declared the missing signatures become responsible in place of the business partners who cheated. In the last case, nobody can be held responsible as apparently a group of untrustworthy business partners was involved in a fraud attempt and the workflow instance is declared inconsistent.

### 6.3 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.

## 7 Related Work

Security of cross-organizational workflows in both centralized and decentralized settings has been an active research field over the past years mainly focusing on access control, separation of duty and conflict of interests [13],[14],[6] issues. However, in the decentralized setting issues related to the integrity of workflow execution and workflow instance forging, which are tackled in our paper have been left aside. In [5],[4] mechanisms are proposed for the management of conflicts of interest [15] 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. Nonetheless, the access control policy models suggested in [5],[4] can be used to augment our work especially in the specification of the sets $J_i^r$ and $J_i^w$ at workflow design time.

Onion encryption techniques have been introduced in [8] and are widely used to enforce anonymity in network routing protocols [16] or mobile agents [17]. In our approach, 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.

Finally, 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 chinese wall [15] security model.

## 8 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 future work will focus on the integration of these security mechanisms into a transactional framework that we developed for the pervasive workflow model.

## References

1. Barbara, D., Mehrotra, S., Rusinkiewicz, M.: Incas: Managing dynamic workflows in distributed environments. Journal of Database Management 7(1) (1996)
2. Cichocki, A., Rusinkiewicz, M.: Providing transactional properties for migrating workflows. Mob. Netw. Appl. 9(5), 473–480 (2004)

3. Montagut, F., Molva, R.: Enabling pervasive execution of workflows. In: Proceedings of the 1st IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing, CollaborateCom, IEEE Computer Society Press, Los Alamitos (2005)
4. Atluri, V., Chun, S.A., Mazzoleni, P.: A chinese wall security model for decentralized workflow systems. In: CCS '01: Proceedings of the 8th ACM conference on Computer and Communications Security, pp. 48–57. ACM Press, New York (2001)
5. Chou, S.C., Liu, A.F., Wu, C.J.: Preventing information leakage within workflows that execute among competing organizations. J. Syst. Softw. 75(1-2), 109–123 (2005)
6. Kang, M.H., Park, J.S., Froscher, J.N.: Access control mechanisms for inter-organizational workflow. In: SACMAT '01: Proceedings of the sixth ACM symposium on Access control models and technologies, pp. 66–74. ACM Press, New York (2001)
7. Montagut, F., Molva, R.: Enforcing integrity of execution in distributed workflow management systems. In: SCC 2007. 2007 International Conference on Services Computing, Salt Lake City, USA, July 9-13, 2007 (2007)
8. Syverson, P.F., Goldschlag, D.M., Reed, M.G.: Anonymous connections and onion routing. In: IEEE Symposium on Security and Privacy, USA, pp. 44–54. IEEE Computer Society Press, Los Alamitos (1997)
9. Bagga, W., Molva, R.: Policy-based cryptography and applications. In: Patrick, A.S., Yung, M. (eds.) FC 2005. LNCS, vol. 3570, Springer, Heidelberg (2005)
10. Ateniese, G., Camenisch, J., Joye, M., Tsudik, G.: A practical and provably secure coalition-resistant group signature scheme. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 255–271. Springer, Heidelberg (2000)
11. Boneh, D., Franklin, M.K.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
12. Paterson, K.: Id-based signatures from pairings on elliptic curves. Electronics Letters 38(18), 1025–1026 (2002)
13. Bertino, E., Ferrari, E., Atluri, V.: The specification and enforcement of authorization constraints in workflow management systems. ACM Trans. Inf. Syst. Secur. 2(1), 65–104 (1999)
14. Hung, P.C.K., Karlapalem, K.: A secure workflow model. In: ACSW Frontiers '03. Proceedings of the Australasian information security workshop conference on ACSW frontiers, pp. 33–41 (2003)
15. Brewer, D.F.C., Nash, M.J.: The chinese wall security policy. In: IEEE Symposium on Security and Privacy, pp. 206–214. IEEE Computer Society Press, Los Alamitos (1989)
16. Kong, J., Hong, X.: 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, pp. 291–302. ACM Press, New York (2003)
17. Korba, L., Song, R., Yee, G.: Anonymous communications for mobile agents. In: Karmouch, A., Magedanz, T., Delgado, J. (eds.) MATA 2002. LNCS, vol. 2521, pp. 171–181. Springer, Heidelberg (2002)
18. Nanda, M.G., Karnik, N.: Synchronization analysis for decentralizing composite web services. In: Matsui, M., Zuccherato, R.J. (eds.) SAC 2003. LNCS, vol. 3006, pp. 407–414. Springer, Heidelberg (2004)
19. Tripathi, A.R., Ahmed, T., Kumar, R.: Specification of secure distributed collaboration systems. In: ISADS '03. Proceedings of the The Sixth International Symposium on Autonomous Decentralized Systems, p. 149 (2003)