Secure Composition of Cascaded Web Services

Basit Shafiq
Lahore Univ. of Management Sciences
Lahore, Pakistan
basit@lums.edu.pk

Soon Chun
City University of New York
Staten Island, NY, USA
Soon.Chun@csi.cuny.edu

Jaideep Vaidya, Nazia Badar, Nabil Adam
Rutgers University
Newark, NJ, USA
{jsvaidya, nbadar, adam}@cimic.rutgers.edu

***Abstract*—A business process can be developed as a composition of Web services provided by different service providers. These service providers may have their own policies and constraints for service provisioning and collaboration. In this paper, we focus on secure composition of services, specifically from the perspective of service enactment. Service enactment requires finding an execution plan for the service composition that conforms to the requirements and constraints of the service requester and all service providers. However, due to privacy and security concerns, participants may selectively expose their Web service operations and process details. We propose an approach for service enactment that does not require the participants to reveal their internal operations and constraints and that can still result in an execution plan which satisfies the requirements and constraints of all participants. The proposed approach uses Finite State Machines (FSM) to model component Web service operations, their interdependencies, as well security and access control policy constraints. Model checking is used to generate an appropriate Web service execution plan in an incremental manner. Commutative encryption based techniques are used to preserve privacy and security.**

***Index Terms*—web service composition, security, privacy.**

I. Introduction

Cloud computing infrastructure and semantic Web technologies have together created unprecedented opportunities for composing large-scale business processes and workflow-based applications that span multiple organizational domains. One of primary hurdles towards wide-spread adoption of Web services in a collaborative environment is security and policy disclosure from the perspective of both service providers and service requesters. For example, sensitive information that can be attributed to individual service providers or service requesters should not be disclosed [1], [2]. Similarly, local business process details of one entity should not be disclosed to other entities. Given such diverse security and privacy requirements, Web service composition in the cloud environment poses major challenges. In this paper, we focus on secure composition of Web service in publicly available clouds as well as in enterprise clouds. We consider Web service composition from *service enactment* perspective [3]. *Service enactment* deals with finding an execution plan, that conforms to the overall requirements and constraints of the composite service specified by the requester, and satisfies the security and access control policies of individual Web services.

Figure 1 depicts the main aspects of distributed composition of Web services that involves orchestration of many atomic or composite Web services to complete a multi-step business process in shared services cloud. We refer to such Web services as component Web services and their composition as a Web service process (WSP). Component Web services can be modeled as a finite state machine (FSM), though due to privacy and security concerns, only a partial view of the FSM may be visible to the requester. For example in Figure 1, the dark-filled circles corresponds to visible Web service operations while the white circles are the invisible internal operations. These internal operations may in turn invoke Web services of other service providers. This is consistent with the notion of partner links in BPEL, where only certain operations of partners are visible to the organization. However, an operation of the partner link may itself be a BPEL process which is invisible to the organization invoking such operation.

The WSP specifications consist of the control-flow and information-flow dependencies among the component Web services and a set of constraints. Constraints are used to define global requirements over the WSP [3]. These requirements can be classified as: i) aggregate service quality constraints (e.g., overall cost of executing the WSP); and ii) Event constraints (e.g., event a cannot occur between events b and c). As depicted in Figure 1, a component Web service can be composed of multiple Web services and therefore can have its own aggregate and event constraints that need to be considered for WSP enactment. Composing cascaded Web services securely in such a distributed environment is a challenging task, that nevertheless occurs in many real life situations, as discussed below:

Illustrative Example: Consider State health department countermeasure inventory management process for tracking inventory of critical medical countermeasures (e.g., antivirals) in different localities. In case of shortage of countermeasure supplies in any local jurisdiction, the State health department may mobilize supplies from its stockpile or from the private sector pharmaceutical supply chain to that local jurisdiction. As depicted in Figure 1, this process involves interaction with the inventory management and resource planning systems of various stakeholders including state and local health agencies and emergency response organizations, hospital, clinics, and private sector pharmaceutical supply chain entities, including retail pharmacies, distributors and manufactures. The individual Web services of the stakeholders' system may also be complex business processes as depicted in Figure 2. These services may be hosted on different enterprise clouds and need to be linked together to compose the global process. For

COLLABORATECOM 2012, October 14-17, Pittsburgh, United States

DOI 10.4108/icst.collaboratecom.2012.250522

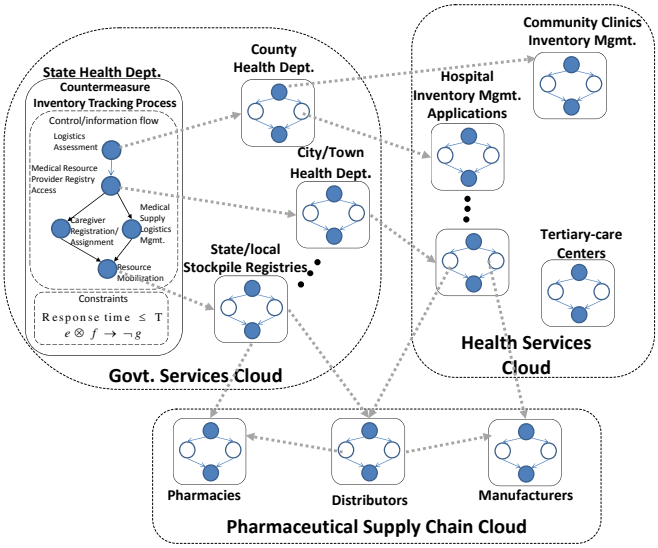


Fig. 1. Example of Web service process composition in shared services cloud environment.

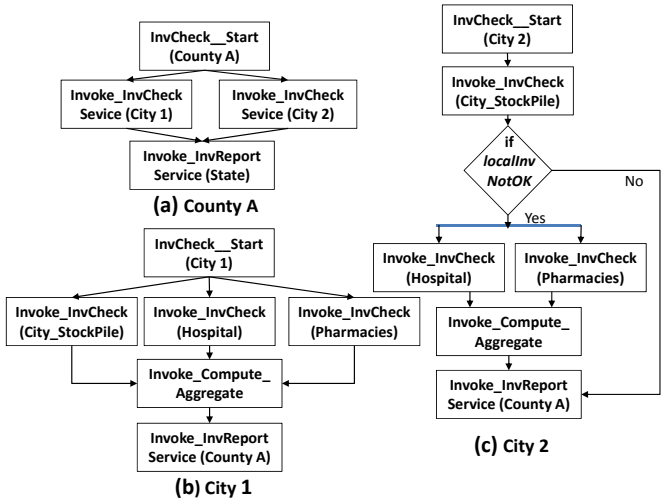


Fig. 2. Countermeasure inventory checking and reporting processes of:(a) County A; (b) City 1; and (c) City 2

example, the inventory management and supply acquisition services of public health agencies are hosted on Govt. services cloud; hospitals and clinic host their inventory tracking and ordering services on health services cloud; and private sector pharmaceutical entities provide their services on the pharmaceutical supply chain cloud as shown in Figure 1.

The entire WSP may have several event constraints: i) If the inventory in the stockpile of the local jurisdiction (city/town; county) is not at satisfactory level, only then the inventory of the private sector entities (hospitals/clinics, pharmacies) is to be checked. ii) If the inventory level of a hospital has already been reported for one locality then the inventory level of that hospital cannot be reported later for another locality. For the first constraint, the temporal dependencies between events (**Event 1:** reporting of the local jurisdiction inventory below satisfactory level. **Event 2:** checking hospitals, pharmacies inventories in that locality) can be represented in the event algebra formalism of Concurrent Transaction Logic [3] as: $\triangledown localInvNotOK \otimes (\triangledown Invoke\_InvCheck(Hospitals) \wedge \triangledown Invoke\_InvCheck(Pharmacies))$; where, $\triangledown e \otimes \triangledown f$ implies that event e must occur before event f.

Given the information disclosure and privacy concerns related to Web service operations and event message attribution, enactment and execution of a WSP become a challenging issue when considering the following scenarios:

S1. The events specified in a WSP constraint may occur as internal operations of component Web services that are not exported globally. Moreover, different events in a WSP constraint may occur at different component Web services. For example in Figure 2, $invokeInvCheck(City\_stockPile)$, $invokeInvCheck(Hospital)$, and $invoke\_invCheck(pharmacies)$ are the internal operations of the City 1 and City 2 Web services and are not visible to County A that has invoked the inventory reporting services of these cities.

S2. The events specified in a WSP constraint have ordering and temporal dependencies that may span multiple component Web services.

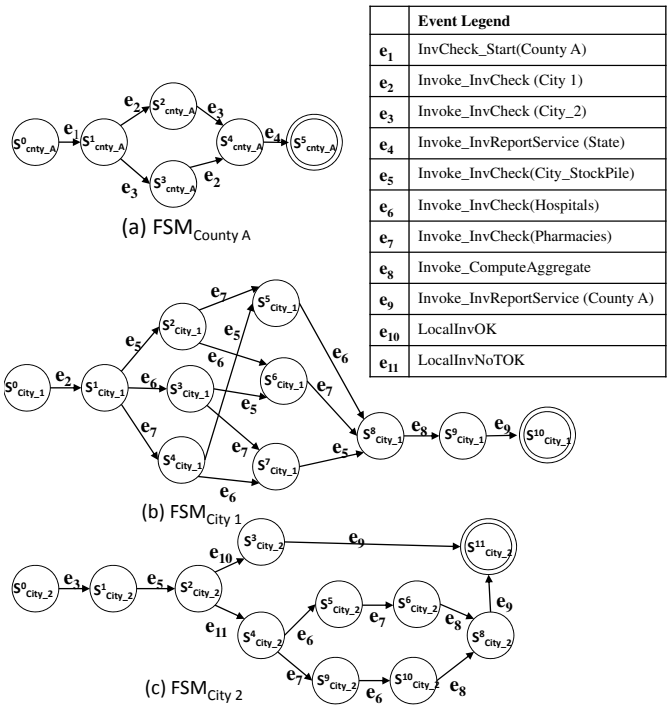
S3. The WSP constraints are not disclosed globally.

Such scenarios are likely to occur in the countermeasure inventory tracking example of Figure 1. For example, a local jurisdiction may not reveal which hospitals/pharmacies' inventory management services it has used for reporting the countermeasure stock information in its region (scenario S1). Similarly, there is a temporal dependency between events included in constraint 1, e.g., stockpile of local health agency need to be checked first before checking the inventory of private sector entities (scenario S2). The challenge here is that there is no single party who is aware of all the operations of component Web services that need to be enacted for WSP composition. Therefore, no party can verify whether the interactions among the component services conform to a given constraint associated with the WSP.

Contribution: In this paper, we develop an automata-theoretic model checking approach with encryption strategy for enactment of a cascaded Web service process in a secure and privacy preserving manner. The proposed approach employs Finite State Machine (FSM) for modeling component Web service operations, their interdependencies, the global constraints of requesters, as well as security policies of service providers. The proposed approach generates the execution plan in an incremental manner. In terms of security and privacy, the proposed approach prevents disclosure of policies and internal operations of service providers against service requesters at different levels in the composition hierarchy.

II. Preliminaries

We now present the formalism and notations used to represent Web services, execution plans, and WSP constraints.



	Event Legend
e_1	InvCheck_Start(County A)
e_2	Invoke_InvCheck (City 1)
e_3	Invoke_InvCheck (City_2)
e_4	Invoke_InvReportService (State)
e_5	Invoke_InvCheck(City_StockPile)
e_6	Invoke_InvCheck(Hospitals)
e_7	Invoke_InvCheck(Pharmacies)
e_8	Invoke_ComputeAggregate
e_9	Invoke_InvReportService (County A)
e_{10}	LocalInvOK
e_{11}	LocalInvNoTOK

Fig. 3. FSM representation of the processes of:(a) County A; (b) City 1; and (c) City 2

A. Web Service Behavior Modeling

We model the behavior of each component Web service sp as a finite state machine denoted by $\mathcal{FSM}_{sp} = \{\Sigma_{sp}, S_{sp}, s^0_{sp}, X, \delta_{sp}, F_{sp}, B, status\}$. Here, Σ_{sp} denotes the Web service operations or events; S_{sp} is a set of states; $s^0_{sp} \in S_{sp}$ is a single initial state which is also called entry state or initiation point of the Web service; $X = \{x_1, ..., x_n\}$ is a finite set of integer variables used to represent data value and data flow dependencies; $\delta_{sp} : S_{sp} \times \Sigma_{sp} \rightarrow S_{sp}$ is the transition function; $F_{sp} \subseteq S_{sp}$ is the set of final states which marks completion of the component web service sp; and B is a set of boolean variables, where each $b \in B$ provides status information of the state; the function $status$ assigns to each state $s \in S_{sp}$, the set of boolean variables that are true in s.

Figure 3 shows the FSM representation of the Web service processes of County A, City 1, and City 2 depicted in Figure 2.

B. Execution Plan

An execution plan of a given Web service is basically a sequence of operations that need to be invoked for successful completion of the given Web service [3]. Since, a Web service is characterized by the set of operations it can execute and the interdependencies/constraints among these operations, it can be modeled by finite state machines (FSM) [4]. In this respect, an execution plan can be viewed as a state-transition path (st-path) $s_1, e_1, s_2, ...e_{n-1}s_n$, where s_i $(1 \leq i \leq n)$ is a state and e_i is an operation, alternatively event that changes the state from s_i to s_{i+1}.

For a WSP, the execution plan and the corresponding st-path span multiple component Web services. Alternatively, the execution plan for a given WSP is an interleaving of the st-paths of the component Web services organized based on the control/information flow requirements of the WSP and the component Web services.

Projection of an execution plan. We define a projection operation $PROJ$ over an execution plan for a given Web service FSM. Given an execution plan and a Web service FSM, this operation projects only those states and events in the execution plan that are included in the given Web service FSM. For example, given the execution plan of the WSP (π_{WSP}) and the component Web service $\mathcal{FSM}$.

$PROJ_{\mathcal{FSM}}(\pi_{WSP}) = \pi'$, where π' is the longest common subsequence of π_{WSP} of the form $s_1, e_1, s_2, \ldots e_{n-1}, s_n$ such that $\mathcal{FSM}$ includes the transition (s_i, e_i, s_{i+1}), $1 \leq i \leq n$.

C. Constraints

We also model constraints as a finite state machine. Constraints can be composed to form more complex constraints.

A constraint c is characterized by a finite state machine $\mathcal{FSM}_c = \{\Sigma_c, S_c, s^0_c, X, \delta_c, err_c, A_c, B, status, b_c\}$. Where, Σ_c denotes the events over which the constraint is defined; S_c is a set of states; $s^0_c \in S_{sp}$ is a single initial state; $X = \{x_1, ..., x_n\}$ is a finite set of integer variables used to represent data value or data flow dependencies; $\delta_c : S_c \times \Sigma_c \rightarrow S_c$ is the transition function; $err_c \in S_c$ is an error state; $A_c \subseteq S_c$ is a set of acceptable states; B is a set of boolean variables, where each $b \in B$ provides status information of the state; the function $status$ assigns to each state $s \in S_{sp}$, the set of boolean variables that are true in s, and $b_c \in B$ is a boolean variable called *constraint status variable* such that $b_c = 1$ in the acceptable states only, and $b_c = 0$ in all other states.

Similar to concurrent transaction logic (CTR) [3], we consider primitive constraints, immediate serial constraints, serial constraints and allow composition of these constraints.

1) **Primitive constraint** (depicted in Figure 4(a)) is used to represent if an event $e \in \Sigma_c$ must occur or must not occur in the execution plan. In the CTR formalism, this is represented by $\triangledown e$ (must occur – positive primitive constraints) and $\neg\triangledown e$ (must not occur – negative primitive constraints). An example of the positive primitive constraint with reference to the inventory tracking process of Figure 1 is checking the local inventory of each jurisdiction for every countermeasure x, i.e., $\triangledown localInvCheck(?x)$ must occur in the execution plan.
2) **Immediate serial constraint** (depicted in Figure 4(b)) is used to represent if events $e_1, ..., e_n \in \Sigma_c$ must occur next to each other with no other events in-between in the execution plan. In CTR, this is represented by $\triangledown \odot (e_1 \otimes ... \otimes e_n)$.
3) **Serial constraint** is used to define a serial order between two or more constraints that are either positive primitive or positive immediate serial constraints. Figure 4(c) depicts the serial constraint composition $c_1 \otimes c_3$ with

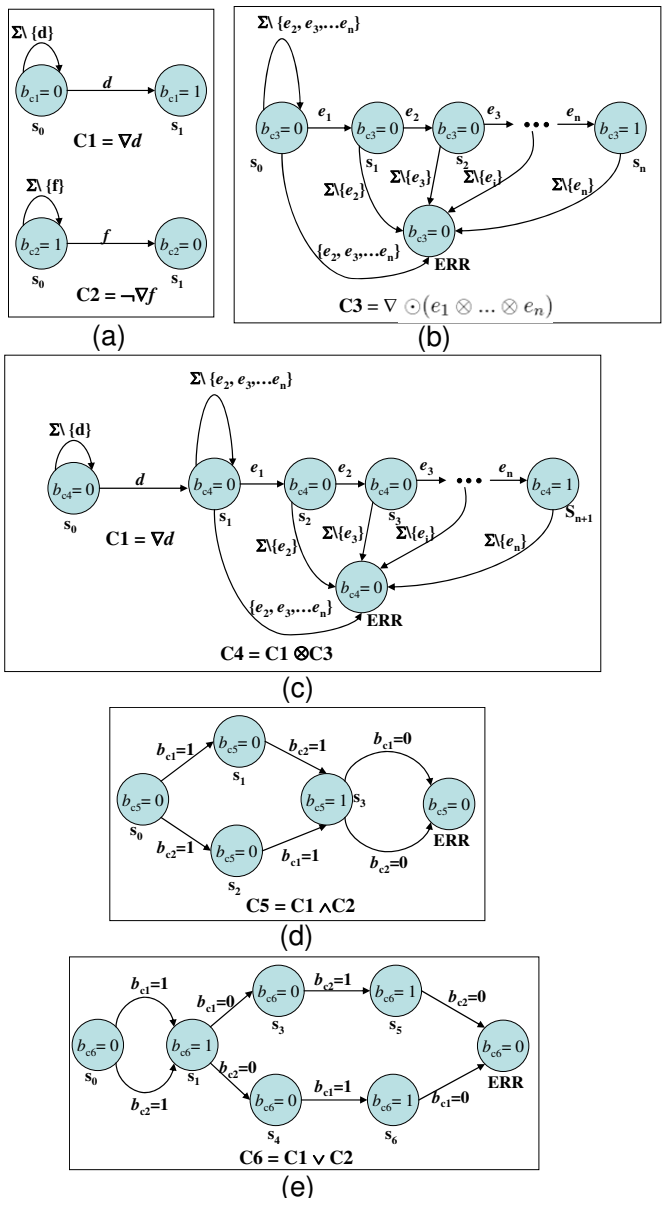
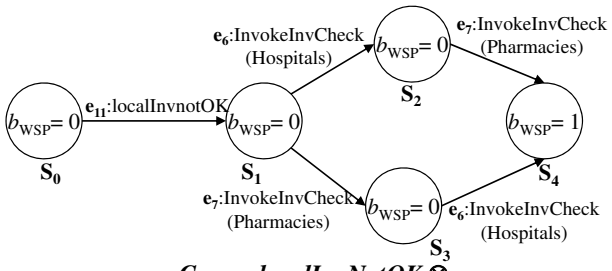


Fig. 4. FSM representation of constraints

$c_1 = \nabla d$ (positive primitive constraints) and $c_3 = \nabla \odot (e_1 \otimes ... \otimes e_n)$ (positive immediate serial constraint).

4) **Complex constraints** can be composed using the conjunction ($\wedge$) or disjunction ($\vee$) operators. In the FSM representation, we use the constraint satisfaction variables of underlying constraints to model complex constraints. Figure 4 (d) and (e) depicts constraints $C_1 \wedge C_2$ and $C_1 \vee C_2$.

The above constraints or constraints composed from the above constraints can be converted to the normal form[3]: $\vee_i(\wedge_j serialconstr_{i,j})$ where each $serialconstr_{i,j}$ is either a primitive constraint or a serial constraint composed of two positive primitive constraints. The set formed by such constraints is a closed set. Since the FSM representation allows



C_{WSP} = ***localInvNotOK*** ⊗ (***InvokeInvCheck(Hositals)*** ∧ ***InvokeInvCheck(Pharmacies)***)

Fig. 5. FSM representation of the constraint $C_{WSP} = \nabla localInvNotOK \otimes (\nabla Invoke_InvCheck(Hospitals) \wedge \nabla Invoke_InvCheck(Pharmacies))$

modeling of complex constraints from primitive constraints, immediate serial constraints, and serial constraints using the $\vee$ and $\wedge$ operators, any constraint that can be represented in CTR can also be represented as an FSM.

Figure 5 shows the FSM representation of the constraint $C_{WSP} = \nabla localInvNotOK \otimes (\nabla Invoke_InvCheck(Hospitals) \wedge \nabla Invoke_InvCheck(Pharmacies))$.

III. Problem Statement

Definition 1 (Service Enactment): Let $\mathcal{FSM}_{WSP}$ and C_{WSP} denote the FSM representation of the WSP and the global constraints associated with WSP. Let $\mathcal{SP}$ denote the set of component Web services that need to be composed for WSP execution. For any $sp \in \mathcal{SP}$, let $\mathcal{FSM}_{sp}$ and C_{sp} denote the FSM representation of the component Web service and the constraints respectively. Service enactment can be formally stated as determining an interleaving of the st-paths π of each component Web service such that: $\pi_{WSP} : interleave(\{\pi_{sp} | sp \in \mathcal{SP}\}) \models (\mathcal{FSM}_{WSP} || C_{WSP}) || _{sp \in \mathcal{SP}} (\mathcal{FSM}_{sp} || C_{sp})$. Here, $A||B$ represents the FSM composed from automata A and B.

The above expression implies that π_{WSP} is a trace of the FSM composed from $\mathcal{FSM}_{WSP}$, C_{WSP}, and the FSMs of all the component Web services and constraints. We consider a hierarchical structure among the different entities involved in service composition as depicted in Figure 6. This hierarchy is established based on the roles of these entities as a service requester or as a service provider at different levels of the composition. At the lowest level (level = 0), the original WSP is the service requester and the component Web services that have direct interaction with the WSP as service providers. At the next level these service providers become service requesters and the component Web services invoked by them are the service providers and so on.

Given the service enactment definition above and the Web service composition hierarchy of Figure 6, the goal is to compute π_{WSP} in a secure and privacy preserving way. The security requirement is entailed in service enactment definition in terms of satisfaction of all the constraints and process specifications. Privacy entails the following three requirements.

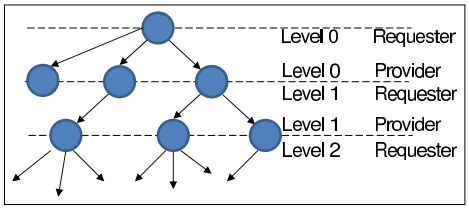


Fig. 6. Web service composition hierarchy

P-I At any level i, the service requester sr_i can only see the combined FSM composed from the component web services FSM and the constraint FSM of all the service providers at level i. However, the individual component Web service FSM and constraint FSM of any service provider at level i or higher is not visible to sr_i.

P-II Even the combined FSM composed from the component web services FSM and the constraint FSM of the service providers at level i is not visible to the service requester sr_i.

P-III The constraints C_{sr_i} are local to the service requester sr_i and are not disclosed to any service provider at level i or higher.

IV. Secure Service Enactment

The proposed approach achieves service enactment and generates a WSP execution plan along the service composition hierarchy (Figure 6) in an incremental and bottom-up manner in accordance with the requester's specification and all providers' security policies. After generation of the WSP execution plan, the execution plan for individual component Web services is extracted and disclosed to the service providers in a top-down manner. In terms of privacy, the approach can be configured to satisfy the meaningful combinations of above privacy requirements P-I, P-II, and P-III.

A. Basic Idea

Service enactment in a cascaded Web service environment involves two major tasks: i) generation of the overall WSP execution plan; and ii) Extraction of the execution plans for individual Web services from the overall WSP execution plan. The second task ensures that the execution plan of any given individual Web service is consistent with the overall execution plan generated at level 0. Since individual Web service may have multiple execution plans, it is critical to select a plan consistent with the overall WSP execution.

Generation of WSP Execution Plan. The basic idea for generating WSP execution plan is simple and can be summarized in the following steps.

1) Assuming that the leaf nodes of the service composition hierarchy tree (Figure 6) is at level n, each service provider sp at level n sends a composition of its component Web service FSM and constraints FSM $(\mathcal{FSM}_{sp}||C_{sp})$ to the service requester at level n that has invoked its service.
2) The service requester (sr_i) at level i $(i \leq n)$ upon receiving the FSMs from the service providers at level i verifies if the composition of all the FSMs satisfies its process specification and constraints. In case the privacy requirement P-III is not considered, i.e., sr_i knows the process specification and constraints of all service requesters at level $j < i$, then these are also included for constraint verification.
3) If the constraint verification is succeeded, sr_i, sends the composition of all the FSMs (including its own component Web service FSM and constraint FSM and the FSMs received from the service providers at level i) to the service requester at level $i-1$. This process continues, until the constraint verification is performed at level 0. Any trace of the combined FSM at level 0 that leads to a final state corresponds to the WSP execution plan that satisfies the specification and constraints of the WSP and all service providers.

This incremental and bottom-up strategy for constraint verification and service enactment ensures that at any given level a service requester only sees the combined FSM of the service providers at the same level. In other words, a service requester at level i does not know who is providing component Web service at level $i+1$ and what portion of the FSM at level i comes from the service providers at level $i+1$. However at level i, the requester can link component Web service FSM to the service providers at level i.

Extraction and Disclosure of Execution Plans for Individual Web Services. The execution plan for individual component Web services is extracted from the overall WSP execution plan and is disclosed to the service providers of the corresponding services in a top down manner along the service composition hierarchy as summarized in the following steps.

1) The original service requester at level 0, sr_0, after computing the execution plan π_{WSP} from the combined FSM sends π_{WSP} to its immediate service providers (who are also service requesters at level 1).
2) When a service requester sr_1 at level 1 receives the overall execution plan π_{WSP}, it extracts the execution plan specific to its Web service, i.e., π_{sr_1} from π_{WSP} by projecting only those states and events that are included in the FSM composed by sr_1 from the FSMs of its service providers. For this extraction, the $PROJ$ operation described in Section II is used.
   After computing π_{sr_1}, sr_1 send its execution plan downward to its immediate service providers for extraction of their execution plans. This process continues, until the execution plan is computed/extracted for all the service providers in the service composition hierarchy.

B. Proposed Approach.

To address the privacy requirements related to disclosure of policies and internal operations, an encryption based strategy can be used that enables constraint verification using model checking over an encrypted set of component Web service FSMs. This requires that all the component Web service FSMs of service providers as well as the constraint FSMs of service requester be encrypted with the same key. This

can be achieved using commutative encryption*, whereby the component Web service FSM of each service provider at level i is commutatively encrypted by all other service providers at the same level using their respective encryption keys. Since service providers may not be known to each other, therefore the encryption of the component Web service and FSM can be coordinated through the service requester (sr_i). Accordingly, each service provider can send its encrypted FSM to the requester which can iteratively route it to other providers for commutative encryption. The service requester (sr_i) also send its own component Web service FSM and constraint FSM $\mathcal{FSM}_{sr_i}||C_{sr_i}$ for commutative encryption by all its service providers at level i. These commutatively encrypted FSMs can be combined by sr_i to generate a composite encrypted FSM, $E_1..E_n(\mathcal{FSM}_{sr_i}||C_{sr_i})||_{sp\in\mathcal{SP}_i}E_1..E_n(\mathcal{FSM}_{sp}||C_{sp})$, where $\mathcal{SP}_i$ denotes the set of service providers whose component Web services are invoked by sr_i. Accordingly the composite FSM can be checked for existence of the trace that lead to the completion state of the Web service process of the service requester sr_i and satisfy all constraints of the service providers and service requester at level i. This corresponds to performing service enactment (checking for satisfaction of the expression in Definition 1) at level i. The resulting trace π_{sr_i} will be in encrypted form.

However, this simple encryption strategy does not prevent the service requester to learn the operations or events in the component Web services of its service providers. For example service requester, sr_i, can correlate the commutatively encrypted value of certain event in its constraints FSM C_{sr_i} with the commutatively encrypted values in the components Web service FSM of a given service provider and learn that such event is included in the component Web service of the given service provider. The main reason for this problem is that at any given level both composition of FSMs and constraint verification is performed by the requester. Therefore, the commutatively encrypted FSM of each service provider is known to the requester. Another issue is the communication overhead associated with the commutative encryption of each component Web service FSM by all service providers.

The proposed approach utilizes the hierarchical structure of the cascaded WS environment to separate out the task of coordinating encryption of component Web service FSMs and verification of the encrypted composite FSM for constraint satisfaction. Additionally, this relaxes the requirement of having each component Web service FSM encrypted by all service providers. Specifically, at any given level i the service requester sr_i receives the encrypted component Web service FSMs from each of its service providers. Each FSM is encrypted only with the encryption key of the corresponding service provider. These encrypted FSMs are used for generation of the encrypted composite FSM which is verified for constraint satisfaction one level up, i.e., at level $(i-1)$. For this, it is assumed that the decryption key of each service provider at level i (where, $i \geq 1$) is known to the service requester sr_{i-1} at level $i-1$. However, the proposed approach ensures that such disclosure of the decryption key does not enable sr_{i-1} to learn the operations or events in the component Web services of the respective service providers.

*An encryption algorithm is commutative if the order of encryption does not matter. Thus, for any two encryption keys E_1 and E_2, and any message m, $E_1(E_2(m)) = E_2(E_1(m))$. The same property applies to decryption as well - thus to decrypt a message encrypted by two keys, it is sufficient to decrypt it one key at a time.

Protocol 1 Executed by service requester sr_i at level i

Require: $\mathcal{SP}_i$ a set of service providers at level i whose component Web services are invoked by sr_i.

Require: constraint FSMs of all service requesters above level i in the composition hierarchy, i.e., $C_{WSP}, C_{sr_1}, ..C_{sr_{i-1}}$ (Not required when considering privacy requirement P-III)

1: **Generation of WSP Execution Plan**
2: sr_i creates commutative encryption and decryption key set (E_{sr_i}, D_{sr_i}).
3: $CFSM \leftarrow \emptyset$
4: **for** each $sp \in \mathcal{SP}_i$ **do**
5: receive $f \leftarrow E_{sp}(FSM_{sp})$ from sp
6: $f \leftarrow E_{sr_i}(f)$
7: $CFSM \leftarrow CFSM \cup \{(f, sp)\}$
8: **end for**
9: **if** P-III is considered **then**
10: send $CFSM, E_{sr_i}(FSM_{sr_i})$ and $E_{sr_i}(C_{sr_i})$ to the service requester sr_{i-1} at level $i-1$
11: **else**
12: send $CFSM, E_{sr_i}(FSM_{sr_i})$ and $E_{sr_i}(C_{WSP}||C_{sr_1}..||C_{sr_i})$ to sr_{i-1}
13: **end if**
14: **Extraction of Execution Plan for individual web services of service providers**
15: extract π_{sr_i} from $\pi_{sr_{i-1}}$ by invoking steps 6-10 of Protocol 2 if $(i \geq 1)$.
16: **for** each $(f, sp) \in CFSM$ **do**
17: send $E_{sr_i}(\pi_{sr_i})$ and f to sp. {Note that $f = E_{sr_i}E_{sp}(FSM_{sp})$}
18: **end for**
19: **for** each $sp \in \mathcal{SP}_i$ **do**
20: receive $E_{sr_i}E_{sp}(\pi_{sp})$ from sp and apply the decryption key D_{sr_i} to generate $E_{sp}(\pi_{sp})$
21: send $E_{sp}(\pi_{sp})$ to sp
22: **end for**

1) Generation of WSP Execution Plan: The proposed approach for privacy preserving generation of WSP execution plan employs three protocols. Protocol 1 (Lines 1-13) is executed by the service requester at level i, Protocol 2 (Lines 1-4) is executed by each service provider at level i, and Protocol 3 is executed by the service requester at level $i-1$. The key interactions are as follows:

Step 1. Each service provider sp_i at level i $(i \geq 1)$ creates a set of commutative encryption and decryption keys and sends its decryption key to the service requester sr_{i-1} at level $i-1$. It then encrypts its component Web service FSM with its encryption key, and sends it to the requester sr_i at level i as

Protocol 2 Executed by each service provider $sp \in \mathcal{SP}_i$ at level i

Require: service requester sr_i and sr_{i-1} at level i and $i-1$ respectively

Generation of WSP Execution Plan
sp creates commutative encryption and decryption key set (E_{sp}, D_{sp}).
sp sends D_{sp} to sr_{i-1}
sp sends $E_{sp}(FSM_{sp})$ to sr_i
Extraction of sp's Execution Plan
receive $E_{sr_i}(\pi_{sr_i})$ and $E_{sr_i}E_{sp}(FSM_{sp})$ from sr_i
Apply the encryption key E_{sp} on sr_i's encrypted execution plan to generate $E_{sr_i}E_{sp}(\pi_{sr_i})$
$E_{sr_i}E_{sp}(\pi_{sp}) \leftarrow PROJ_{E_{sr_i}E_{sp}(FSM_{sp})}(E_{sr_i}E_{sp}(\pi_{sr_i}))$
send $E_{sr_i}E_{sp}(\pi_{sp})$ to sr_i for decryption and receive $E_{sp}(\pi_{sp})$
$\pi_{sp} \leftarrow D_{sp}E_{sp}(\pi_{sp})$

Protocol 3 Executed by service requester sr_{i-1} at level $i-1$ for constraint verification

Require: service requester (sr_i) and $(\mathcal{SP}_i)$ a set of service providers at level i whose component Web services are invoked by sr_i

receive D_{sp} from each $sp \in \mathcal{SP}_i$
receive $CFSM, E_{sr_i}(FSM_{sr_i})$, and $E_{sr_i}(C_{WSP}||C_{sr_1}||..C_{sr_i})$ from sr_i
$DFSM \leftarrow \emptyset$
for each $(f, sp) \in CFSM$ **do**
  $f \leftarrow D_{sp}(f)$
  $DFSM \leftarrow DFSM \cup \{f\}$
end for
compute $F \leftarrow E_{sr_i}(FSM_{sr_i})||E_{sr_i}(C_{sr_i})||E_{sr_i}(C_{WSP})||,..,E_{sr_i}(C_{sr_{i-1}})||_{f \in DFSM}(f)$
if exists a trace π such that $\pi \models F$ **then**
  send **constraint verified** message to sr_i
else
  send **Error** message to sr_i
end if

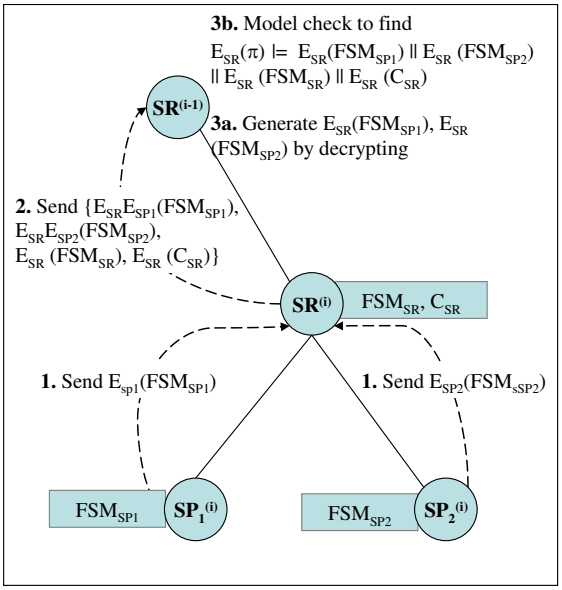


Fig. 7. Generation of execution plan and constraint verification at level i.

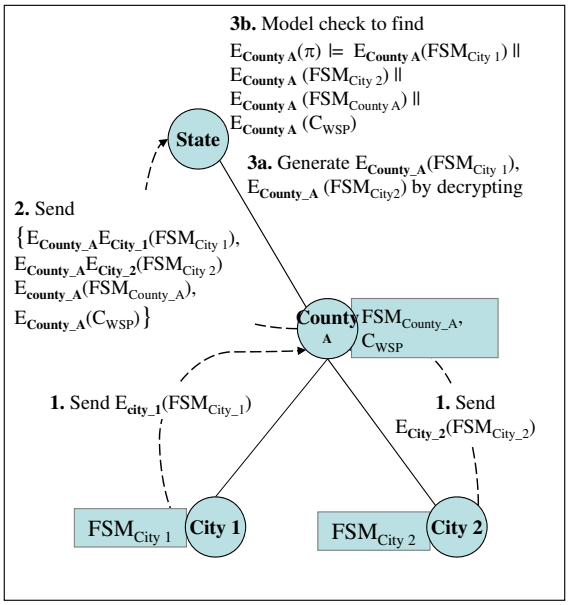


Fig. 8. Generation of execution plan and constraint verification for the counter measure inventory tracking example.

illustrated in Figure 7.

With reference to the running example, Figure 8 shows the execution plan generation and constraint verification for the counter measure inventory tracking process. In this figure, *City 1* and *City 2* are service providers at level 1, and *County A* is the service requester at level 1. At level 0, *State* is the service requester. Both *City 1* and *City 2* send their decryption keys to the *State*. Moreover, each city encrypts is component Web service FSM with its encryption key and send it to the *County A*. The component Web service FSMs of *City 1* and *City 2* are depicted in Figure 3(b) and (c), respectively.

Step 2. Upon receiving the encrypted FSM from a service provider, sr_i further encrypts it with its key E_{sr_i}. After applying its encryption key on all the component Web service FSMs sr_i sends all the commutatively encrypted component Web service FSMs of the service providers to the service requester sr_{i-1} at level $i-1$ as illustrated in Figure 7. In addition, sr_i encrypts its constraint FSM C_{sr_i} and constraints FSMs of service requesters at higher levels in the composition hierarchy, in case such constraints are visible to sr_i (i.e., privacy requirement P-III is not considered). These constraints FSMs encrypted with sr_i's encryption key are then sent to service requester sr_{i-1} at level $i-1$. Note that sr_{i-1} cannot correlate any event in the component Web service FSMs of the service providers and the encrypted constraint FSM of service requesters, as all constraint FSMs of service requesters

($C_{sr_j}, j \leq i$) are encrypted with the encryption key of sr_i only, whereas the component Web service and constraint FSM of service providers are encrypted with the encryption keys of all the service providers as well as by sr_i.

In the context of the running example, *County A* receives the encrypted FSM ($E_{City\_1}(FSM_{City\_1})$) from *City 1* and the encrypted FSM ($E_{City\_2}(FSM_{City\_2})$) from *City 2* as depicted in Figure 8. *County A* further encrypts these FSMS with its encryption key E_{County_A}. In addition, *County A* encrypts its component Web service FSM (FSM_{County_A}, shown in Figure 3(a)) and the WSP constraint FSM (C_{WSP}, shown in Figure 5) with its encryption key E_{County_A}. Here we assume that C_{WSP} which is the constraint defined by the *State* is visible to *County A*. After applying its encryption key, *County A* sends all the encrypted FSMs to its service requester, i.e., *State* as depicted in Figure 8. For reason discussed above, *State* cannot learn which of the events in C_{WSP} occur at *City 1* or *City 2*.

Step 3. sr_{i-1} receives the following set of encrypted FSM from sr_i: i) component Web service and constraint FSMs of service providers (FSM_{sp}); ii) component Web service FSM and constraint FSM of the service requester sr_i ($FSM_{sr_i}||C_{sr_i}$); iii) and if privacy requirement P-III is not considered, the encrypted constraint FSM of all service requesters above sr_i in the composition hierarchy. sr_{i-1} applies the decryption key of the respective service provider at level i to the corresponding component Web service FSM. Even after application of the decryption keys, such FSMs still remain encrypted with the encryption key of sr_i as the decryption key of sr_i is not known to sr_{i-1}. sr_{i-1} then generates a composite FSM by combining all the FSMs and check for existence of a trace in the composite FSM that leads to a service completion state without causing any of the constraint FSM to go into error state. The existence of such trace implies satisfaction of all the constraints and service requester specifications.

Figure 7 illustrates the process of generating the execution plan and verifying constraints at level i with two service providers $SP_1^{(i)}$, $SP_2^{(i)}$, a service requester $SR^{(i)}$ at level i and service requester $SR^{(i-1)}$ at level $i-1$.

With reference to the running example, the role of sr_{i-1} is played by *State* as shown in Figure 8. The *state* receives the set of encrypted FSMs from *County A*. This set of FSMs includes: i) FSM_{City_1} (shown in Figure 3(b)) encrypted with E_{City_1} and E_{County_A}; ii) FSM_{City_2} (shown in Figure 3(c))encrypted with E_{City_2} and E_{County_A}; iii) FSM_{County_A} (shown in Figure 3(a)) encrypted with E_{County_A}; and iv) FSM of C_{WSP} (shown in Figure 5) encrypted with E_{County_A}. The *state* has the decryption keys for *City 1* and *City 2*. It applies the respective decryption key on the received FSM of *City 1* and *City 2*. After this, the only encryption remaining on all the FSMs is that of *County A*. This enables the *state* to perform model checking , i.e., find an execution plan that leads to service completion without causing violation of the constraint C_{WSP}.

The above steps are repeated for levels $i-1, i-2, \ldots$ and so on. However, at level 0 there is no other requester beyond sr_0 that can verify the overall C_{WSP}, i.e., the constraints of the original requester. Therefore, at level 0, sr_0 performs both composition of the FSM and checking for constraint verification. This may allow sr_0 to learn the internal operations and events occurring during the execution of the component Web services by service providers at level 0. However, the internal operations and events of service providers at level 1 or greater are not disclosed to sr_0.

In case the service providers at level 0 do not want to reveal their internal operations or events to the service requester sr_0, two alternatives can be considered: i) use a third party for constraint verification; and ii) use secure multiparty computation (SMC)approach. For the first alternative, protocol 1 is executed by sr_0 and Protocol 3 is executed by the third party. In addition, all the service providers running Protocol 2 at level 0 send their decryption keys to the third party. The second alternative is the most general and secure. However, as discussed in the Introduction, the existing SMC techniques designed for related problems such as distributed policy composition [5], trust negotiation [6], set intersection and association mining [7], [8] do not consider any ordering relation between input data. Also, the generic circuit evaluation solutions [9], [10] are likely to be very inefficient and impractical.

2) Extraction of Individual WS Execution Plan: For extraction of individual Web service execution plan Protocol 1 (Lines 14-22) is executed by the service requester at level i, Protocol 2 (Lines 5-10) is executed by each service provider at level i. We now explain the extraction process assuming that the service requester is at level 0 and the overall WSP execution plan π_{WSP} has been computed using the protocol steps discussed above:

Step 1. Service requester sr_0 has the component Web service FSM of each of its immediate service provider's component Web service FSM which is commutatively encrypted with the encryption key of the respective service provider and sr_0. For any given service provider sp, sr_0 sends the corresponding commutatively encrypted FSM (i.e., $E_{sr_0}E_{sp}(FSM_{sp})$) to sp. In addition sr_0 also encrypts its execution plan and send $E_{sr_0}(\pi_{WSP})$ to sp.

For example, suppose that the execution plan with respect to the running example of countermeasure inventory tracking process is:

$\pi_{WSP} = S^0_{State}\ Invoke\_InvCheck(County\_A)\ S^0_{County\_A}\ InvCheck\_Start(County\_A)\ S^1_{County\_A}\ Invoke\_InvCheck(City\_1)\ S^1_{City\_1} \ldots S^n_{State}$

The *state* encrypts this execution plan (π_{WSP}) with its encryption key E_{State} and sends it to County A.

Step 2. The extraction of the execution plan is performed by the service requester by performing the $PROJ$ operation (discussed in Section II-B) over the execution plan received from the requester. Before applying the $PROJ$ operation, the service provider encrypts the received execution plan with its encryption key E_{sp} to ensure that projection is carried out over the component Web service FSM that is encrypted with the same set of keys. The execution plan extracted by applying the PROJ operation is also commutatively encrypted with the

keys E_{sr_0} and E_{sp}. For decryption, sp first sends the encrypted execution plan $E_{sr_i}E_{sp}(\pi_{sp})$ to receive $E_{sp}(\pi_{sp})$, which can be decrypted by sp using its key to compute π_{sp} securely.

With reference to the countermeasure inventory tracking example, *County A* receives $E_{State}(\pi_{WSP})$ from the *state* as mentioned above in Step 1. Upon receiving this execution plan, *County A* further encrypt it with its encryption key to generate $E_{County\_A}E_{State}(\pi_{WSP})$. It then applies the $PROJ$ operation to extract its execution plan.

$PROJ_{FSMCountyA||City1||City2}(E_{CountyA}E_{State}(\pi_{WSP}))$ = $E_{CountyA}E_{State}(S^0_{CountyA}\ InvCheck\_Start(CountyA)$ $S^1_{CountyA}Invoke\_InvCheck(City1)\ldots)$ = $E_{CountyA}E_{State}(\pi_{CountyA})$.

For decryption of the extracted execution plan, $County\_A$ sends $E_{County\_A}E_{State}(\pi_{County\_A})$ to the *state* and it receives $E_{County\_A}(\pi_{County\_A})$, which is then decrypted by *County A* using its decryption key to compute π_{County_A} securely.

The above steps are repeated for level 1, 2, and so on.

C. Complexity Analysis

We assume that the service composition hierarchy of Figure 6 has k levels and at any given level each service requester invokes the services of at most m service providers. We also assume that no component Web service FSM has more than n states.

1) Computation Complexity: Consider Protocol 1 for generation of the WSP execution plan (lines 1-13). Since the cost of encryption dominates, for a service requester sr_i at level i, the computation complexity for encrypting the service providers FSMs is $O(mn_i)$, where $n_i = mn_{i+1}, 0 \leq i \leq k-1$ and $n_{k-1} = n$. Expanding this recursion makes the expression $O(mn_i) = O(m^{k-i}n)$. For execution plan extraction phase, the complexity of Protocol 1 remains the same as it involves encryption and decryption of the extracted plan which is linear in the number of states of the FSM.

Protocol 2 also involves encryption of service providers FSM. At level i, such FSM is composed from the FSM of m service providers that are at level $i+1$. Therefore, the computation complexity for Protocol 2 is $O(mn_{i+1}) = O(m^{k-i+1}n)$. For Protocol 3, the complexity is dominated by model checking of the composite FSM, which is linear in the number of its states [11]. Therefore, at level i the computation complexity of Protocol 3 is $O(m^{k-i}n)$

2) Communication Complexity: In Protocol 1, during the WSP execution plan generation phase, service requester sr_i at level i receives m messages of size $O(m^{k-i+1}n)$ one from each its immediate service providers. In addition sr_i sends one message of size $O(m^{k-i}n)$ to the requester one level up. For extraction of the individual Web service execution plan, sr_i sends and receive one message to each of the m service providers of size $O(m^{k-i+1}n)$. In Protocol 2, three messages of size $O(m^{k-i}n)$ are exchanged between the each service provider and service requester at level i. Finally, in Protocol 3, sr_{i-1} receives one message from sr_i of size $O(m^{k-i}n)$.

3) Experimental Results: The table below shows the experimental results for service enactment and constraint verification at different levels of the service composition hierarchy. For these experiments, we consider 4 levels in the service composition hierarchy ($k = 4$). In each level, a service requester interacts with 2 distinct service providers ($m = 2$) and the FSM of each service provider has 10-15 states ($10 \leq n \leq 15$) states. For constraint verification, we ran the model checking tool HyTech [12] on a quad processor Sun SPARC machine with 8 GB RAM.

Level	Computation time for constraint verification
3	0.59s
2	1.82s
1	7.31s
0	84.5s

D. Disclosure Analysis

A service requester at level i ($i \geq 1$) receives the component Web service FSM from its immediate service providers. Each of these FSMs is encrypted with the encryption key of the respective service provider. Therefore, the service requester cannot learn the internal operations or events in the FSMs of its service providers. However, at level 0, since there is no other requester above sr_0, sr_0 can see the composite FSM during constraint verification and can attribute the components of the composite FSM to its immediate service providers.

The service requester sr_{i-1} at level $i-1$, when verifying constraint satrisfaction for sr_i cannot see the events and internal operations included in the FSMs of service providers at level i as such FSMs even after decryption by applying the respective service providers' keys remain encrypted with the key of sr_i. Similarly, the component Web service FSM of sr_i sent to sr_{i-1} is also encrypted with the key of sr_i and so sr_{i-1} cannot learn the internal operations/events of sr_i except at level 0. However, sr_{i-1} can learn which service providers are contracted by sr_i. Note that if P-III is not considered (line 12 of Protocol 1) then sr_{i-1} receives the product automata of constraints, i.e., $E_{sr_i}(C_{WSP}||C_{sr_1}||..C_{sr_i})$. If instead, the service requester sr_i sends the encrypted constraints $(C_{WSP}, C_{sr_1}, ..., C_{sr_i})$ separately (i.e., without composing them in a parallel product automaton), sr_{i-1} could correlate the encrypted values for the individual constraint, say $E_{sr_i}(C_{sr_j})$, to the events in C_{sr_j} that are visible to sr_{i-1}. This would enable sr_{i-1} to infer if some particular event occurs in any of the service providers FSM. To avoid such disclosure, sr_i sends the parallel product automata of constraints rather than the individual constraint FSMs. Since the structure of $E_{sr_i}(C_{WSP}||C_{sr_1}||..C_{sr_i})$ (constraints that are visible to sr_i)and $C_{WSP}||C_{sr_1}||..C_{sr_{i-1}}$ (constraints that are visible to sr_{i-1}) is different, it is unlikely that $C_{WSP}||C_{sr_1}||..C_{sr_{i-1}}$ is isomorphic to the subgraph of $E_{sr_i}(C_{WSP}||C_{sr_1}||..C_{sr_i})$. Hence, it is unlikely that sr_{i-1} can find the association between the encrypted value of some event that appears in any of the constraint visible to sr_{i-1}.

During the execution plan extraction phase, sr_i ($i \geq 1$) sees the overall execution plan, which is an interleaving of

the execution plan of the service providers. But sr_i cannot determine which portions of the execution plan comes from which service provider. However sr_0 may learn the execution plan of its immediate service providers though not of service providers at level 1 or greater. In summary, the privacy requirement P-II which is stricter than P-I is satisfied at each level $i \geq 1$. At level 0, neither P-I nor P-II can be satisfied without involving a trusted third party. Also as discussed earlier, P-III can be satisfied at each level at the expense of increased computational and communication complexity due to delaying the detection of constraint violation of service requesters at higher levels.

V. Related Work

The related work can be broadly categorized into i) policy conformance verification; ii) peer-to-peer service composition; and iii) information flow control.

Policy conformance approaches deal with checking the compatibility between service requester's security/privacy requirements and Web service policies and process model [13], [14]. However, these approaches are primarily designed for single-level service composition and also do not prevent disclosure of internal operations/business process for both service providers and requester.

Peer-to-peer service composition approaches [15], [16], [17] deal with decentralized orchestration of a global business process, wherein participants only provide certain degree of inter-visibility to support peer-to-peer interactions. However, in the cascaded Web service environment the global process constraints may include dependency between events that may occur at multiple non-interacting component Web services. The peer-to-peer based service composition approaches may not ensure compliance of such event constraints without assuming that some special peers/participants have visibility to the processes of other peers at various levels of the service composition hierarchy.

Information flow control in the context of service composition has been recently addressed by She et al. [2], [18]. Their approach enforces information flow control policies in service chains and enables runtime filtering of compositions that do not satisfy the policy requirements. Information flow control approaches assume a multi-level security hierarchy and ensure that information only flows from low security clearance services to high security clearance services. However, these approaches cannot be directly applied to the service composition, specifically enactment problem discussed in this paper that does not assume any multilevel security hierarchy and requires to prevent disclosure of the internal operations of an organization to its partners irrespective of their security clearance level.

VI. Conclusion

In this paper, we study the problem of secure composition of cascaded web services in a collaborative environment. We model three varying levels of privacy requirements that organizations may have. Our main contribution is to develop an encryption based automata theoretic approach that prevents the disclosure of policies and internal operations of service providers against service requesters at different levels in the composition hierarchy, and can satisfy all three levels of privacy. Our experimental results show that our approach is robust and scalable. In the future, we will also explore the use of threshold based encryption to make the protocol completely secure by removing the necessity of service providers having to disclose their decryption keys to the second level requester. We also plan to look at the privacy-preserving aspects of service discovery, service negotiation, and service execution.

Acknowledgements

This work is supported in part by the LUMS Departmental and by the National Science Foundation under Grants No. CNS-0746943 and DUE-1141000.

References

[1] M. Mecella, M. Ouzzani, F. Paci, and E. Bertino, "Access control enforcement for conversation-based web services," in *WWW '06*. pp. 257–266.

[2] W. She, I.-L. Yen, B. Thuraisingham, and E. Bertino, "Policy-driven service composition with information flow control," in *ICWS*, july 2010, pp. 50 –57.

[3] D. Roman and M. Kifer, "Reasoning about the behavior of semantic web services with concurrent transaction logic," in *VLDB '07*. VLDB Endowment, 2007, pp. 627–638.

[4] A. Betin-Can, T. Bultan, and X. Fu, "Design for verification for asynchronously communicating web services," in *WWW '05*. pp. 750–759.

[5] U. Meyer, S. Wetzel, and S. Ioannidis, "Distributed privacy-preserving policy reconciliation," *ICC '07*, pp. 1342–1349, 24-28 June 2007.

[6] J. Li and N. Li, "Oacerts: Oblivious attribute certificates," *IEEE TDSC*, vol. 3, no. 4, pp. 340–352, Oct.-Dec. 2006.

[7] J. Vaidya and C. Clifton, "Secure set intersection cardinality with application to association rule mining," *J. Comput. Secur.*, vol. 13, no. 4, pp. 593–622, 2005.

[8] ——, "Privacy preserving association rule mining in vertically partitioned data," in *KDD '02*. pp. 639–644.

[9] O. Goldreich, "Secure multi-party computation (working draft)," Tech. Rep., 1998.

[10] R. Canetti, U. Friege, O. Goldreich, and M. Naor, "Adaptively secure multi-party computation," Cambridge, MA, USA, Tech. Rep., 1996.

[11] P. Schnoebelen, "The complexity of temporal logic model checking," 2002.

[12] T. A. Henzinger, P.-H. Ho, and H. Wong-toi, "Hytech: A model checker for hybrid systems," *Software Tools for Technology Transfer*, vol. 1, pp. 460–463, 1997.

[13] G. O. M. Yee, "A privacy controller approach for privacy protection in web services," in *ACM SWS '07*. pp. 44–51.

[14] Y. H. Li, H.-Y. Paik, and B. Benatallah, "Formal consistency verification between bpel process and privacy policy," in *PST '06*. pp. 1–10.

[15] Q. Xiaoqiang and W. Jim, "A decentralized services choreography approach for business collaboration," in *IEEE SCC '06*, Sept. 2006, pp. 190–197.

[16] Q. Z. Sheng, B. Benatallah, M. Dumas, and E. O.-Y. Mak, "Self-serv: A platform for rapid composition of web services in a peer-to-peer environment," in *VLDB '02*. pp. 1051–1054.

[17] T. Bultan, X. Fu, R. Hull, and J. Su, "Conversation specification: a new approach to design and analysis of e-service composition," in *WWW '03*, pp. 403–410.

[18] W. She, I.-L. Yen, B. Thuraisingham, and S.-Y. Huang, "Rule-based run-time information flow control in service cloud," in *IEEE ICWS*, july 2011.