Secure service composition with information flow control in service clouds

Ning Xi\textsuperscript{a,b,*}, Cong Sun\textsuperscript{a}, Jianfeng Ma\textsuperscript{a}, Yulong Shen\textsuperscript{a}

\textsuperscript{a} School of Computer Science and Technology, Xidian University, Xi’an, Shaanxi, China
\textsuperscript{b} School of Telecommunications Engineering, Xidian University, Xi’an, Shaanxi, China

**Highlights**
- For the dynamic dependences in service chain, we propose a Secure Information Flow Model for service composition in service clouds.
- We specify the security constraints for each service participant based on the dependences and lattice model.
- We propose a distributed compositional information verification algorithm for the secure service composition in service clouds.
- Our approach simplifies the complexity of model checking and decreases the cost of the verification work effectively.

**Article Info**

Article history:
Received 28 February 2014
Received in revised form 12 December 2014
Accepted 28 December 2014
Available online 28 January 2015

Keywords:
Service cloud
Service composition
Data dependencies
Information flow security

**Abstract**

Service clouds built on cloud infrastructures and service-oriented architecture provide users with a novel pattern of composing basic services to achieve complicated tasks. However, in multiple clouds environment, outsourcing data and applications pose a great challenge to information flow security for the composite services, since sensitive data may be leaked to unauthorized attackers during service composition. Although model checking has been considered as a promising approach to enforce information flow security precisely, its high complexity on modeling and the heavy cost on verification cause great burdens to the process of service composition. In this paper, we propose a distributed approach to composing services securely with information flow control. In our approach, each service component is first verified through model checking, and then a compositional verification procedure is executed to ensure the information flow security along with the composition of these services. The experimental results indicate that our approach can reduce the cost of verification compared with the global verification approach.

© 2014 Elsevier B.V. All rights reserved.

**1. Introduction**

With the advancement of sharing the utility resources in a virtualization way, cloud platforms provide a new and promising paradigm for delivering IT services more effectively and conveniently [1,2]. In service-oriented clouds, people can access various types of services integrated by cloud platform anytime and anywhere. Meanwhile the composition and cooperation of different services have become a new trend for the service delivery in clouds. By composing services together, customers can access more powerful applications, e.g. travel planning composed by tickets booking and room reservation services [3]. For each service, there are different candidate services located in multiple clouds. From a variety of the candidate services, users can select appropriate services to compose the desired applications dynamically according to different criteria and requirements, e.g. QoS, trustworthiness, security and so on [4,5].

However, outsourcing data and applications in clouds pose a great challenge to data security for the service composition in clouds. Due to the multi-domain characteristic of the service clouds, data located in different clouds may have different security levels. For instance, the personal medical records in e-health cloud are with high security level while the position of the ambulance in e-transportation cloud are with lower security level. When these services are composed together for the patient’s emergency, data with different security levels are transmitted among these services respectively. If these services are composed in an insecure way, an operation in a service may transmit confidential data to a public channel and cause the information leakage. Access control has
be widely used for protecting sensitive information of individual service from being released to unauthorized attackers [6–8]. However, for a composite service, data may be processed by several service components from multiple clouds. Access control cannot detect the information leakage caused by the subsequent operations in other service components. Therefore, information flow security is one of the major concerns about the service composition in service clouds.

Due to the dynamic dependencies among objects storing data in different service participants, it is critical to analyze information flow in these composite services precisely. There are various approaches for the information flow enforcements, e.g., type system, program analysis and model checking. Dieter et al. [9] specify the secure composition rules based on type system. These rules are used to ensure the security of dynamically derived data and their proliferation to other web services. Xi et al. [10] obtain the dynamic intra and inter dependencies among the objects in composite service based on program slicing. Nakajima [11] verifies the information flow of composite services in BPEL (Business Process Execution Language) based on model checking. Rossi et al. [12] expand Nakajima’s model [11] to support more flexible and dynamic security policies rather than security labels based on a simple lattice-based model. The contributions using type system and program analysis mainly focus on the information flow enforcement on programming languages, while model checking can be used to validate both programs and models in abstract forms.

Besides, service-oriented clouds compose a distributed system with multiple domains. It is necessary to design a secure and efficient service composition algorithm in a distributed environment. She et al. [13] proposes a policy-driven service composition approach with information flow control in multiple service domains. In She’s approach, the security levels of the output in a component is computed according to the transformation factor, then the insecure composite service is filtered. However, it is hard to define the transformation factor precisely. She et al. also develops a run-time information flow control model for service composition in clouds to analyze the dependencies between the inputs and outputs dynamically in [14]. But it requires that the initial user inputs for the service execution be static. So when user’s initial inputs change, the verification process needs to be rebuilt, which brings extra cost for the secure service composition. Although model checking [11,12,15] can be used to analyze secure information flow precisely, the traditional model checking approaches must perform a global verification on the composite service. It is impractical to employ a centralized entity in multiple clouds to verify the information flow security in a global way. Moreover, the cost of verification can increase rapidly when the application involves more components and the number of the candidate services increases. First, the same service component has to be verified in different composite services. Second, the state explosion problem arises if each service component is complicated.

In order to ensure the information flow security and improve the efficiency of the service composition process, we present a distributed service composition approach with information flow control in service clouds. In our approach, each service component is verified through model checking, which can reduce the complexity of modeling compared to the global verification approach. Then compositional information flow verification algorithm is proposed for the secure service composition in service clouds, which works in a distributed way.

The rest of the paper is structured as follows. Section 2 presents the secure information flow model for service composition in service clouds. In Section 3, we propose the secure service composition approach based on the secure information flow model. Section 4 evaluates the proposed approach. Section 5 concludes the paper.
2.3. Secure information flow in service component

During the execution of the service chain, each service $s_i$ may read a set of input data $In_i$ and write a set of output data $Out_i$. As shown in Fig. 3, for the input data of $s_i$, we have $In_i = \{In^M_i, In^N_i\}$, where

- $In^M_i = \{In^M_{i,1}, In^M_{i,2}, \ldots, In^M_{i,n}\}$ is the set of input data that $s_i$ receives from its predecessor $s_{i-1}$.
- $In^N_i = \{In^N_{i,1}, In^N_{i,2}, \ldots, In^N_{i,n}\}$ is the set of input data that $s_i$ reads from cloud’s resources.

For the output data of $s_i$, we have $Out_i = \{Out^M_i, Out^N_i\}$, where

- $Out^M_i = \{Out^M_{i,1}, Out^M_{i,2}, \ldots, Out^M_{i,n}\}$ is the set of all output data that $s_i$ sends to its successor $s_{i+1}$.
- $Out^N_i = \{Out^N_{i,1}, Out^N_{i,2}, \ldots, Out^N_{i,n}\}$ is the set of all output data that $s_i$ writes to cloud’s resources.

According to the following syntax of core language, we can specify the function $F_i$ of $s_i$.

$$F_i := a; a'$$
$$a := \text{skip} \mid \text{input} (in_{i,x}, \text{var}) \mid \text{output} (out_{i,y}, \text{var})$$
$$| \text{var} := e \mid a; a' \mid \text{if} (e) \text{then} a \text{else} a' \mid \text{while} (e) \text{a}$$

$$y = M | R$$

$$e := \text{var} | e 0 e Q$$
$$Q := + | - | = | < .$$

Based on the definition of multi-level security model and the analysis of information flow in component $s_i$, the concept of interference and noninterference [16] are introduced to ensure the information flow security. We use $\rightarrow$ and $\rightarrow'$ to represent the interference and noninterference between different objects. Then secure information flow in $s_i$ can be defined as follows:

Definition 2.2. The information flow in service component $s_i$ is considered secure if it satisfies that for $\forall u \in HIn_i, \forall v \in LOut_i$, there is no interference between $u$ and $v$, namely, $u \rightarrow v$.

In Definition 2.2, $HIn_i$ and $LOut_i$ are the sets of inputs and outputs with the security level $H$, i.e. $HIn_i = \{\text{in}_{i,x}| \text{in}_{i,x} \in In_i \land \text{Sec}(\text{in}_{i,x}) = H\}; LOut_i = \{\text{out}_{i,y}| \text{out}_{i,y} \in Out_i \land \text{Sec}(\text{out}_{i,y}) = H\}$. $In_i$ and $Out_i$ are the set of inputs and outputs with the security level $I$, i.e. $In_i = \{\text{in}_{i,x}| \text{in}_{i,x} \in In_i \land \text{Sec}(\text{in}_{i,x}) = I\}; LOut_i = \{\text{out}_{i,y}| \text{out}_{i,y} \in Out_i \land \text{Sec}(\text{out}_{i,y}) = I\}$.

2.4. Secure information flow in service chain

In the service chain $S_{nk}$, data is processed in different components. For example, $\forall u \in In^M_{nk}$ may be computed from $S_{i-1}$ or its predecessor, and $\forall v \in Out^M_{nk}$ may be further processed by the successors of $s_i$ and finally delivered to service $s_j$, $j > i$. The information flow security of $S_{nk}$ is defined as follows:

Definition 2.3. The information flow in service chain $S_{nk}$ is considered secure if it satisfies that for $\forall u \in HIn_{nk}, \forall v \in LOut_{nk}$, there is no interference between $u$ and $v$, namely, $u \rightarrow v$, where $HIn_{nk} = \{\text{in}_{n,x}| \text{in}_{n,x} \in In_{nk} \land \text{Sec}(\text{in}_{n,x}) = H\}; LOut_{nk} = \{\text{out}_{n,x}| \text{out}_{n,x} \in Out_{nk} \land \text{Sec}(\text{out}_{n,x}) = L\}$.

According to the definition of the service chain, we can obtain that $In_{nk} = \bigcup In^M_i$, $Out_{nk} = \bigcup Out^M_i$, $0 \leq i \leq n + 1$, and $\forall i, 0 \leq i \leq n$, $Out_{i} = In_{i+1}$. There is $\text{In}_0 = \phi$, $\text{Out}_0 = \phi$, $\text{In}_{n+1} = \phi$ and $\text{Out}_{n+1} = \phi$.

Lemma 2.1. In a service chain $S_{nk}$, for $\forall u \in In_i \cup Out_i$, $\forall v \in In_j \cup Out_j$, $0 \leq i < j$, if $u \rightarrow v$, there exist $w_1 \in Out_{i-1}$, $w_2 \in In_j$ such that $u \rightarrow w_1 \land w_1 \rightarrow w_2 \land w_2 \rightarrow v$.

Proof. Assume that $\forall u \in Out_{i-1}$, $w_2 \in In_j$, $u \rightarrow w_1 \land w_1 \rightarrow w_2 \land w_2 \rightarrow v$, then there is also $u \rightarrow v$.

Case 1: $u \rightarrow w_{i-1}$

$Out_{i-1}$ is the only way that $s_i (0 \leq i < j)$ passes the value of intermediate result to $In^M_j$ according to the definition of the service chain. So if $u \rightarrow w_1$, there is $u \rightarrow w_2$, and for $\forall u \in In_j$, $u \rightarrow v$. Then we can obtain that $u \rightarrow v$, which is contradictory with our assumption.

Case 2: $u \rightarrow w_1 \land w_1 \rightarrow w_2$

Because $Out_{i-1} = In_{i+1}$ in $S_{nk}$, for $\forall w_1 \in Out_{i-1}$, there must be $w_2 \in In^M_j$ satisfying $w_1 \rightarrow w_2$. It is also contradictory with our assumption.

Case 3: $u \rightarrow w_1 \land w_1 \rightarrow w_2 \land w_2 \rightarrow v$.

$In^M_j$ is the only way that $s_j$ receives the output of $s_i (0 \leq i < j)$ according to definition of the service chain, and $\forall w_3 \in In^M_j$, $u \rightarrow w_3$. So if $w_2 \rightarrow v$, it can be obtained that $u \rightarrow v$, which is also contradictory with our assumption.

In conclusion, our assumption is false and Lemma 2.1 is proved. □

Lemma 2.2. If the information flow of each service in the first $m$ steps of $S_{nk}$ is secure and for $\forall w_1 \in Out_{m}$, $w_2 \in In_{m+1}$, $0 \leq i \leq m$, they satisfy $\text{Sec}(w_1) \leq \text{Sec}(w_2)$ when $w_1 \rightarrow w_2$, then we have $\forall u \in HIn_{nk}, v \in LOut_{nk}$, $0 \leq i \leq j \leq m$, $u \rightarrow v$.

Proof. First, let $m = 1$, then there are three cases to be considered, i.e. the information flow in $s_0$ in $s_1$ and between $s_0$ and $s_1$.

Case 1: $u \in HIn_0, v \in LOut_0$, $u \rightarrow v$. $s_0$ is secure according to Definition 2.2.

Case 2: $u \in HIn_1, v \in LOut_1$, $u \rightarrow v$. $s_1$ is secure according to Definition 2.2.

Case 3: $u \in HIn_0, v \in LOut_1$, $u \rightarrow v$.

Assume that $\exists u \in HIn_0$, $\exists v \in LOut_0$, $u \rightarrow v$. Then it can be obtained that $\exists w_1 \in Out_0$, $\exists w_2 \in In_{m+1}$, $u \rightarrow w_1$, $w_1 \rightarrow w_2$ and $w_2 \rightarrow v$ according to Lemma 2.1.

$s_0$ is secure, so $u \in HIn_0$ and $u \rightarrow w_1$ provides $w_1 \in HOut_m$.

And $\text{Sec}(w_1) \leq \text{Sec}(w_2)$ provides $w_2 \in HIn_1$.

Because $v \in LOut_1$, $w_2 \rightarrow v$ is contradictory with the condition that $s_1$ is secure. Then Case 3 is true.

In conclusion, when $m = 1$, Lemma 2.2 is proved.

Then we assume Lemma 2.2 is true when $m = n$, i.e. the information flow of each service in first $n$ step of $S_{nk}$ is secure and for $\forall w_2 \in Out^M_m \cup In_{m+1}$, $0 \leq i \leq n$, $w_1 \rightarrow w_2$, they satisfy $\text{Sec}(w_1) \leq \text{Sec}(w_2)$, there is $\forall u \in HIn_n$, $v \in LOut_n$, $0 \leq i \leq m$, $i \leq j \leq m$, there is $u \rightarrow v$. The lemma is proved as follows when $m = n + 1$.

Case 1: $u \in HIn_{n+1}, v \in LOut_{n+1}$, $u \rightarrow v$.

Because information flow in $S_{nk}$ satisfies the noninterference, the proposition can be proved.
flow security of the service chain. The framework is composed of Candidate Services (CS), Security Authorities (SA) and Cloud Platforms (CP). According to Theorem 2.1, there are two phases for the information flow verification, i.e. component verification and compositional verification. First, each service component is verified by its local SA, and SA generates service certificate for the following compositional verification. When these components are going to be composed, the information flow between adjacent service components is verified by cooperating different SAs in multiple clouds.

3.2. Component verification by model checking

The component verification is the preparation phase for the compositional verification. When a service is going to be deployed into the cloud platform, it needs to be verified by SA first. A certificate specifying security property of the secure service is generated for the following verification by SA, while the insecure one is not allowed to be deployed. The verification procedure is shown in Fig. 5.

In the phase of component verification, SA validates each service component through model checking. For the dynamic dependencies in service, self-composition [17] is adapted for the verification. There are four steps for the model checking, i.e. service modeling, model preprocessing, properties modeling and model checking.

3.2.1. Service modeling

In this step, LTS (Labeled Transition System) is used to model s_i. First, the state of s_i is defined as follows:

**Definition 3.1.** A state of s_i during the execution is \( \mu = (I, V, O) \), I, V and O represent the mappings from inputs, outputs and variables to their values respectively, i.e. \( I : I_n \rightarrow Val \), \( V : Var \rightarrow Val \) and \( O : Out \rightarrow Val \). Val is the domain of values used in F_i.

The initial state of F_i is \( \mu_0 \) while the end state is \( \mu_n \), which are determined by the value of input and output respectively.

**Definition 3.2.** s_i can be modeled as a LTS, i.e. \( \mathcal{M} = (\mu, \rightarrow) \), where \( \mu \) is the state of s_i, and \( \rightarrow \) represents the transitions among the states.

According to the syntax of F_i, the rules of state transitions \( \Phi(a, n_k, n_l, \rightarrow) \) is defined as follows, where \( n_k \) and \( n_l \) are the entry and exit point of the action a in F_i.

\[
\Phi(\text{skip}, n_k, n_l, \rightarrow) = (\pi(n_k) \rightarrow \pi(n_l)) [I = I \wedge O = O \wedge V = V]
\]

\[
\Phi(\text{input}(\text{in}_{i,x}^y, \text{var}), n_k, n_l, \rightarrow) = (\pi(n_k) \rightarrow \pi(n_l)) [I = I \wedge O = O \wedge V = V] \]

\[
\Phi(\text{output}(\text{out}_{i,x}^y, \text{var}), n_k, n_l, \rightarrow) = (\pi(n_k) \rightarrow \pi(n_l)) [I = I \wedge O = O \wedge V = V]
\]
In this step, service model represented by LTS is preprocessed by a self-composition way for the information flow verification. For a service model \( \mathcal{M} \) with initial state \( \mu_0 \), the self-composition process is shown as follows. 

1. Copy the service model \( \mathcal{M} \) and generate a new model \( \mathcal{M}' \).
2. In the initial state \( \mu_0 \) and \( \mu_0' \), for all \( i \in L_{in} \), let \( I_{lo,i}(li_{-i,x}) := I_{lo,i}(li_{-i,x}) \), which means the values of low-level inputs of two models are equal in the initial state.

### 3.2.3. Properties modeling

For \( \mu = (I, O, V) \), \( \mu' = (I', O', V') \), \( \mu \approx^P \mu' \) means that for all \( i \in L_{in} \), there is \( I(I(i)) = I'(li_{-i,x}) \). \( \mu \approx^P \mu' \) means that for all \( i \in L_{out} \), there is \( O(lo_i) = O'(lo_i) \). So we can obtain the following theorem based on Definition 2.2.

**Theorem 3.1.** The information flow in \( s_i \) is secure if and only if for all \( \forall \mu_0, \mu'_0 \) when \( \mu_0 \approx^P \mu'_0 \), there is \( \mu_e \approx^P \mu'_e \), where \( \mu_e = F_i(\mu_{lo,i}), \mu'_e = F_i(\mu'_{lo,i}) \).

According to Theorem 3.1, the security condition of information flow can be represented as the following assertion:

**For each** \( l_{-i,x} \in L_{out} \) \n
\[
\text{assert } \left( \bigwedge_{0 \leq x \leq |L_{out}|} (O_{lo_i}(lo_{-i,x}) = O'_{lo_i}(lo_{-i,x})) \right).
\]

### 3.2.4. Model checking

In this step, we input our self-composition service model \( \mathcal{M} \) and \( \mathcal{M}' \) into model checking tools, e.g. SPIN [18], and we input the security properties as assertions. If there is no error returned, it means that information flow in \( s_i \) is secure. Otherwise, information leakage is found and the tool returns a counterexample.

If \( s_i \) is secure, a service certificate \( C_e \) signed by the local SA is generated for the compositional verification. \( C_e \) is described as the attribute certificates defined by [19], which can specify the properties of service \( s_i \) as a set of statements, i.e. service id, input \( l_{-i} \), output \( Out_i \), and data security level Sec(o). In order to improve the efficiency of the verification process and decrease the time cost on service composition, the component verification phase can be executed in an off-line way.

### 3.3. Compositional verification

In SoCS, \( s_{10} \) is dynamically composed by different candidate service components \( s_{1,j} \in S_i \) in each step. We propose compositional verification algorithm to ensure the information flow security based on Theorem 2.1. The compositional verification procedure is shown in Fig. 6.

**Fig. 6.** Compositional verification procedure.

### Algorithm 1 Secure_Chain_Composition()

**Input:** Selected Service \( s_{1,0} \), Successor’s Candidate Service Set \( S_{1,1} \).

**Output:** Current Secure Execution Path \( P \).

1. \( \text{SA} \) waits for the message to start the verification work.
2. If the message is start message then
3. \( \text{SA} \) pushes selected service \( s_{1,0} \) into Execution Path \( P \).
4. If \( s_{1,0} \) is the final step of service chain then
5. \( \text{SA} \) sends success message and the secure execution path to user.
6. else
7. \( \text{SA} \) requests each candidate service’s cert, and validates them where illegal one is dropped.
8. \( \text{SA} \) verifies the information flow between \( s_{1,0} \) and each candidate service \( s_{1,j} \) based on the theorem 2.1.
9. The passed services are push into passed candidate service set \( S^p_{1,j} \).
10. if
11. if There is no passed services then
12. \( \text{SA} \) sends fail message to its predecessor’s SA.
13. else
14. \( \text{SA} \) sends start message to each passed service \( s_{1,j} \)’s SA.
15. if
16. if the message is fail message then
17. \( \text{SA} \)’s failure counter increases.
18. if failure counter equals to the number of the candidate services in \( S^p_{1,j} \) then
19. if \( f = 0 \) then
20. \( \text{SA} \) sends fail message to its predecessor’s SA.
21. else
22. \( \text{SA} \) notices the user that there is no secure service execution path.
23. else
24. end if
25. end if

\( s_i \) requests \( s_{1,j} \)’s certificate first, and sends it with \( s_{1,0} \)’s certificate to \( s_i \)’s SA, i.e. \( S_A \). During the verification, \( S_A \) checks the signature of \( s_{1,j} \)’s SA first, and the illegal one will be dropped. Then \( S_A \) verifies whether the information flow between \( s_i \) and \( s_{1,j} \) is secure according to Theorem 2.1.

### 3.4. Secure service chain composition algorithm with information flow control in service clouds

Based on the component and compositional verification, we propose a distributed secure service chain composition algorithm with information flow control in multiple clouds, which is implemented in each cloud’s SA. For stepwise composition of the service chain, cloud platform first performs the compositional verification procedure to verify the candidate services in \( S_{1,1} \) and obtains the validated candidate service set \( S^p_{1,1} \), and each validated service will be put into a secure execution path \( P \), and then the cloud platform will notice these validated services to continue the composition process of the following candidate services. Because \( S_{0} \) and \( S_{n+1} \) represent the user, the first composition step is executed by the user while the secure composition path \( P \) is finally sent to the user. The distributed secure composition algorithm for the service chain in service clouds is presented in Algorithm 1.

In Algorithm 1, we use three types of messages for the synchronization of the verification procedure, i.e. start, success and fail. The message start is used to allow the SA to execute the verification work. When the verification work is done, the message success
with the executable path is sent to user, i.e. $s_{n+1}$. If there are no secure candidate services in the following steps, the message fail is sent to its predecessor, and user will be noticed that there is no secure service execution path when all validated services in $s_{p}$ are failed. When the verification returns a failure to the user, the user needs to enlarge the scope of service discovery to add more candidate services for the composition procedure.

4. Performance analysis and evaluations

Through the security analysis in Section 3, the information flow security can be ensured by Theorem 2.1. In this section, we investigate the performance of our approach compared to that of the global model checking.

4.1. Time complexity analysis

According to the Algorithm 1, for each verification, the time complexity is $O(n)$, where $n$ is the number of the candidate service set $S_i$. So the complexity of each stepwise verification is $O(n \cdot m)$, where $m (m \leq n)$ is the number of the validated service set $S_{p_i}$. Therefore, the complexity of the compositional verification is $O(n \cdot m \cdot k)$, where $k$ is the number of steps in the composite service. Besides, the time complexity for component verification is $O(n \cdot k)$. Therefore, the time complexity of the whole verification is $O(n \cdot k) + O(n \cdot m \cdot k)$. Meanwhile, the time complexity of global model checking is $O(n^m)$. Compared with the global model checking, our algorithm is superior in time complexity.

4.2. Experiments and evaluations

Here we use service chain in [20] as the test case. In our approach, there are two different phases, i.e. component and compositional verification. We use SPIN to verify each component and then use NS-3 [21] to simulate our secure service chain composition algorithm.

Fig. 7 shows the number of verified states and the time costs of the verification. In global model checking approach, we need to build all possible models of the composite service and verify them with SPIN. With the increment of the number of candidate services, the complexity of modeling composite service increase at an exponential rate. So the number of the states that SPIN needs to search rises vastly, which also make the verification time to increase sharply.

Fig. 8 shows the time cost of different phases in our approach. When the amount of the candidate services is small, the time cost mainly comes from component verification. But with the increase of the number of candidate services, the time cost of component verification raises slowly while that on compositional verification rises vastly due to the complexity of the composite service. Besides, the synchronization procedure of the compositional verification costs extra communication overhead. When the amount of the candidate services is greater than 18, the time cost mainly comes from compositional verification.

Fig. 9 shows the total time cost of different approaches. For the global model checking, it is a repetitive and complex work to model and verify each executive composite service. However, in our approach, individual component is verified by model checking tools first, and then the composition of these components are verified according to the security conditions in Theorem 2.1. Because our approach avoids the repetitive verification of each component, the complexity of compositional verification is reduced and the efficiency of composition process is improved compared to the global verification.

5. Conclusion

In this paper, we propose a distributed secure service composition approach with information flow control in service clouds. For the dynamic composition of different services, our approach first verifies each service component by model checking, and then ensures the information flow security during the process of service composition. Through experiments and evaluations, we show that our approach can reduce the cost of verification compared to the global verification. Service chain is a simple composite service, and more complicated services with the conditional and loop structure will be considered in the future. The prototype system is under development to show our approach can be leveraged in service systems with a large scale.
Acknowledgments

We wish to thank the anonymous reviewers for their highly valuable and constructive comments. This work is supported by Program for the Key Program of NSF-Guangdong Union Foundation (U1135002), National Natural Science Foundation of China (61303033), Aviation Science Foundation of China (2013ZC31003, 20141931001).

References


