Future Generation Computer Systems 49 (2015) 142–148

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Secure service composition with information flow control in service clouds

Ning Xi [a,b,*], Cong Sun [a], Jianfeng Ma [a], Yulong Shen [a]

[a] School of Computer Science and Technology, Xidian University, Xi'an, Shaanxi, China
[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.

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

* Corresponding author at: School of Computer Science and Technology, Xidian University, Xi'an, Shaanxi, China.

E-mail addresses: xining@stu.xidian.edu.cn (N. Xi), suncong@xidian.edu.cn (C. Sun), jfma@mail.xidian.edu.cn (J. Ma), ylshen@mail.xidian.edu.cn (Y. Shen).

http://dx.doi.org/10.1016/j.future.2014.12.009
0167-739X/

been 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 reverified 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 secure 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.

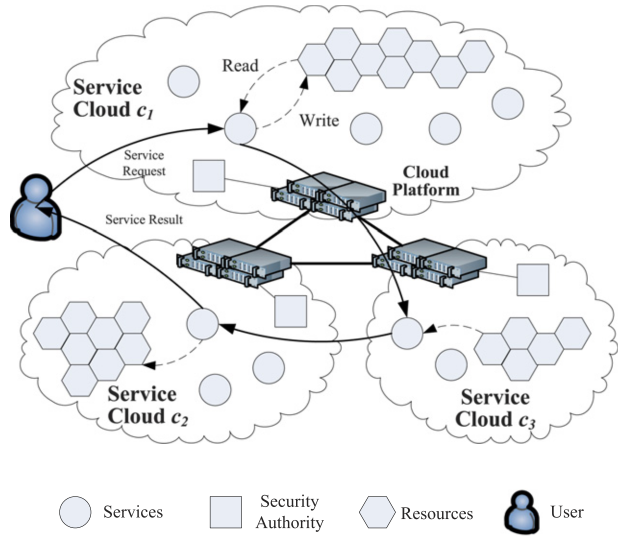


Fig. 1. Service-oriented clouds system.

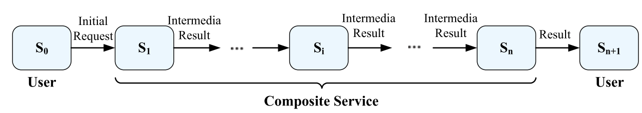


Fig. 2. Service chain.

2. Secure information flow model for service composition in service clouds

2.1. Service-oriented clouds system model

Fig. 1 shows a typical architecture of the Service-Oriented Clouds System (SoCS) consisting of multiple service clouds. In each service cloud c, there are various services s_i and resources R_c, and a security authority SA_c, where SA_c is responsible for the security management of cloud resources and services, e.g., data security level, security certificate, and so on.

The individual services s_i located in different service clouds can be composed to generate a more powerful service. In this paper, we investigate a typical scenario of service composition, i.e., the service chain s_{ch} [10], as shown in Fig. 2. Compared to the complex composite service with conditional and loop structures, service chain is easy to control and deploy, which is widely used in service composition. In a service chain s_{ch}, each service s_i has a single predecessor s_{i-1} and a single successor s_{i+1}. The initial service s_0 denotes the user who sends the initial request. Service s_{n+1} denotes the user who receives the result from the composite service.

2.2. Multi-level security model

In SoCS, data are with different sensitivities, e.g. environment data are public while personal position and medical data are private. The multi-level security model can be defined on a lattice as Definition 2.1.

Definition 2.1. Multi-Level Security Model is defined as a lattice $(SL, \leq)$, where SL is a finite set of security levels that is totally ordered by $\leq$.

For simplicity, we mainly consider the binary security model, i.e., $SL = \{L, H\}$, where $L \leq H$.

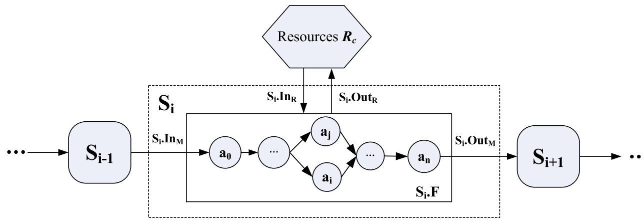


Fig. 3. Information flow in the component.

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 $In_i = \{In_i^M, In_i^R\}$, where

- $In_i^M = \{In_{i,1}^M, In_{i,2}^M, \ldots, In_{i,n}^M\}$ is the set of input data that s_i receives from its predecessor s_{i-1}.
- $In_i^R = \{In_{i,1}^R, In_{i,2}^R, \ldots, In_{i,n}^R\}$ is the set of input data that s_i reads from cloud's resources.

For the output data of s_i, we have $Out_i = \{Out_i^M, Out_i^R\}$, where

- $Out_i^M = \{Out_{i,1}^M, Out_{i,2}^M, \ldots, Out_{i,n}^M\}$ is the set of all output data that s_i sends to its successor s_{i+1}.
- $Out_i^R = \{Out_{i,1}^R, Out_{i,2}^R, \ldots, Out_{i,n}^R\}$ 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.

$$F ::= a;\, F'$$
$$\begin{aligned} a \ ::= \ & skip \mid input\ (in_{i,x}^y, var) \mid output\ (out_{i,x}^y, var) \\ & \mid \mathrm{var} := e \mid a;\, a' \mid if\,(e)\ then\ a\ else\ a' \mid while\ (e)\ a \\ & (y = M|R) \end{aligned}$$
$$e ::= var|eQe$$
$$Q ::= + \mid - \mid = \mid < .$$

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 $\rightsquigarrow$ and $\not\rightsquigarrow$ 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, $u \not\rightsquigarrow v$.

In Definition 2.2, HIn_i and $HOut_i$ are the set of inputs and outputs with the security level H, i.e. $HIn_i = \{in_{i,x}|in_{i,x} \in In_i \wedge Sec(in_{i,x}) = H\}$; $HOut_i = \{out_{i,x}|out_{i,x} \in Out_i \wedge Sec(out_{i,x}) = H\}$. LIn_i and $LOut_i$ are the set of inputs and outputs with the security level L, i.e. $LIn_i = \{in_{i,x}|in_{i,x} \in In_i \wedge Sec(in_{i,x}) = L\}$, $LOut_i = \{out_{i,x}|out_{i,x} \in Out_i \wedge Sec(out_{i,x}) = L\}$. Sec maps the objects to their security levels, i.e. $Sec : In_i \cup Out_i \rightarrow SL$. It can also be obtained that $In_i = HIn_i \cup LIn_i$, $Out_i = HOut_i \cup LOut_i$.

2.4. Secure information flow in service chain

In the service chain s_{ch}, data is processed in different components. For example, $\forall in \in In_i^M$ may be computed from s_{i-1} or its predecessor, and $\forall out \in Out_i^M$ 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_{ch} is defined as follows:

Definition 2.3. The information flow in service chain s_{ch} is considered secure if it satisfies that for $\forall u \in HIn_c$, $\forall v \in LOut_c$, there is no interference between u and v, namely, $u \not\rightsquigarrow v$, where $HIn_c = \{in_{c,x}|in_{c,x} \in In_c \wedge Sec(in_{c,x}) = H\}$, $LOut_c = \{out_{c,x}|out_{c,x} \in Out_c \wedge Sec(out_{c,x}) = L\}$.

According to the definition of the service chain, we can obtain that $In_c = \bigcup In_i^R$, $Out_c = \bigcup Out_i^R$, $0 \le i \le n+1$, and $\forall i$, $0 \le i \le n$, $Out_i^M = In_{i+1}^M$. There is $In_0^M = \phi$, $Out_0^R = \phi$, $In_{n+1}^R = \phi$ and $Out_{n+1}^M = \phi$.

Lemma 2.1. *In a service chain s_{ch}, for $\forall u \in In_i \cup Out_i$, $\forall v \in In_j \cup Out_j$, $0 \le i < j$, if $u \rightsquigarrow v$, there exist $w_1 \in Out_{j-1}^M$, $w_2 \in In_j^M$ such that $u \rightsquigarrow w_1 \cap w_1 \rightsquigarrow w_2 \cap w_2 \rightsquigarrow v$.*

Proof. Assume that $\forall w_1 \in Out_{j-1}^M$, $w_2 \in In_j^M$, $u \not\rightsquigarrow w_1 \cup w_1 \not\rightsquigarrow w_2 \cup w_2 \not\rightsquigarrow v$, there is also $u \rightsquigarrow v$.

Case 1: $u \not\rightsquigarrow w_1$.
Out_{j-1}^M is the only way that s_i $(0 \le i < j)$ passes the value of intermediate result to In_j^M according to the definition of the service chain. So if $u \not\rightsquigarrow w_1$, there is $u \not\rightsquigarrow w_2$, and for $\forall w_3 \in In_j^R$, $u \not\rightsquigarrow w_3$. Then we can obtain that $u \not\rightsquigarrow v$, which is contradictory with our assumption.

Case 2: $u \rightsquigarrow w_1$ and $w_1 \not\rightsquigarrow w_2$.
Because $Out_i^M = In_{i+1}^M$ in s_{ch}, for $\forall w_1 \in Out_{j-1}^M$, there must be $w_2 \in In_j^M$ satisfying $w_1 \rightsquigarrow w_2$. It is also contradictory with our assumption.

Case 3: $u \rightsquigarrow w_1$, $w_1 \rightsquigarrow w_2$ and $w_2 \not\rightsquigarrow v$.
In_j^M is the only way that s_j receives the output of s_i $(0 \le i < j)$ according to definition of the service chain, and $\forall w_3 \in In_j^R$, $u \not\rightsquigarrow w_3$. So if $w_2 \not\rightsquigarrow v$, it can be obtained that $u \not\rightsquigarrow v$, 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_{ch} is secure and for $\forall w_1 \in Out_i^M$, $w_2 \in In_{i+1}^M$, $0 \le i \le m$, they satisfy $Sec(w_1) \le Sec(w_2)$ when $w_1 \rightsquigarrow w_2$, then we have $\forall u \in \bigcup HIn_i$, $v \in \bigcup LOut_j$, $0 \le i \le j \le m$, $u \not\rightsquigarrow v$.*

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: $\forall u \in HIn_0$, $\forall v \in LOut_0$, $u \not\rightsquigarrow v$. s_0 is secure according to Definition 2.2.

Case 2: $\forall u \in HIn_1$, $\forall v \in LOut_1$, $u \not\rightsquigarrow v$. s_1 is secure according to Definition 2.2.

Case 3: $\forall u \in HIn_0$, $\forall v \in LOut_1$, $u \not\rightsquigarrow v$.

Assume that $\exists u \in HIn_0$, $\exists v \in LOut_1$, $u \rightsquigarrow v$.
Then It can be obtained that $\exists w_1 \in Out_0^M$, $\exists w_2 \in In_1^M$, $u \rightsquigarrow w_1$, $w_1 \rightsquigarrow w_2$ and $w_2 \rightsquigarrow v$ according to Lemma 2.1.
s_0 is secure, so $u \in HIn_0$ and $u \rightsquigarrow w_1$ provides $w_1 \in HOut_0^M$.
And $Sec(w_1) \le Sec(w_2)$ provides $w_2 \in HIn_1^M$.
Because $v \in LOut_1$, $w_2 \rightsquigarrow v$ 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_{ch} is secure and for $\forall w_2 \in Out_i^M \cup In_{i+1}^M$, $0 \le i \le n$, $w_1 \rightsquigarrow w_2$, they satisfy $Sec(w_1) \le Sec(w_2)$, there is $\forall u \in \bigcup HIn_i$, $v \in \bigcup LOut_j$, $0 \le i \le m$, $i \le j \le m$, there is $u \not\rightsquigarrow v$. The lemma is proved as follows when $m = n + 1$:

Case 1: $\forall u \in HIn_{n+1}$, $\forall v \in LOut_{n+1}$, $u \not\rightsquigarrow v$.
Because information flow in s_{n+1} satisfies the noninterference, the proposition can be proved.

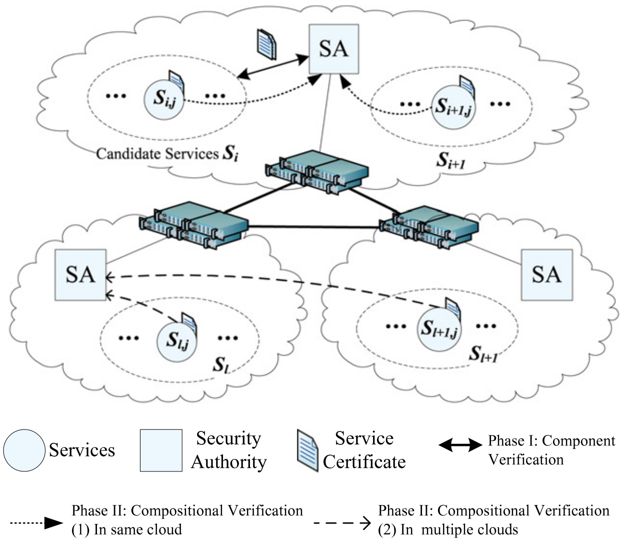


Fig. 4. Information flow verification framework for service chain in multiple clouds.

Case 2: $\forall u \in HIn_i,\ \forall v \in LOut_{n+1},\ 0 \le i < n+1, u \nrightarrow v$.

Assume that $\exists u \in HIn_i,\ \exists v \in LOut_{n+1},\ u \rightsquigarrow v$.

Lemma 2.1 provides that $\exists w_1 \in Out_n^M,\ \exists w_2 \in In_{n+1}^M,\ u \rightsquigarrow w_1$, $w_1 \rightsquigarrow w_2$ and $w_2 \rightsquigarrow v$.

Since Lemma 2.2 is true when $m = n,\ \forall u \in HIn_i,\ 0 \le i < n$, $u \to w_1$ provides $w_1 \in HOut_n^M$.

And $Sec(w_1) \le Sec(w_2)$ provides $w_2 \in HIn_{n+1}^M$.

Because $v \in LOut_{n+1},\ w_2 \in v$ is contradictory with the condition that information flow in s_{n+1} is secure. Then Case 2 is true.

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

Based on Lemmas 2.1 and 2.2, we can obtain the following theorem.

Theorem 2.1. *For a service chain s_{ch}, the information flow is secure if each service component s_i, $0 \le i \le n+1$, satisfies the following two conditions:*

(1) *In each service component s_i, for $\forall u \in HIn_i$, $\forall v \in LOut_i$, there is $u \nrightarrow v$.*
(2) *In adjacent services s_i and s_{i+1}, for $\forall w_1 \in Out_i^M,\ w_2 \in In_{i+1}^M$, $w_1 \rightsquigarrow w_2$, there is $Sec(w_1) \le Sec(w_2)$.*

Proof. Let $m = n + 1$. It can be obtained that for $\forall u \in HIn_i,\ v \in LOut_j,\ 0 \le i \le j \le n+1$, there is $u \nrightarrow v$ according to Lemma 2.2. Theorem 2.1 is proved. □

3. Secure service chain composition approach with information flow control

3.1. Information flow verification framework for service chain in multiple clouds

In SoCS, there are different candidate service components $s_{i,j}$ for each service s_i in s_{ch}. The candidate service set is defined as $S_i = \{s_{i,j} | 0 \le i \le n+1,\ 0 \le j \le |S_i|\}$. In order to prevent information leakage during service composition, it is essential for us to select appropriate candidate services $s_{i,j}$ to compose a secure service chain. Based on the secure information flow model, the information flow verification framework for service chain in SoCS is proposed in Fig. 4.

This is a distributed verification framework in which SAs cooperate with each other for the verification of information 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.

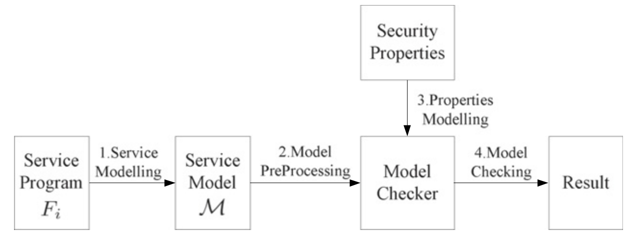


Fig. 5. Component verification procedure.

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. $I : In_i \to Val$, $V : Var_i \to Val$ and $O : Out_i \to Val$. Val is the domain of values used in F_i.

The initial state of F_i is μ_0 while the end state is μ_e, which are determined by the value of input and output respectively.

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

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

$$\Phi(skip, n_k, n_l, \to) = \{\langle n_k\rangle \to \langle n_l\rangle | I' = I \wedge O' = O \wedge V' = V\}$$

$$\begin{aligned}&\Phi(input(in_{i,x}^y, var), n_k, n_l, \to)\\ &\quad = \{\langle n_k\rangle \to \langle n_l\rangle | I' = I \wedge O' = O \wedge V'(var)\\ &\quad = I(in_{i,x}^y) \wedge (\forall var' \ne var, V'(var') = V(var'))\}, \quad (y = M|R)\end{aligned}$$

$$\begin{aligned}&\Phi(output(out_{i,x}^y, var), n_k, n_l, \to)\\ &\quad = \{\langle n_k\rangle \to \langle n_l\rangle | I' = I \wedge O'(out_{i,x}^y)\\ &\quad = V(var) \wedge V' = V\}, \quad (y = M|R)\end{aligned}$$

$$
\begin{aligned}
&\Phi(var := e, n_k, n_l, \rightarrow) \\
&\quad = \{\langle n_k\rangle \rightarrow \langle n_l\rangle \mid I' = I \wedge O' = O \wedge V'(var) \\
&\quad = V(e) \wedge (\forall var' \neq var, V'(var') = V(var'))\} \\
&\Phi(a; a', n_k, n_l, \rightarrow) = \Phi(a, n_k, n_r, \rightarrow) \cup \Phi(a', n_r, n_l, \rightarrow) \\
&\Phi(if\ (e)\ then\ a\ else\ a', n_k, n_l, \rightarrow) \\
&\quad = \{\langle n_k\rangle \rightarrow \langle n_r\rangle \mid I' = I \wedge O' = O \wedge V' = V \wedge e\} \\
&\quad\ \cup \{\langle n_k\rangle \rightarrow \langle n_t\rangle \mid I' = I \wedge O' = O \wedge V' = V \wedge \neg e\} \\
&\quad\ \cup \Phi(a, n_r, n_l, \rightarrow) \cup \Phi(a', n_t, n_l, \rightarrow) \\
&\Phi(while\ (e)\ a, n_k, n_l, \rightarrow) \\
&\quad = \{\langle n_k\rangle \rightarrow \langle n_r\rangle \mid I' = I \wedge O' = O \wedge V' = V \wedge e\} \\
&\quad\ \cup \{\langle n_k\rangle \rightarrow \langle n_l\rangle \mid I' = I \wedge O' = O \wedge V' = V \wedge \neg e\} \\
&\quad\ \cup \Phi(a, n_r, n_k, \rightarrow).
\end{aligned}
$$

3.2.2. Model preprocessing

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 μ_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 μ_0 and μ_0', for $\forall li_{i,x} \in LIn_i$, let $I_{\mu_0'}(li_{i,x}) := I_{\mu_0}(li_{i,x})$, which means the values of low-level inputs of two models are equal in the initial state.

3.2.3. Properties modeling

For $\mu = (I, O, V)$, $\mu' = (I', O', V')$, $\mu \approx_L^{In} \mu'$ means that for $\forall li \in LIn_i$, there is $I(li) = I'(li)$. $\mu \approx_L^{Out} \mu'$ means that for $\forall lo \in LOut_i$, there is $O(lo) = O'(lo)$. 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 $\forall \mu_0$, μ_0', when $\mu_0 \approx_L^{In} \mu_0'$, there is $\mu_e \approx_L^{Out} \mu_e'$, where $\mu_e = F_i(\mu_0)$, $\mu_e' = F_i(\mu_0')$.*

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

For *each $lo_{i,x} \in LOut_i$*

$$
\textbf{\textit{assert}} \left(\bigwedge_{0 \le x \le |LOut_i|} (O_{\mu_e}(lo_{i,x}) == O_{\mu_e'}(lo_{i,x})) \right).
$$

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 Ce_i signed by the local SA is generated for the compositional verification. Ce 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 id, input In_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 a off-line way.

3.3. Compositional verification

In SoCS, s_{ch} is dynamically composed by different candidate service components $s_{i,j} \in S_i$ 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.

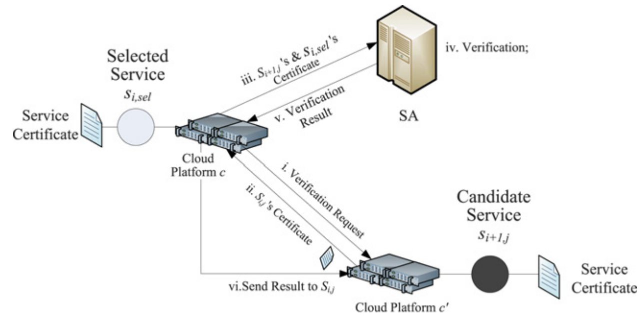


Fig. 6. Compositional verification procedure.

Algorithm 1 *Secure_Chain_Composition*()

Input: Selected Service $s_{i,sel}$, Successor's Candidate Service Set S_{i+1}.
Output: Current Secure Execution Path P

```
SA waits for the message to start the verification work
if The message is start message then
    SA pushes selected service s_{i,sel} into Execution Path P.
    if s_{i,sel} is the final step of service chain then
        SA sends success message and the secure execution path P to user.
    else
        SA requests each candidate service's cert, and validates them where illegal one
        is dropped.
        SA verifies the information flow between s_{i,sel} and each candidate service s_{i+1,j}
        based on the theorem 2.1.
        The passed services are push into passed candidate service set S^p_{i+1}.
    end if
    if There is no passed services then
        SA sends fail message to its predecessor's SA.
    end if
    SA sends start message to each passed service s_{i+1,k} 's SA.
end if
if The message is fail message then
    SA's failure counter increases.
    if failure counter equals to the number of the candidate services in S^p_{i+1} then
        if i!=0 then
            SA sends fail message to its predecessor's SA.
        else
            SA notices the user that there is no secure service execution path.
        end if
    end if
end if
```

---

s_i requests $s_{i+1,j}$'s certificate first, and sends it with $s_{i,sel}$'s certificate to s_i's SA, i.e. SA_c. During the verification, SA_c checks the signature of $s_{i+1,j}$'s SA first, and the illegal one will be dropped. Then SA_c verifies whether the information flow between s_i and $s_{i+1,j}$ 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_{i+1} and obtains the validated candidate service set S^p_{i+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*

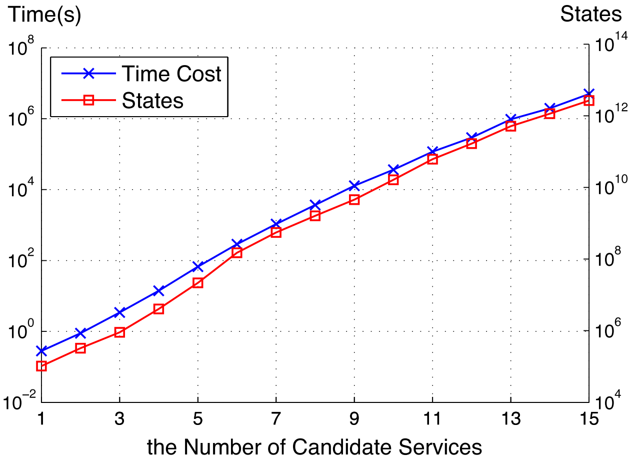


Fig. 7. Verification cost by global model checking.

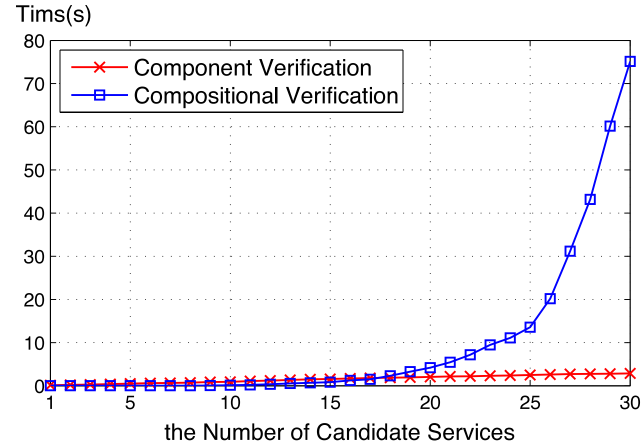


Fig. 8. Time cost of different phases in our approach.

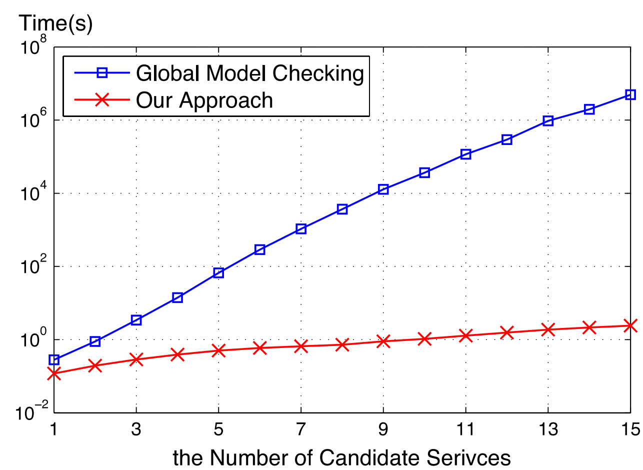


Fig. 9. Time cost by global model checking and our approach.

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_1^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 $O(n * m)$, where $m(m \leq n)$ is the number of the validated service set S_{i-1}^p. Therefore, the complexity of the compositional verification is $O(n * m * k)$, where k is the number of steps in the composite service. Besides, the time complexity for component verification is $O(n * k)$. Therefore, the time complexity of the whole verification is $O(n * k) + O(n * m * k)$. Meanwhile, the time complexity of global model checking is $O(n^k)$. 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

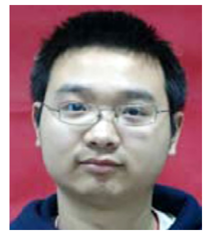
In this paper, we propose a distributed secure service composition approach with information flow control in service clouds. For the dynamic composition of different services, our approach first verifies each service component by model checking, and then ensures the information flow security during the 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 NSFC-Guangdong Union Foundation (U1135002), National Natural Science Foundation of China (61303033), Aviation Science Foundation of China (2013ZC31003, 20141931001).

References

[1] C. Li, Z. Deng, Value of cloud computing by the view of information resources, in: 2011 International Conference on Network Computing and Information Security, NCIS, Vol. 1, 2011, pp. 108–112.
[2] Y. Wei, M. Blake, Service-oriented computing and cloud computing: challenges and opportunities, IEEE Internet Comput. 14 (6) (2010) 72–75.
[3] B. Benatallah, Q. Sheng, M. Dumas, The self-serv environment for Web services composition, IEEE Internet Comput. 7 (1) (2003) 40–48.
[4] T. Yu, Y. Zhang, K.-J. Lin, Efficient algorithms for Web services selection with end-to-end QoS constraints, ACM Trans. Web 1 (1) (2007) 1–26.
[5] C.-W. Hang, M.P. Singh, Trustworthy service selection and composition, ACM Trans. Auton. Adapt. Syst. 6 (1) (2011) 1–17.
[6] H. Takabi, J. Joshi, G.-J. Ahn, Security and privacy challenges in cloud computing environments, IEEE Secur. Privacy 8 (6) (2010) 24–31.
[7] E. Bertino, A. Squicciarini, D. Mevi, A fine-grained access control model for Web services, in: Services Computing, 2004, (SCC 2004). Proceedings. 2004 IEEE International Conference on, IEEE, 2004, pp. 33–40.
[8] R. Bhatti, E. Bertino, A. Ghafoor, A trust-based context-aware access control model for Web services, in: Web Services, 2004. Proceedings. IEEE International Conference on, IEEE, 2004, pp. 184–191.
[9] D. Hutter, M. Volkamer, Information flow control to secure dynamic Web service composition, in: J. Clark, R. Paige, F. Polack, P. Brooke (Eds.), Security in Pervasive Computing, in: Lecture Notes in Computer Science, vol. 3934, Springer, Berlin, Heidelberg, 2006, pp. 196–210.
[10] N. Xi, J. Ma, C. Sun, Y. Shen, T. Zhang, Distributed information flow verification framework for the composition of service chain in wireless sensor network, Int. J. Distrib. Sens. Netw. 2013 (2013) 10.
[11] S. Nakajima, Model-checking of safety and security aspects in Web service flows, in: Web Engineering, in: Lecture Notes in Computer Science, vol. 3140, Springer, Berlin, Heidelberg, 2004, pp. 488–501.
[12] S. Rossi, Model checking adaptive multilevel service compositions, in: Formal Aspects of Component Software, in: Lecture Notes in Computer Science, vol. 6921, Springer, Berlin, Heidelberg, 2012, pp. 106–124.
[13] W. She, I.-L. Yen, B. Thuraisingham, E. Bertino, Security-aware service composition with fine-grained information flow control, IEEE Trans. Serv. Comput. 6 (3) (2013) 330–343.
[14] W. She, I.-L. Yen, B. Thuraisingham, S.-Y. Huang, Rule-based run-time information flow control in service cloud, in: 2011 IEEE International Conference on Web Services, ICWS, 2011, pp. 524–531.
[15] C. Sun, L. Tang, Z. Chen, Secure information flow by model checking pushdown system, in: Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing, 2009, UIC-ATC'09, 2009, pp. 586–591.
[16] J.A. Goguen, J. Meseguer, Security policies and security models, in: Security and Privacy, 1982, IEEE Symposium on, IEEE, 1982, pp.11–20.
[17] G. Barthe, P. D'Argenio, T. Rezk, Secure information flow by self-composition, in: Computer Security Foundations Workshop, 2004, Proceedings, 17th IEEE, 2004, pp. 100–114.
[18] G.J. Holzmann, The SPIN Model Checker: Primer and Reference Manual. Vol. 1003, Addison-Wesley Reading, 2004.
[19] S. Farrell, R. Housley, An Internet attribute certificate profile for authorization, in: Internet RFC328l.
[20] C. Groba, S. Clarke, Opportunistic composition of sequentially-connected services in mobile computing environments, in: Web Services (ICWS), 2011 IEEE International Conference on, IEEE, 2011, pp. 17–24.
[21] T.R. Henderson, S. Roy, S. Floyd, G.F. Riley, NS-3 project goals, in: Proceeding from the 2006 Workshop on NS-2: The IP Network Simulator, WNS2'06, ACM, New York, NY, USA, 2006.



Ning Xi received the B.S. and M.S. degree in Computer Science and Technology from Xidian University, China in 2008 and 2011. Now he is a Ph.D. student in school of Computer Science and Technology, Xidian University. His major research is in heterogeneous network convergence, service computing and network security.

Cong Sun received the B.S. degree in computer science from Zhejiang University, in 2005, and the Ph.D. degree in computer science from Peking University, in 2011. He is currently an associate professor in the School of Computer Science at Xidian University. His research interests include information flow security and program analysis.

Jianfeng Ma received the M.S. and Ph.D. degree in Xidian University, China in 1989 and 1995 separately. He has been a professor in Department of Computer Science and Technology, Xidian University since 1998. He was also the special engaged professor of the Yangtze River scholar in China. He is currently working on the heterogeneous wireless networks convergence, service computing, wireless network security, the survivability of network system and so on. He is the leader of the Key Program of NSFC-Guangdong Union Foundation, National Natural Science Foundation of China and Major national S & T program.

Yulong Shen received the Ph.D. degree in Xidian University, China in 2008. He has been an associate professor in Department of Computer Science and Technology, Xidian University. He is the senior member of the China Computer Federation. His major research is in the wireless network convergence, network security and so on.