Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Preserving privacy information flow security in composite service evolution*

Huan-feng PENG[†1,2], Zhi-qiu HUANG[†‡1], Lin-yuan LIU[3], Yong LI[1], Da-juan FAN[2], Yu-qing WANG[4]

[1]*College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China*

[2]*College of Computer Engineering, Nanjing Institute of Technology, Nanjing 211167, China*

[3]*College of Technology, Nanjing Audit University, Nanjing 210029, China*

[4]*Software Engineering Research Unit, University of Oulu, Oulu 90029, Finland*

[†]E-mail: penghf@njit.edu.cn; zqhuang@nuaa.edu.cn

Received June 6, 2017; Revision accepted Dec. 3, 2017; Crosschecked May 10, 2018

Abstract: After a composite service is deployed, user privacy requirements and trust levels of component services are subject to variation. When the changes occur, it is critical to preserve privacy information flow security. We propose an approach to preserve privacy information flow security in composite service evolution. First, a privacy data item dependency analysis method based on a Petri net model is presented. Then the set of privacy data items collected by each component service is derived through a privacy data item dependency graph, and the security scope of each component service is calculated. Finally, the evolution operations that preserve privacy information flow security are defined. By applying these evolution operations, the re-verification process is avoided and the evolution efficiency is improved. To illustrate the effectiveness of our approach, a case study is presented. The experimental results indicate that our approach has high evolution efficiency and can greatly reduce the cost of evolution compared with re-verifying the entire composite service.

Key words: Composite service; Privacy information flow security; Service evolution; Petri net
https://doi.org/10.1631/FITEE.1700359 **CLC number:** TP309

1 Introduction

Service-oriented architecture (SOA) has become an effective approach to implement loosely coupled, flexible, and interoperable service-oriented systems. Initially, researchers were concerned with the earlier phases in the service life cycle, especially the related issues of service composition, such as formal modeling of composite services, analysis, and validation of the related properties. Services are subject to changes with respect to the changing organizational and regulatory policies (Wang and Wang, 2013). In recent years, studies concerning the challenge issues in service evolution have increased in number (Andrikopoulos et al., 2012).

In traditional software applications, users have full control over their privacy data. However, when the privacy data are collected by composite services, the internal operations usually become transparent for users (Liu et al., 2011). Recently, many studies have been devoted to controlling the propagation of privacy information with the information flow

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (Nos. 61562087 and 61772270), the National High-Tech R&D Program (863) of China (No. 2015AA015303), the Natural Science Foundation of Jiangsu Province, China (No. BK20130735), the Universities Natural Science Foundation of Jiangsu Province, China (No. 13KJB520011), and the Science Foundation of Nanjing Institute of Technology, China (No. YKJ201420)

ORCID: Zhi-qiu HUANG, http://orcid.org/0000-0001-6843-1892

control (IFC) technology in the domain of service computing. The usage of the IFC technology by composite service providers will increase users' confidence so that their privacy data are used correctly (Bacon et al., 2014). However, the control flow, data flow, user privacy requirement, and trust levels of component services may change after a composite service is deployed. Preserving privacy information flow security in a composite service becomes an important evolutionary requirement.

In this study, we focus on how to preserve privacy information flow security in a composite service when user privacy requirements and trust levels of component services change. To address this issue, a possible approach is to re-verify the entire composite service. However, this approach incurs great evolutionary cost. To avoid the complex re-verification process and decrease the evolutionary cost, a better approach is to first identify the impact of the changes, and then enforce the evolution according to the impact identified. However, existing studies gave little attention to the evolution issues involved in privacy information flow control mechanisms. Based on the above discussion, we propose an evolution approach to preserve privacy information flow security in composite services. To give support to the subsequent evolution, when we initially verify the privacy information flow security, the privacy data dependencies are analyzed. Then for each component service, the privacy dataset it collects is derived from a privacy data item dependency graph, and its security scope is calculated. Finally, evolution operations are proposed for the changes of user privacy requirements and trust levels of component services. By applying these evolution operations, the complex re-verification process is avoided and the evolution efficiency is improved.

2 Related work

In this section, we discuss related studies on information security verification, change impact analysis, and approaches that preserve relevant properties during the process of evolution.

Since the IFC technology can control the legal propagation of information according to security policies such as the no-read-up and no-write-down rules of the Bell-LaPadual model (Bell and LaPadula, 1973), it has been considered as a promising approach to ensure information security. Knorr (2001) used a workflow net to model the information and control flow of business processes, and verified information security through a coverability graph augmented with objects' security levels. However, the coverability graphs will grow rapidly with the number of places and transitions in workflow net models. Zeng et al. (2016) proposed a dynamic flow-sensitive security model containing the Bell-Lapadula rules and cloud security rule for federated cloud systems. Colored Petri nets were used to represent the security model, and information security was verified by using existing Petri net techniques. Accorsi et al. (2015) used Petri net reachability to verify information security in business processes based on the concept of place-based non-interference and declassification. A lattice model was employed to describe a security policy, and the verification work could be performed automatically. Note that Accorsi et al. (2015) focused on the examination of whether there was any information leakage between the multiple instances of a business process. Xi et al. (2015) proposed a distributed secure service composition approach to comply with a security policy formalized by a security lattice. This can reduce the cost of verification compared with the global verification approach. However, this study focused only on a service chain, and more complicated composite services with the conditional and loop structure were not considered. In brief, the studies mentioned above did not consider user privacy requirements. Users may define release constraints on the combination of privacy data items in their privacy requirements. Therefore, the privacy information flows in a composite service are necessarily required to comply with both security policies and user privacy requirements. To address this issue, we propose a static analysis method for securing privacy information flows in composite services according to the characteristics of privacy protection in our previous study (Peng et al., 2017).

Impact analysis is one of the key topics in the field of service evolution, which evaluates the change effects and provides evolution strategies to reduce risks and maintenance costs (Alam et al., 2015). Fokaefs et al. (2011) performed an empirical study on the evolution of services and discussed how the changes of the web services description language (WSDL) files could potentially affect client applications. By analyzing the changes in different versions

of five real-world services, they draw the conclusion that services are usually expanded rather than being changed or having their elements removed. Qi et al. (2012) proposed a set of evolution operations and analyzed the trust impact of these operations by a trust dependency graph and a control flow graph. Three metrics were proposed to quantify the impact degree on the trust of component services and the composite service. Wang and Capretz (2011) proposed an approach to evaluate the change effects on a service-oriented system based on dependency analysis and information entropy. Dependency analysis was first used to measure the relative importance of each component service. Then the change effects were evaluated by combining information entropy with dependency analysis, which could facilitate decisions for service-oriented system evolution. Wang et al. (2012) focused on the analysis of dependencies between services and their supporting business processes. A set of change impact patterns was defined to analyze the change propagations. The proposed algorithms could determine the impact scopes affected by the service and process changes. Wang and Capretz (2009) proposed an impact analysis model by constructing the intra-service and inter-service relation matrices. Thus, the dependencies and impact effects could be measured. The above-mentioned studies on impact analysis focused mainly on the interface, behavior, and non-function changes (e.g., trust and quality of service (QoS)). However, change impact analysis for privacy information flow security has not been addressed in existing studies.

How to ensure the relevant properties during evolution is a more challenging research issue. van der Aalst (1997) proposed eight basic transformation rules to modify a sound process. The resulting process kept soundness by applying these rules, which could make the modification more efficient than reverifying the entire process. Similarly, Zeng et al. (2010) proposed a set of change operations including replacement, addition, deletion, and process structural adjustments, and the soundness of a new process could be satisfied after applying these change operations. Song et al. (2010) focused on the problem of how to preserve data flow correctness for process adaptation, and proposed three criteria for process adaptation operations. Data flow correctness of the adapted process could be preserved with these criteria. However, this study assumed that each task has one output at most, which limits practical applications. In brief, an important feature of the existing studies is that these approaches were proposed to meet a specific property of systems. However, the studies on evolution approaches for preserving information security are generally neglected.

The challenging issues in service evolution have been studied actively in recent years. However, existing studies focused mainly on the interface, behavior, and non-function (e.g., trust and QoS) changes. In this study, we pay special attention to the issue of how to preserve privacy information flow security in composite service evolution, which is generally neglected in the existing studies.

3 Preliminaries

To clearly present our approach, we introduce a lattice model and privacy workflow nets used to analyze data dependencies in composite services.

3.1 A lattice for secure information flow

Information security is generally related to the important aspects of confidentiality, integrity, and availability (Bishop, 2002). In the domain of service computing, privacy protection focuses mainly on the confidentiality of user privacy information. We can formalize a privacy information flow security policy with a lattice proposed by Denning (1976).

Definition 1 (Lattice) A lattice is defined by a tuple $(\mathrm{SC}, \rightarrow)$, where SC denotes the finite set of security classes and $\rightarrow$ represents the partial order relation defined on pairs of security classes.

In what follows, we assume $\mathrm{SC} = \{N, L, M, H, \mathrm{TH}\}$. For all $\mathrm{sc}, \mathrm{sc}' \in \mathrm{SC}$, there exist a least upper bound operator $\oplus$ and a greatest lower bound operator $\otimes$. $(\mathrm{SC}, \rightarrow)$ formalizes a security policy of privacy information flows from the perspective of confidentiality, which allows the no-read-up and no-write-down rules to be implemented.

Let P denote the set of privacy data items and O denote the power set of P. The set of component services is denoted by S. Note that $s \in S$ and $o \in O$ are assigned security classes in SC by a trust level map $L_{\mathrm{tl}} : S \rightarrow \mathrm{SC}$ and a sensitivity level map $L_{\mathrm{sl}} : O \rightarrow \mathrm{SC}$.

3.2 Privacy workflow net

The Petri net, a graphical language for modeling and validating the concurrent and distributed systems, provides rich analysis techniques and tools. By using Petri nets as a formalism, Tan et al. (2009) analyzed the compatibility and mediation issues in service composition. Yu et al. (2014) validated the rationality and the transaction consistency of e-commerce business processes. Yu et al. (2016) detected malicious behaviors in the online shopping business processes.

To ensure privacy information security, we need to analyze data dependencies. For dependency analysis in service-based systems, formal dependency modeling, especially Petri nets based formalism, is more effective compared with directed graph based dependency analysis (Alam et al., 2015). Workflow nets (WF-nets) as a class of Petri nets have become one of the standard ways to model and analyze business processes (Liu C et al., 2016; Liu G et al., 2016). Compared with other formalisms such as automata and process algebra, there exists an open source tool BPEL2oWFN (Lohmann et al., 2006), which provides a feature-complete mapping of the BPEL (a standard language for web service orchestration) process's control flow to WF-nets. We can extend this tool to make it support modeling data flow. By extending the classical WF-net with data places (read arcs and write arcs), Peng et al. (2017) proposed a privacy workflow net (PWF-net) that had the capability to model both control and data flows in composite services. Therefore, we use PWF-nets as a formalism to analyze data dependencies in composite services.

Definition 2 (PWF-net) A tuple (P_c, T, F, P_d, R, W, T_S, T_A) is a PWF-net if and only if:

1. (P_c, T, F) is a WF-net, with the set of control places P_c, the set of transitions T, and the set of arcs $F \subseteq P_\mathrm{c} \times T \cup T \times P_\mathrm{c}$;

2. P_d is a set of data places;

3. $P_\mathrm{c} \cup P_\mathrm{d} \cup T \neq \varnothing \wedge P_\mathrm{c} \cap P_\mathrm{d} = \varnothing \wedge P_\mathrm{c} \cap T = \varnothing \wedge P_\mathrm{d} \cap T = \varnothing$;

4. $R \subseteq P_\mathrm{d} \times T$ is a set of read arcs used only in connecting data places and transitions;

5. $W \subseteq T \times P_\mathrm{d}$ is a set of write arcs used only in connecting transitions and data places;

6. T_S: $T \to$ SUBJECT is the subject labeling function. SUBJECT denotes the set of subjects involved in a composite service such as component services, the composite service, and the user;

7. T_A: $T \to$ ACTION is the action labeling function. ACTION = {RECV, SND, ASGN, STRC}, where the first three elements denote message receiving, message sending, and assign activity, respectively, and STRC denotes auxiliary transitions used to model the structural activities.

The read set of t is denoted by $r(t) = \{d|(d,t) \in R\}$, and the write set of t is denoted by $w(t) = \{d|(t,d) \in W\}$.

Definition 3 (Control view) Given a PWF-net PN = (P_c, T, F, P_d, R, W, T_S, T_A), the control view of PN is defined by $\mathrm{PN}_\mathrm{cv} = (P_\mathrm{c}, T|_{P_\mathrm{c}}, F)$, where $T|_{P_\mathrm{c}}$ is the set of transitions connecting the control places.

Definition 4 (Data view) Given a PWF-net PN = (P_c, T, F, P_d, R, W, T_S, T_A), the data view of PN is defined by $\mathrm{PN}_\mathrm{dv} = (P_\mathrm{d}, T|_{P_\mathrm{d}}, R, W)$, where $T|_{P_\mathrm{d}}$ is the set of transitions connecting the data places.

Definition 5 (Path) Let $M_\mathrm{i} \xrightarrow{\sigma} M_\mathrm{o}$ denote that the firing sequence $\sigma = \{t_1, t_2, \ldots, t_n\}$ leads from initial state M_i to end state M_o, where σ is called one path from M_i to M_o.

4 Underlying data acquisition method

To give support to the subsequent evolution, when we initially verify the privacy information flow security, the underlying data for evolution should be recorded. In this section, we present the underlying data acquisition method.

4.1 Privacy requirement description

Privacy data items can be divided into two categories. The privacy data item provided by users in the interactions with a composite service is referred to as a direct privacy data item. The data item newly generated during the execution of a composite service is referred to as an indirect privacy data item, which can indirectly expose user privacy information.

In the enforcement of the privacy information security policy, the sensitivity levels of direct privacy data items are specified in user privacy requirements.

Definition 6 (Privacy rule) A privacy rule is defined by $r = (\mathrm{Ds}, \mathrm{sc})$, where Ds is a finite set of direct privacy data items, and $L_\mathrm{sl}(\mathrm{Ds}) = \mathrm{sc}$, $\mathrm{sc} \in \mathrm{SC}$.

One rule specifies the sensitivity level of single or multiple direct privacy data items. In general, for

Ds, the stricter the usage constraint is, the higher the sensitivity level is. Moreover, when users release multiple privacy data items together, they worry more about privacy information leakage than releasing a single data item. Therefore, the usage constraint on the combination of privacy data items will be stricter. For example, a user defines privacy rules $r_1 = (\{\text{name}\}, H)$, $r_2 = (\{\text{phone}\}, H)$ for the single release of name and phone, and $r_3 = (\{\text{name, phone}\}, \text{TH})$ for releasing name and phone together.

Definition 7 (Privacy requirement) The user privacy requirement is defined by $\text{PR} = \{r_1, r_2, \ldots, r_n\}$, which is a finite set of privacy rules.

4.2 Declassification policy

Definition 8 (Component service) A component service is defined by $s = (\text{name, sc})$.

The element name uniquely identifies one component service, and sc is the trust level stated by s. $L_{\text{tl}}(s)$ can be statically bounded to s.sc. However, to comply with the security policy, the sensitivity level of output data d will be s.sc, even if s has never collected any privacy data. Let s.Hds denote the set of privacy data items s has ever collected. In fact, $L_{\text{sl}}(d)$ is determined by s.Hds. Therefore, when $L_{\text{sl}}(d)$ is calculated, $L_{\text{tl}}(s)$ should be dynamically bounded to $L_{\text{sl}}(s.\text{Hds})$. This can be regarded as a declassification policy for component services. If s has never collected any privacy data, $L_{\text{tl}}(s)$ is N. As long as the operations of services comply with the security policy, $L_{\text{sl}}(s.\text{Hds})$ is equal to or lower than s.sc. Note that when the operations that release privacy data to s are checked, $L_{\text{tl}}(s)$ is still s.sc.

4.3 Dependency analysis rule

The sensitivity levels of direct privacy data items are specified in user privacy requirements, while the sensitivity level of an indirect privacy data item is determined by the direct ones on which it depends. Many information flow analysis mechanisms can be used to derive data dependencies (She et al., 2011; Xi et al., 2015). However, most of the existing mechanisms are insufficient when they are used for the privacy protection scenario on which this study focuses. Let d denote an indirect privacy data item newly generated or updated during the execution of a composite service and $\text{Dep}(d)$ denote the set of privacy data items on which d depends. We present our dependency analysis rules as follows:

DAR_1: $T_{\text{A}}(t) = \text{ASGN} \Rightarrow \text{Dep}(d) = r(t)$, with $w(t) = \{d\}$;

DAR_2: $T_{\text{A}}(t) = \text{RECV} \Rightarrow \forall\, d \in w(t)$, $\text{Dep}(d) = T_{\text{S}}(t).\text{Hds}$;

DAR_3: $T_{\text{A}}(t) = \text{SND} \Rightarrow T_{\text{S}}(t).\text{Hds} = T_{\text{S}}(t).\text{Hds} \cup r(t)$;

DAR_4: $\forall\ d_i, d_j,\ \exists\ d_k, d_k \in \text{Dep}(d_i) \wedge d_j \in \text{Dep}(d_k) \Rightarrow \text{Dep}(d_i) = \text{Dep}(d_i) \cup \{d_j\}$.

Rule DAR_1 is used to analyze the transitions of type ASGN which correspond to assign activities. For fine-grained analysis of privacy information flows, one assign activity should be transformed to multiple transitions; i.e., each transition outputs only one indirect privacy data item. Therefore, $\text{Dep}(d) = r(t)$ and $w(t) = \{d\}$, where $r(t)$ and $w(t)$ denote the read and write sets of t, respectively.

Rule DAR_2 is used to analyze the transitions of type RECV which correspond to receive activities. According to the declassification policy, $\forall\, d \in w(t)$, $\text{Dep}(d) = T_{\text{S}}(t).\text{Hds}$, where $T_{\text{S}}(t)$ denotes the component service corresponding to t, and $T_{\text{S}}(t).\text{Hds}$ denotes the privacy data items $T_{\text{S}}(t)$ has ever collected. We assume that users do not generate indirect privacy data items. Thus, if a composite service receives data from users (i.e., $T_{\text{S}}(t) = \text{user}$), there is no need to apply rule DAR_2.

Rule DAR_3 is used to analyze the transitions of type SND which correspond to reply and one-way invoke activities. Both $\text{reply}(s, v)$ and $\text{invoke}(s, v)$ essentially send the content of variable v to component service s. Thus, we need to update the set of privacy data items s has ever collected, i.e., $T_{\text{S}}(t).\text{Hds} = T_{\text{S}}(t).\text{Hds} \cup r(t)$. Since users are trusted entities, there is no need to analyze t using rule DAR_3 when $T_{\text{S}}(t) = \text{user}$. For request-response $\text{invoke}(s, v_1, v_2)$, it first sends the content of variable v_1 to s, and then outputs variable v_2. It can be modeled with two sequential transitions of type SND and RECV, and rules DAR_3 and DAR_2 are applied, respectively.

Rule DAR_4 derives the dependencies caused by transitivity. If d_i is dependent on d_k and d_k is in turn dependent on d_j, d_i is also dependent on d_j.

Let $\text{DDep}(d)$ denote the set of direct privacy data items on which d depends. According to rule DAR_4, $\text{DDep}(d)$ can be derived from $\text{Dep}(d)$. Furthermore, we can easily draw the conclusion that $L_{\text{sl}}(d) = L_{\text{sl}}(\text{Dep}(d)) = L_{\text{sl}}(\text{DDep}(d))$.

4.4 Dataset acquisition

To give support to the subsequent evolution, we need to obtain the set of direct privacy data items each component service collects when verifying privacy information flow security.

Definition 9 (Direct dependency) A direct dependency dDep is a tuple (d, Ds, t), where d is an indirect privacy data item, Ds is a finite set of privacy data items, and t is called associated transition with its input Ds and output d. A direct dependency can be denoted by $d \xrightarrow{t} \mathrm{Ds}$.

Definition 10 (Dependency chain) Let $\mathrm{chainD} = \{\mathrm{dDep}_1, \mathrm{dDep}_2, \ldots, \mathrm{dDep}_n\}$ be a sequence of direct dependencies. For any two adjacent dependencies dDep_i and dDep_{i+1} $(1 \le i \le n-1)$, if $d_{i+1} \in \mathrm{Ds}_i$, chainD forms a dependency chain.

Definition 11 (Indirect dependency) Let $\mathrm{chainD} = \{\mathrm{dDep}_1, \mathrm{dDep}_2, \ldots, \mathrm{dDep}_n\}$ be a dependency chain. For any two direct dependencies dDep_i and dDep_j $(i < j)$, we can say that d_i indirectly depends on Ds_j, which can be denoted by $d_i \xrightarrow{*} \mathrm{Ds}_j$.

A privacy data item dependency graph (PDIDG) can be constructed through the dependency analysis rules proposed in Section 4.3.

Definition 12 (PDIDG) A privacy data item dependency graph is a tuple (V, E, L), where

1. V is a set of vertices;
2. E is a set of edges;
3. $L : E \to T$ is the transition labeling function, where T is the set of transitions in a PWF-net.

The vertices with zero outdegree represent direct privacy data items, while the vertices with nonzero outdegree represent indirect privacy data items. The edge can be denoted by (v_i, v_j, t_m), $v_i, v_j \in V$, $t_m \in T$. This means that v_i depends on v_j through transition t_m. Fig. 1 shows an example of a PDIDG, where d_1, d_2, d_3, and d_4 represent direct privacy data items, and d_5, d_6, and d_7 represent indirect ones. The dependency relationships are as follows:

Direct dependencies: $d_6 \xrightarrow{t_2} \{d_5, d_3\}$, $d_5 \xrightarrow{t_1} \{d_1, d_2\}$, $d_7 \xrightarrow{t_3} \{d_3, d_4\}$;

Dependency chains: $d_6 \xrightarrow{t_2} \{d_5, d_3\}$, $d_5 \xrightarrow{t_1} \{d_1, d_2\}$;

Indirect dependencies: $d_6 \xrightarrow{*} \{d_1, d_2\}$.

When we initially verify the privacy information flow security statically, all possible paths of a composite service should be analyzed. Let $D(s)$, $D^p(s)$, and $D^{p,t}(s)$ denote the sets of direct privacy data items that are collected by component service s, path p, and transition t, respectively. Then the calculation of $D^p(s)$ and $D(s)$ can be respectively expressed as follows:

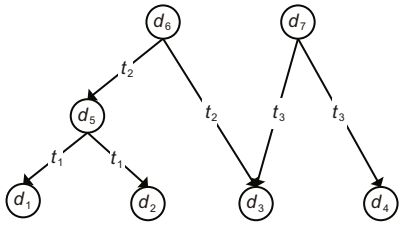

Fig. 1 An example of a privacy data item dependency graph

$$D^p(s) = \bigcup_{i=1}^{n} D^{p,t_i}(s), \tag{1}$$

$$D(s) = \bigcup_{i=1}^{n} D^{p_i}(s). \tag{2}$$

The steps of calculating $D^p(s)$ are listed below:

Step 1: The privacy data item dependency graph G is initialized based on the user privacy requirement. G contains only direct privacy data item vertices after initialization.

Step 2: $\forall\, s \in S$, $D^p(s) = \varnothing \wedge s.\mathrm{Hds} = \varnothing$.

Step 3: Let V denote the set of vertices in current G and d denote the indirect privacy data item newly generated or updated. Each transition t in path p needs to be processed in turn.

1. As mentioned in Section 4.3, if the type of t is ASGN (i.e., $T_{\mathrm{A}}(t) = \mathrm{ASGN}$), the write set $w(t)$ contains only one indirect privacy data item.

(1) If $d \notin V$, the corresponding vertex and dependency edges (d, d_i, t), $\forall\, d_i \in r(t)$ are created.

(2) If $d \in V$, the firing of transition t redefines d, and the relevant dependencies need to be updated. Suppose transition t' has defined d before t. All existing dependency edges (d, d_j, t'), $\forall\, d_j \in r(t')$ should be removed first, and then new dependencies should be created.

For example, the dependency $d_6 \xrightarrow{t_2} \{d_5, d_3\}$ already exists in Fig. 1. There exist two dependency edges: (d_6, d_5, t_2) and (d_6, d_3, t_2). This shows that the indirect privacy data item d_6 has been defined by transition t_2. Now d_6 is redefined by t. The edges (d_6, d_5, t_2) and (d_6, d_3, t_2) need to be removed, and the new edge (d_6, d_4, t) needs to be added. The updated graph is shown in Fig. 2.

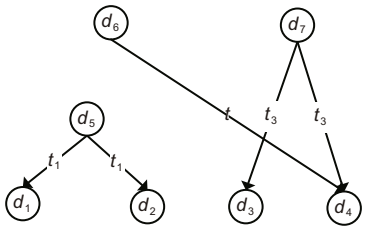

Fig. 2 An updated privacy data item dependency graph

2. If the type of t is RECV (i.e., $T_{\mathrm{A}}(t) = \mathrm{RECV}$), the write set $w(t)$ may contain multiple indirect privacy data items. The read set $r(t)$ is $T_{\mathrm{S}}(t)$.Hds. In this case, each indirect privacy data item in $w(t)$ should be processed as described above.

3. If the type of t is SND (i.e., $T_{\mathrm{A}}(t) = \mathrm{SND}$), $T_{\mathrm{S}}(t)$.Hds needs to be updated, i.e., $T_{\mathrm{S}}(t)$.Hds $= T_{\mathrm{S}}(t)$.Hds $\cup\ r(t)$. Let $\mathrm{DDep}(r(t))$ denote the set of direct privacy data items on which $r(t)$ depends. $D^{p,t}(T_{\mathrm{S}}(t))$ is equal to $\mathrm{DDep}(r(t))$, which can be derived from the dependency analysis rules and current G. Then $D^p(T_{\mathrm{S}}(t))$ can be updated by $D^p(T_{\mathrm{S}}(t)) = D^p(T_{\mathrm{S}}(t)) \cup D^{p,t}(T_{\mathrm{S}}(t))$ according to Eq. (1).

Repeat step 3 until all the transitions in path p have been processed.

For each component service s, the resulting $D^p(s)$ is the set of direct privacy data items s collects through path p. Finally, we can obtain $D(s)$ according to Eq. (2) after processing every path. $D(s)$ needs to be calculated and recorded as the underlying data for the subsequent evolution when the privacy information flow security is initially verified.

4.5 Path reduction

The paths can be derived from the control view of a PWF-net. However, concurrency of transitions may lead to the issue of path explosion. To address this issue, we propose a path reduction method in this subsection.

Fig. 3 shows an example of the control view of a PWF-net and Fig. 4 shows the corresponding coverability graph. In Fig. 3, t_3, t_4, and t_5 are concurrent transitions. Then there exist six paths: $p_1 = t_1t_2t_3t_4t_5t_6$, $p_2 = t_1t_2t_3t_5t_4t_6$, $p_3 = t_1t_2t_4t_3t_5t_6$, $p_4 = t_1t_2t_4t_5t_3t_6$, $p_5 = t_1t_2t_5t_3t_4t_6$, and $p_6 = t_1t_2t_5t_4t_3t_6$.

Let $p.T$ denote the set of transitions in p and PE denote the set of paths including the same transitions. For all $p_i,\ p_j \in \mathrm{PE}$, $p_i.T = p_j.T$. The set of paths derived from a coverability graph is denoted by $P = \mathrm{PE}_1 \cup \mathrm{PE}_2 \cup \cdots \cup \mathrm{PE}_n$, where $\forall\, i \neq j$,

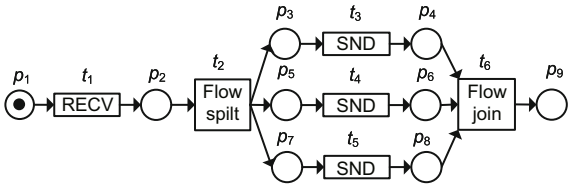

Fig. 3 An example of the control view of a PWF-net

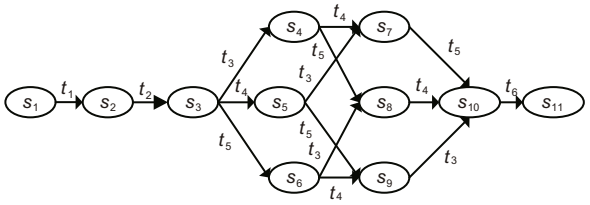

Fig. 4 An example of a coverability graph

$\mathrm{PE}_i \cap \mathrm{PE}_j = \varnothing$. As shown in Fig. 4, all paths in P have the same set of transitions. Then $\mathrm{PE}_1 = \{p_1, p_2, p_3, p_4, p_5, p_6\}$, $P = \mathrm{PE}_1$. The paths in PE have the following property if any two concurrent transitions do not correspond to the same component service.

Property 1 The paths in PE are equivalent for the security verification of privacy information flows.

Proof When verifying the privacy information flow security, we need to analyze the data dependencies according to the security policy proposed in Section 3.1. Because the control and data flow in a composite service are determined, we need only to prove that the data dependencies in the paths of PE are consistent. Let p_i and p_j be two arbitrary paths in PE. The set of concurrent transitions in p_i and p_j is denoted by CT and the set of non-concurrent transitions is denoted by NT. Obviously, the transitions in NT do not lead to inconsistent data dependencies. Let t_i and t_j be two arbitrary concurrent transitions in CT. We assume that t_i and t_j do not correspond to the same component service. The operations on privacy data items can be summarized as follows:

(1) t_i and t_j read or write different privacy data items;

(2) t_i and t_j read the same privacy data item d;

(3) t_i writes privacy data item d, while t_j reads privacy data item d;

(4) t_i and t_j write the same privacy data item d.

Obviously, (1) and (2) will not lead to inconsistent data dependencies in p_i and p_j, while (3) and (4) may lead to data access exception, which should be avoided by establishing a synchronous relationship between t_i and t_j. Note that if t_i and t_j correspond

to the same component service s, the concurrent execution of t_i and t_j may result in inconsistent data dependencies according to rules DAR_2 and DAR_3. Therefore, the paths in PE are equivalent for the security verification of privacy information flows if two arbitrary concurrent transitions correspond to different component services.

Definition 13 (Independent path) An arbitrary path selected from PE is called an independent path.

Definition 14 (Independent path set) An independent path set is defined by $\mathrm{PI} = \{p_i \mid p_i \in \mathrm{PE}_i, 1 \le i \le n\}$, where p_i is the independent path of PE_i, $P = \mathrm{PE}_1 \cup \mathrm{PE}_2 \cup \cdots \cup \mathrm{PE}_n$, and P is the set of paths derived from a coverability graph.

According to Property 1, we can select an arbitrary path from PE as an independent path, and need only to analyze the paths in PI. Thus, the number of paths that need to be analyzed can be greatly reduced, when two arbitrary concurrent transitions do not correspond to the same component service.

5 Evolution operations

5.1 Sensitivity level binding

Let DS denote a set of direct privacy data items. Because users may specify the sensitivity level for the combination of direct privacy data items, the calculation of $L_{\mathrm{sl}}(\mathrm{DS})$ should comply with user privacy requirements. For example, suppose that the user privacy requirement is as follows: $r_1 = (\{\text{email}\}, M)$, $r_2 = (\{\text{name}\}, M)$, and $r_3 = (\{\text{email, name}\}, H)$. Given a set of direct privacy data items DS = {email, name}, $L_{\mathrm{sl}}(\mathrm{DS})$ should not be $L_{\mathrm{sl}}(\{\text{email}\}) \oplus L_{\mathrm{sl}}(\{\text{name}\}) = M \oplus M = M$. Since there exists rule r_3 in the user requirement, $L_{\mathrm{sl}}(\mathrm{DS})$ should be $L_{\mathrm{sl}}(\{\text{email}, \text{name}\}) = H$.

The calculation of $L_{\mathrm{sl}}(\mathrm{DS})$ can be expressed as

$$L_{\mathrm{sl}}(\mathrm{DS}) = L_{\mathrm{sl}}(\mathrm{Ds}_1) \oplus L_{\mathrm{sl}}(\mathrm{Ds}_2) \oplus \cdots \oplus L_{\mathrm{sl}}(\mathrm{Ds}_n), \quad (3)$$

where $\mathrm{Ds}_i \in 2^{\mathrm{DS}} \wedge \mathrm{Ds}_i \in \mathrm{PR\_DSet}$ $(1 \le i \le n)$, 2^{DS} denotes the power set of DS, $\mathrm{PR\_DSet} = \{r_k.\mathrm{Ds} \mid r_k \in \mathrm{PR}, 1 \le k \le |\mathrm{PR}|\}$, and $|\mathrm{PR}|$ is the number of elements in PR, which denotes the user privacy requirement. $L_{\mathrm{sl}}(\mathrm{Ds}_i) = r_k.\mathrm{sc}$, where $\mathrm{Ds}_i = r_k.\mathrm{Ds}$.

5.2 Evolution operations for the trust level changes

The trust levels of component services may change after the deployment of a composite service. To avoid the re-verification process, the security scope of each component service should be identified. As long as the new trust level belongs to the security scope, the security of privacy information flows in a composite service can be preserved.

Theorem 1 For each $s \in S$, if $L_{\mathrm{sl}}(D^p(s)) \to L_{\mathrm{tl}}(s)$, the privacy information flows in path p are secure.

Proof There are four categories of transitions: ASGN, SND, RECV, and STRC. Note that we do not consider the information leakage caused by the control flow in this study. As long as the transitions of type ASGN, SND, and RECV are secure, the privacy information flows in p are secure.

1. Transition t of type ASGN is secure, if $L_{\mathrm{sl}}(r(t)) \to L_{\mathrm{sl}}(d)$. From rule DAR_1, $\mathrm{Dep}(d) = r(t)$. Hence, $L_{\mathrm{sl}}(d) = L_{\mathrm{sl}}(r(t))$ so that $L_{\mathrm{sl}}(r(t)) \to L_{\mathrm{sl}}(d)$. Therefore, t is always secure.

2. Transition t of type RECV is secure, if $L_{\mathrm{sl}}(T_{\mathrm{S}}(t).\mathrm{Hds}) \to L_{\mathrm{sl}}(d)$, $\forall d \in w(t)$, where $T_{\mathrm{S}}(t).\mathrm{Hds}$ denotes the set of privacy data items that component service $T_{\mathrm{S}}(t)$ has ever collected. From rule DAR_2, $\mathrm{Dep}(d) = T_{\mathrm{S}}(t).\mathrm{Hds}$, $\forall d \in w(t)$. Hence, $L_{\mathrm{sl}}(T_{\mathrm{S}}(t).\mathrm{Hds}) = L_{\mathrm{sl}}(d)$ and $L_{\mathrm{sl}}(T_{\mathrm{S}}(t).\mathrm{Hds}) \to L_{\mathrm{sl}}(d)$. Therefore, t is always secure.

3. Transition t of type SND is secure, if $L_{\mathrm{sl}}(r(t) \cup T_{\mathrm{S}}(t).\mathrm{Hds}) \to L_{\mathrm{tl}}(s)$. Let $D(r(t) \cup T_{\mathrm{S}}(t).\mathrm{Hds})$ denote the set of direct privacy data items on which $r(t) \cup T_{\mathrm{S}}(t).\mathrm{Hds}$ depends. Then $L_{\mathrm{sl}}(r(t) \cup T_{\mathrm{S}}(t).\mathrm{Hds}) = L_{\mathrm{sl}}(D(r(t) \cup T_{\mathrm{S}}(t).\mathrm{Hds}))$. $D(r(t) \cup T_{\mathrm{S}}(t).\mathrm{Hds}) \subseteq D^p(s) \wedge L_{\mathrm{sl}}(D^p(s)) \to L_{\mathrm{tl}}(s)$ so that $L_{\mathrm{sl}}(D(r(t) \cup T_{\mathrm{S}}(t).\mathrm{Hds})) \to L_{\mathrm{tl}}(s)$. Therefore, t is secure.

Let SC_s^p denote the security scope of s in path p. The calculation of SC_s^p can be expressed as follows:

$$\mathrm{SC}_s^p = \{\mathrm{sc} \mid L_{\mathrm{sl}}(D^p(s)) \to \mathrm{sc} \wedge \mathrm{sc} \in \mathrm{SC}\}. \quad (4)$$

According to Theorem 1, as long as $L_{\mathrm{tl}}(s) \in \mathrm{SC}_s^p$, the privacy information flows in path p are secure.

Theorem 2 For each $s \in S$, if $L_{\mathrm{sl}}(D(s)) \to L_{\mathrm{tl}}(s)$, the privacy information flows in a composite service are secure.

Proof $\forall p \in P, \forall s \in S, L_{\mathrm{sl}}(D(s)) \to L_{\mathrm{tl}}(s) \wedge D^p(s) \subseteq D(s)$ such that $L_{\mathrm{sl}}(D^p(s)) \to L_{\mathrm{tl}}(s)$. Therefore, from Theorem 1, we know that the privacy information flows in a composite service are secure.

Let SC_s denote the security scope of s. The calculation of SC_s can be expressed as follows:

$$\mathrm{SC}_s = \{\mathrm{sc} \mid L_{\mathrm{sl}}(D(s)) \rightarrow \mathrm{sc} \wedge \mathrm{sc} \in \mathrm{SC}\}. \quad (5)$$

According to Theorem 2, as long as $L_{\mathrm{tl}}(s) \in \mathrm{SC}_s$, the privacy information flows in a composite service are secure.

Evolution operation 1 When the trust level of component service s changes, to preserve privacy information flow security, the new trust level of s should belong to SC_s.

For example, suppose $L_{\mathrm{sl}}(D(s)) = H$; thus, $\mathrm{SC}_s = \{H, \mathrm{TH}\}$. To preserve privacy information flow security, the new trust level of s should be H or TH.

When we initially verify privacy information flow security, $D(s)$ and SC_s of each component service s need to be calculated and recorded. We assume that the control flow, data flow, and user privacy requirements remain unchanged. As long as the new trust level of s belongs to its security scope, the privacy information flows in a composite service are still secure.

5.3 Evolution operations for the requirement changes

Assuming that the set of direct privacy data items remains unchanged, the changes in user privacy requirements can be summarized as follows:

(1) Modifying privacy rule r = (Ds, sc): Ds does not change, and its sensitivity level sc is downgraded;

(2) Modifying privacy rule r = (Ds, sc): Ds does not change, and its sensitivity level sc is upgraded;

(3) Modifying privacy rule r = (Ds, sc): Ds changes, and its security class sc also changes;

(4) Modifying privacy rule r = (Ds, sc): Ds changes, and its security class sc remains unchanged;

(5) Adding privacy rule r = (Ds, sc);

(6) Deleting privacy rule r = (Ds, sc).

Since the privacy information flows are secure before the user requirements change, (1) and (6) do not undermine privacy information flow security. When (2)–(5) happen, we can recalculate $L_{\mathrm{sl}}(D(s))$ and SC_s for all component services, and then verify whether $L_{\mathrm{tl}}(s)$ belongs to SC_s.

Evolution operation 2.1 For each $s \in S$, $L_{\mathrm{sl}}(D(s))$ and SC_s are recalculated. If $L_{\mathrm{sl}}(D(s)) \rightarrow L_{\mathrm{tl}}(s)$ does not hold, to preserve privacy information flow security, $L_{\mathrm{tl}}(s)$ should be upgraded to sc, $\mathrm{sc} \in \{\mathrm{sc} \mid L_{\mathrm{sl}}(D(s)) \rightarrow \mathrm{sc} \wedge \mathrm{sc} \in \mathrm{SC}\}$.

According to Theorem 2, the privacy information flow security can be preserved through Evolution operation 2.1. In fact, the changes in user privacy requirements may affect only part of the component services, so we need only to evolve the component services involved in the changes. Let R_{c} denote the privacy rules in (2)–(5) and D_{c} denote the set of direct privacy data items involved. The calculation of D_{c} can be expressed as follows:

$$D_{\mathrm{c}} = \bigcup_{i=1}^{|R_{\mathrm{c}}|} r_i.\mathrm{Ds}, \ \forall r_i \in R_{\mathrm{c}}. \quad (6)$$

Let S_{c} denote the set of component services involved in the changes; thus, $S_{\mathrm{c}} = \{s \mid D(s) \cap D_{\mathrm{c}} \neq \varnothing \wedge s \in S\}$.

Theorem 3 For each $s \in S_{\mathrm{c}}$, if $L_{\mathrm{sl}}(D(s)) \rightarrow L_{\mathrm{tl}}(s)$, the privacy information flows in a composite service are still secure.

Proof Let S_{nc} denote the set of component services that are not affected by the changes in user requirements. Thus, $S_{\mathrm{nc}} = \{s \mid D(s) \cap D_{\mathrm{c}} = \varnothing \wedge s \in S\}$ and $S = S_{\mathrm{c}} \cup S_{\mathrm{nc}} \wedge S_{\mathrm{c}} \cap S_{\mathrm{nc}} = \varnothing$. The privacy information flows are secure before changes such that $\forall s \in S_{\mathrm{nc}}, \ L_{\mathrm{sl}}(D(s)) \rightarrow L_{\mathrm{tl}}(s)$. Therefore, $\forall s \in S$, $L_{\mathrm{sl}}(D(s)) \rightarrow L_{\mathrm{tl}}(s)$. From Theorem 2, the privacy information flows are still secure.

Evolution operation 2.2 For each $s \in S_{\mathrm{c}}$, $L_{\mathrm{sl}}(D(s))$ and SC_s are recalculated. If $L_{\mathrm{sl}}(D(s)) \rightarrow L_{\mathrm{tl}}(s)$ does not hold, to preserve privacy information flow security, $L_{\mathrm{tl}}(s)$ should be upgraded to sc, $\mathrm{sc} \in \{\mathrm{sc} \mid L_{\mathrm{sl}}(D(s)) \rightarrow \mathrm{sc} \wedge \mathrm{sc} \in \mathrm{SC}\}$.

According to Theorem 3, the privacy information flow security can be preserved through Evolution operation 2.2.

5.4 Discussion

In this study, we propose a set of evolution operations that preserve privacy information flow security. By applying these evolution operations, we can avoid the re-verification process and improve the evolutionary efficiency. When the trust levels of component services change, we can use Evolution operation 1. When the user privacy requirements change, we can use Evolution operation 2.1 or Evolution operation 2.2. For Evolution operation 2.1, we need to recalculate $L_{\mathrm{sl}}(D(s))$ and SC_s for all component services, and its time cost is the highest among these operations. For Evolution operation 2.2, we need

only to recalculate $L_{\mathrm{sl}}(D(s))$ and SC_s for the component services involved in the changes. Obviously, Evolution operation 2.2 is more suitable than Evolution operation 2.1 if the user requirements have small changes. If the trust levels of component services and user privacy requirements change at the same time, we can use Evolution operation 2.1 to deal with this case.

Users may interact with a composite service multiple times, and each time a different path is executed. For example, users release name and phone to component service s through paths p_1 and p_2, respectively, and the result of interactions is that s collects $D(s) = \{\text{name, phone}\}$. Suppose the user privacy requirement is as follows: $r_1 = (\{\text{name}\}, M)$, $r_2 = (\{\text{phone}\}, M)$, $r_3 = (\{\text{name}, \text{phone}\}, H)$, and the trust level of s is $L_{\mathrm{tl}}(s) = M$. Although $L_{\mathrm{sl}}(\{\text{name}\}) \rightarrow L_{\mathrm{tl}}(s) \wedge L_{\mathrm{sl}}(\{\text{phone}\}) \rightarrow L_{\mathrm{tl}}(s)$ holds, $L_{\mathrm{sl}}(\{\text{name}, \text{phone}\}) \rightarrow L_{\mathrm{tl}}(s)$ does not hold, which violates privacy rule r_3. Therefore, privacy information leakage can be prevented more strictly in our approach.

6 Case study and performance analysis

6.1 Case study

In this subsection, a travel agent (TA) is used to illustrate our evolution approach. Through combining three existing and independent services, flight, hotel, and pay, TA can provide one-stop service according to a user's travel plan. Suppose user Bob regards name, phone, id-number, and credit-card-info as direct privacy data items, and the privacy requirement is as follows:

$$\begin{cases} r_1 = (\{\text{name}\}, M), \\ r_2 = (\{\text{phone}\}, M), \\ r_3 = (\{\text{id-number}\}, H), \\ r_4 = (\{\text{credit-card-info}\}, H), \\ r_5 = (\{\text{name, id-number, credit-card-info}\}, \text{TH}). \end{cases}$$

Furthermore, suppose the trust levels of hotel, flight, and pay are M, H, and TH, respectively. Fig. 5 shows the control view of TA's PWF-net model. To simplify the presentation, we assume that the payments are processed together after the completion of flight and hotel booking. We skip the assign activities and the process of querying flight and hotel information in Fig. 5.

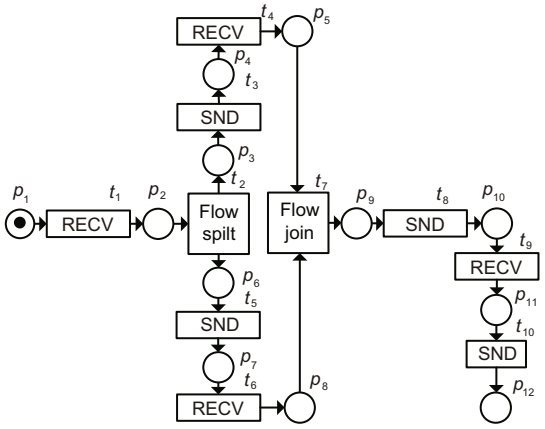

Fig. 5 Control view of the travel agent

$\{t_3, t_4\}$ and $\{t_5, t_6\}$ correspond to hotel and flight booking operations, respectively, and the transitions will execute concurrently with the ones in another group. Since any two concurrent transitions do not correspond to the same component service, the paths that need to be analyzed can be reduced. In this case, we need only to verify $p_1 = t_1t_2t_3t_4t_5t_6t_7t_8t_9t_{10}$ according to the path reduction method proposed in Section 4. Table 1 shows the analysis process of path p_1.

During the static verification of privacy information flow security, we need to record the underlying data for the subsequent evolution. Table 2 shows the underlying data.

According to our approach, $D(\text{hotel}) = D^{p_1}(\text{hotel}) = \{\text{name, phone}\}$, $D(\text{flight}) = D^{p_1}(\text{flight}) = \{\text{name, id-number}\}$, and $D(\text{pay}) = D^{p_1}(\text{pay}) = \{\text{name, phone, id-number, credit-card-info}\}$. From Eq. (5), $\mathrm{SC}_{\text{hotel}} = \{M, H, \text{TH}\}$, $\mathrm{SC}_{\text{flight}} = \{H, \text{TH}\}$, and $\mathrm{SC}_{\text{pay}} = \{\text{TH}\}$. The current trust levels of hotel, flight, and pay are M, H, and TH, respectively. In this case, the privacy information flows in the TA are secure because $M \in \mathrm{SC}_{\text{hotel}} \wedge H \in \mathrm{SC}_{\text{flight}} \wedge \text{TH} \in \mathrm{SC}_{\text{pay}}$. The privacy information flows remain secure, as long as the new trust level of each component service belongs to its security scope according to Evolution operation 1.

Suppose that Bob updates the third privacy rule to $r_3 = (\{\text{id-number}\}, \text{TH})$. The change set of privacy rules is $R_{\mathrm{c}} = \{r_3\}$, the set of direct privacy data items involved is $D_{\mathrm{c}} = \{\text{id-number}\}$, and the set of component services involved is $S_{\mathrm{c}} = \{\text{flight, pay}\}$. For service flight, $L_{\mathrm{sl}}(D(\text{flight})) = L_{\mathrm{sl}}(\{\text{name, id-number}\}) = \text{TH}$ and $L_{\mathrm{tl}}(\text{flight}) = H$ such that $L_{\mathrm{sl}}(D(\text{flight})) \rightarrow L_{\mathrm{tl}}(\text{flight})$ does not hold. The new

Table 1 Analysis process of path p_1

Transition	Type	Corresponding operation	Set of input privacy data items	Set of output privacy data items	Subject	Subject.Hds
t_1	RECV	Travel booking	None	{name, id-number, credit-card-info, phone}	User	N/A
t_2	Flow split	N/A	None	None	TA	N/A
t_3	SND	Hotel booking	{name, phone}	None	Hotel	{name, phone}
t_4	RECV	Response to hotel booking	{name, phone}	{hotel-booking-info}	Hotel	{name, phone}
t_5	SND	Flight booking	{name, id-number}	None	Flight	{name, id-number}
t_6	RECV	Response to flight booking	{name, id-number}	{flight-booking-info}	Flight	{name, id-number}
t_7	Flow join	N/A	None	None	TA	N/A
t_8	SND	Payment request	{hotel-booking-info, flight-booking-info, credit-card-info}	None	Pay	{hotel-booking-info, flight-booking-info, credit-card-info}
t_9	RECV	Response to payment	{hotel-booking-info, flight-booking-info, credit-card-info}	{pay-result}	Pay	{hotel-booking-info, flight-booking-info, credit-card-info}
t_{10}	SND	Response to travel booking	{hotel-booking-info, flight-booking-info, credit-card-info}	None	User	N/A

Subject.Hds: set of privacy data items a subject has ever collected; N/A: not available

Table 2 Underlying data for evolution

Service	$D^{p_1}(s)$	$D(s)$	SC_s
Hotel	{name, phone}	{name, phone}	{*M*, *H*, TH}
Flight	{name, id-number}	{name, id-number}	{*H*, TH}
Pay	{name, phone, id-number, credit-card-info}	{name, phone, id-number, credit-card-info}	{TH}

$D(s)$ and $D^p(s)$ denote the sets of direct privacy data items collected by component service s and path p, respectively; SC_s: the security scope of s

security scope is $SC_{flight} = \{TH\}$. To preserve privacy information flow security, the new trust level of service flight should be TH according to Evolution operation 2.2. For service pay, since $L_{tl}(pay) = TH$ and $L_{sl}(D(pay)) = L_{sl}(\{name, phone, id\text{-}number, credit\text{-}card\text{-}info\}) = TH$, the changes do not undermine the security of privacy information flows to it. Because service hotel is not affected by the changes, the privacy information flows to it are still secure.

6.2 Performance analysis

Since the time cost of Evolution operation 2.1 is the highest among the three evolution operations, we evaluate it through simulations. The simulation program is developed with JDK1.7 and runs in a PC (Intel Core i3 CPU @ 2.4 GHz, 4 GB memory, and Windows 7 Professional). Let n denote the number of direct privacy data items and m denote the number of component services. Suppose the number of privacy rules is $2n$. The number of privacy rules for the combination (including $n/2$ data items) of privacy data items is n. The number of direct privacy data items collected by each component service is $n/2$. Fig. 6 shows the execution time of Evolution operation 2.1.

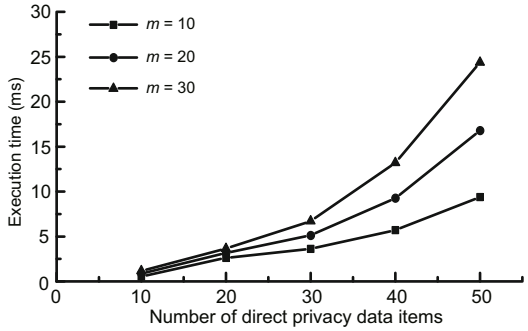

Fig. 6 Time cost of Evolution operation 2.1

According to Eq. (3), the time complexity of the calculation of $L_{sl}(D(s))$ is $O(n^3)$. Because Evolution operation 2.1 needs to verify $L_{sl}(D(s)) \rightarrow L_{tl}(s)$

for all component services, its time complexity is $O(mn^3)$. As shown in Fig. 6, the execution time of Evolution operation 2.1 does not grow drastically as m and n increase. In practical applications, the numbers of direct privacy data items and component services are usually small. The simulation results show that the evolution operations have high evolution efficiency.

Now, we investigate the time cost of our approach compared to that of re-verifying the entire composite service. We still choose Evolution operation 2.1 for comparative analysis. Suppose that there is only one path that needs to be verified. The number of transitions is $2m$ and the number of privacy data items processed by each transition is $n/2$. Fig. 7 shows the time costs of different approaches when m is 10.

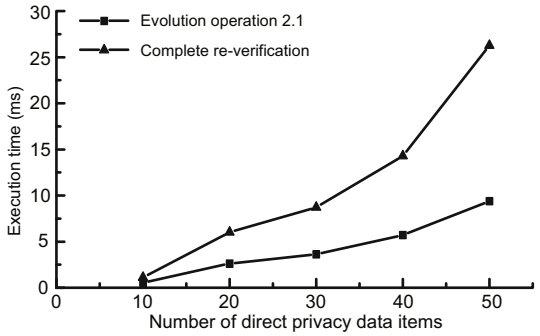

Fig. 7 Time cost by Evolution operation 2.1 and complete re-verification ($m = 10$)

As shown in Fig. 7, compared with re-verifying the entire composite service, our approach is superior in terms of time cost. The underlying data for the subsequent evolution have already been recorded when we initially verify the privacy information flow security. Therefore, the time cost of our approach is irrelevant to the number of paths in a composite service, and this can greatly reduce the cost of evolution.

7 Conclusions and future work

In this study, we have investigated how to preserve privacy information flow security in a composite service, when the trust levels of component services and user requirements change. Based on the underlying data recorded during static verification, evolution operations that preserve privacy information flow security have been proposed. Furthermore, we have illustrated the effectiveness of our approach through a case study. Compared with re-verifying the entire composite service, we have shown through simulations that our approach can reduce the evolution cost more effectively. In the scenario of online evolution, too long verification time is generally unacceptable. Through our approach, the complicated re-verification process can be avoided, and the evolution efficiency can be improved greatly. However, our approach does not consider privacy information leakage caused by control flow. In the future, we will intend to extend our work to address this issue.

References

Accorsi R, Lehmann A, Lohmann N, 2015. Information leak detection in business process models: theory, application, and tool support. *Inform Syst*, 47:244-257. https://doi.org/10.1016/j.is.2013.12.006

Alam KA, Ahmad R, Akhunzada A, et al., 2015. Impact analysis and change propagation in service-oriented enterprises: a systematic review. *Inform Syst*, 54:43-73. https://doi.org/10.1016/j.is.2015.06.003

Andrikopoulos V, Benbernou S, Papazoglou MP, 2012. On the evolution of services. *IEEE Trans Softw Eng*, 38(3):609-628. https://doi.org/10.1109/TSE.2011.22

Bacon J, Eyers D, Pasquier TFJM, et al., 2014. Information flow control for secure cloud computing. *IEEE Trans Netw Serv Manag*, 11(1):76-89. https://doi.org/10.1109/TNSM.2013.122313.130423

Bell DE, LaPadula LJ, 1973. Secure computer systems: mathematical foundations. Technical Report, No. 2547. MITRE Corporation, Massachusetts, USA.

Bishop M, 2002. Computer Security: Art and Science. Addison Wesley, New Jersey, USA.

Denning DE, 1976. A lattice model of secure information flow. *Commun ACM*, 19(5):236-243. https://doi.org/10.1145/360051.360056

Fokaefs M, Mikhaiel R, Tsantalis N, et al., 2011. An empirical study on web service evolution. 9th IEEE Int Conf on Web Services, p.49-56. https://doi.org/10.1109/ICWS.2011.114

Knorr K, 2001. Multilevel security and information flow in Petri net workflows. 9th Int Conf on Telecommunication Systems, p.613-615.

Liu C, Duan H, Zeng Q, et al., 2016. Towards comprehensive support for privacy preservation cross-organization business process mining. *IEEE Trans Serv Comput*, in press. https://doi.org/10.1109/TSC.2016.2617331

Liu G, Reisig W, Jiang C, et al., 2016. A branching-process-based method to check soundness of workflow systems. *IEEE Access*, 4:4104-4118. https://doi.org/10.1109/ACCESS.2016.2597061

Liu L, Zhu H, Huang Z, 2011. Analysis of the minimal privacy disclosure for web services collaborations with role mechanisms. *Expert Syst Appl*, 38(4):4540-4549. https://doi.org/10.1016/j.eswa.2010.09.128

Lohmann N, Massuthe P, Stahl C, et al., 2006. Analyzing interacting BPEL processes. 4th Int Conf on Business Process Management, p.17-32. https://doi.org/10.1007/11841760_3

Peng HF, Huang ZQ, Liu LY, et al., 2017. Static analysis method of secure privacy information flow for service composition. *J Softw*, in press.

Qi SS, Li BX, Liu CC, et al., 2012. A trust impact analysis model for composite service evolution. 19th IEEE Asia-Pacific Software Engineering Conf, p.73-78. https://doi.org/10.1109/APSEC.2012.30

She W, Yen IL, Thuraisingham B, et al., 2011. Rule-based run-time information flow control in service cloud. 9th IEEE Int Conf on Web Services, p.524-531. https://doi.org/10.1109/ICWS.2011.35

Song W, Ma XX, Cheung SC, et al., 2010. Preserving data flow correctness in process adaptation. 7th IEEE Int Conf on Services Computing, p.9-16. https://doi.org/10.1109/SCC.2010.24

Tan W, Fan YS, Zhou MC, 2009. A Petri net-based method for compatibility analysis and composition of web services in business process execution language. *IEEE Trans Autom Sci*, 6(1):94-106. https://doi.org/10.1109/TASE.2008.916747

van der Aalst WMP, 1997. Verification of workflow nets. 18th Int Conf on Application and Theory of Petri Nets, p.407-426. https://doi.org/10.1007/3-540-63139-9_48

Wang SY, Capretz MAM, 2009. A dependency impact analysis model for web services evolution. 7th IEEE Int Conf on Web Services, p.359-365. https://doi.org/10.1109/ICWS.2009.62

Wang SY, Capretz MAM, 2011. Dependency and entropy based impact analysis for service-oriented system evolution. 10th IEEE/WIC/ACM Int Conf on Web Intelligence and Intelligent Agent Technology, p.412-417. https://doi.org/10.1109/WI-IAT.2011.196

Wang Y, Wang Y, 2013. A survey of change management in service-based environments. *Serv Orient Comput Appl*, 7(4):259-273. https://doi.org/10.1007/s11761-013-0128-4

Wang Y, Yang J, Zhao WL, et al., 2012. Change impact analysis in service-based business processes. *Serv Orient Comput Appl*, 6(2):131-149. https://doi.org/10.1007/s11761-011-0093-8

Xi N, Sun C, Ma JF, et al., 2015. Secure service composition with information flow control in service clouds. *Fut Gener Comput Syst*, 49:142-148. https://doi.org/10.1016/j.future.2014.12.009

Yu WY, Yan CG, Ding ZJ, et al., 2014. Modeling and validating e-commerce business process based on Petri nets. *IEEE Trans Syst Man Cybern Syst*, 44(3):327-341. https://doi.org/10.1109/TSMC.2013.2248358

Yu WY, Yan CG, Ding ZJ, et al., 2016. Modeling and verification of online shopping business processes by considering malicious behavior patterns. *IEEE Trans Autom Sci Eng*, 13(2):647-662. https://doi.org/10.1109/TASE.2014.2362819

Zeng J, Sun HL, Liu XD, et al., 2010. PRV: an approach to process model refactoring in evolving process-aware information systems. 7th IEEE Int Conf on Services Computing, p.441-448. https://doi.org/10.1109/SCC.2010.19

Zeng W, Koutny M, Watson P, et al., 2016. Formal verification of secure information flow in cloud computing. *J Inform Secur Appl*, 27:103-116. https://doi.org/10.1016/j.jisa.2016.03.002