Security-Aware Service Composition with Fine-Grained Information Flow Control

Wei She, Student Member, IEEE, I-Ling Yen, Member, IEEE, Bhavani Thuraisingham, Fellow, IEEE, and Elisa Bertino, Fellow, IEEE

Abstract—Enforcing access control in composite services is essential in distributed multidomain environment. Many advanced access control models have been developed to secure web services at execution time. However, they do not consider access control validation at composition time, resulting in high execution-time failure rate of composite services due to access control violations. Performing composition-time access control validation is not straightforward. First, many candidate compositions need to be considered and validating them can be costly. Second, some service composers may not be trusted to access protected policies and validation has to be done remotely. Another major issue with existing models is that they do not consider information flow control in composite services, which may result in undesirable information leakage. To resolve all these problems, we develop a novel three-phase composition protocol integrating information flow control. To reduce the policy evaluation cost, we use historical information to efficiently evaluate and prune candidate compositions and perform local/remote policy evaluation only on top candidates. To achieve effective and efficient information flow control, we introduce the novel concept of transformation factor to model the computation effect of intermediate services. Experimental studies show significant performance benefit of the proposed mechanism.

Index Terms—Secure service composition, access control, information flow control

1 INTRODUCTION

SERVICE composition has been extensively studied in recent years. Although a lot of new models and mechanisms have been proposed, many issues in service composition still remain unsolved. Among them, access control is one of the major concerns. Though it is essential to develop access control models and techniques to secure individual component services as well as composite services, it is also necessary to consider when to evaluate the access control policies. All existing works consider access control only at execution time. However, without the composition-time access control validation, the composite service may be very likely to fail at the execution time due to access control violations, wasting composition and execution efforts. To avoid repeating failed compositions, bookkeeping of the failure becomes necessary, resulting in a more complicated and time-consuming composition and execution protocol. To resolve this problem, it is desirable to enforce the access control policies of individual component services at the service composition time in addition to the execution-time access control enforcement.

Existing works do not consider composition time access control validation. There have been some works on security-aware composition [3], [6], [9], [10], [18]. These works characterize security as a set of attributes that can be quantitatively measured. The security properties of a service can be specified in terms of these attributes. To protect critical data resources and/or services, the user and service providers may define security constraints (i.e., policies) for a composition in terms of these attributes. A composite service is considered to be secure if the security properties of all individual services satisfy all the security constraints defined by the service providers and the user.

One major issue with the above works is that they consider very simple attributes such as the type of encryption algorithm, the type of authentication protocol, and so on. Although it is possible to extend these mechanisms to include access control policies as security constraints, none of these works consider the specific issues involved in modeling, specification, and evaluation of these policies.

There are several issues when considering access control at composition time. The first issue is who should evaluate the access control policies. Existing secure service composition mechanisms assume a fully trusted service composer, which is not always true, especially in a multidomain environment. In such an environment, there are many users in different domains and they may use different service composers. These distributed service composers may be in different domains from those of the web services. Some of these domains may have protected access control policies that should not be released to some parties (e.g., some service composers). Thus, it is unlikely that a service composer is fully trusted by the providers of all involved services (all the concrete services considered by the composer, not just those actually selected) for accessing their protected policies. Consequently, the service composer cannot complete policy evaluation without interacting with the service providers with protected policies.

W. She, I. Yen, and B. Thuraisingham are with the Computer Science Department, University of Texas at Dallas, 800 West Campbell Road, Richardson, TX 75080.
E-mail: jws061000, ilyen, bhavani.thuraisingham@utdallas.edu.
E. Bertino is with the Computer Sciences Department, Purdue University, 305 N. University Street, West Lafayette, IN 47907.
E-mail: bertino@cs.purdue.edu.

Manuscript received 22 Apr. 2011; revised 10 Nov. 2011; accepted 27 Dec. 2011; published online 16 Jan. 2012.
For information on obtaining reprints of this article, please send e-mail to: tsc@computer.org, and reference IEEECS Log Number TSC-2011-04-0035. Digital Object Identifier no. 10.1109/TSC.2012.3.
Second, the performance issue in secure service composition should be carefully considered. The service composer may have to explore a lot of candidate compositions to find one candidate that satisfies the security constraints of all component services. To validate each candidate, the composer needs to validate each service pair \((s_i, s_j)\) by evaluating \(s_j\)'s security properties against \(s_i\)'s access control policies. The policies of some services may be very complex and require a significant amount of evaluation time. Also, a service composer may not readily have the policies of all services from all domains and may need to download the needed policies at the composition time. In case some policies are protected, the composer needs to interact with the security authorities of the corresponding domains for remote policy evaluation or negotiation. If such a policy evaluation process is applied to every candidate composition, the cost can be extremely high. Note that if policy evaluation is delayed to execution time, the problem of exploring many potentially illegitimate compositions would occur and the cost would be even higher due to the involvement of actual execution.

We introduce a three-phase service composition protocol to address the performance issue and trusted composer issue in composition-time access control validation. In the first phase, as the search space may be very large, a more efficient but less precise method is used to quickly prune the candidate compositions. Specifically, we use the information of historical service composition transactions to estimate the fitness of candidate compositions, rank them, and select the top candidates. In the second phase, we consider a local policy evaluation process to achieve more precise evaluation of candidate compositions. In this process, the composer uses the policies and/or certificates cached or newly downloaded from the security authorities of involved services to locally validate the candidate compositions. The accesses to the policies by the composers are considered as special privileges granted rather than assumed. Although it is unlikely that the service composer can validate all candidate compositions, a majority of the service pairs may still be validated, and many invalid candidate compositions can be eliminated. In the third phase, we consider a remote policy evaluation process to validate previously unverifiable pairs of services. In this process, the security authorities of the involved services evaluate their protected access control policies and return their decisions to the composer to help derive the final composition decisions. Negotiation may be needed in this phase to exchange the credentials and/or policy information between the service composer and the security authorities and between the security authorities in different domains. Since the service pairs validated in the second phase need not be checked again, this time-consuming process will only be performed on a few service pairs of very few final candidates.

Besides not considering access control at composition time, existing access control models also do not effectively handle information flow control (IFC) (neither composition time nor execution time). Existing works consider advanced models to secure individual web services, including adaptive action-based access control [4], context-aware access control [5], access control for conversational web services [19], credential-based access control [2], [1], and so on. Some of these works also consider securing composite services at the execution time [1], [30]. But they only consider direct accesses to the services and do not consider the flow of sensitive information among indirectly interacting services. Consider a service chain \(<s_0, s_1, s_2>\). Assume that \(s_1\)'s output is computed from some of its own sensitive information and some sensitive data received from \(s_0\). When \(s_1\)'s output is sent to \(s_2\), \(s_2\) may use the received data to derive the sensitive information of \(s_0\), resulting in an information flow from \(s_0\) to \(s_2\). Such information flows, if not handled carefully, may result in undesired information leakage.

There have been some limited works that address the information flow problem in composite services. But they are either too strict, treating direct and indirect accesses exactly the same way [7], [27], or too complex, exhaustively enumerating all the possible combinations of intermediate services and specifying IFC policies accordingly [23]. To achieve fine-grained IFC while avoiding the pitfalls in existing models, we introduce the concept of transformation factor (TF), which specifies how likely the sensitive input or local data of a service can be derived from its output, and consider it in making IFC decisions. Note that the IFC policies should also be evaluated at both composition time and execution time. We integrate the IFC model into our three-phase composition protocol.

To study the performance of the proposed mechanism, we develop a simulation system to simulate various protocols and compare their performance, including the three-phase composition protocol, the single-phase composition protocol, and the protocol without composition-time access control validation. The result shows that, without composition-time access control validation, the composition and execution cost increases dramatically as the success rate decreases (If the access control policies of component services are strict, then it is difficult to find a valid composition and the success rate is low). When the success rate is around 50 percent, even the single-phase protocol performs much better than the other two mechanisms even when the success rate is high (90 percent). We also compare the performance of the protocols under various service chain sizes. With the increasing service chain size, the performance gain by the three-phase protocol becomes more significant.

The rest of this paper is organized as follows: Section 2 presents the system model, including a general model for web service systems, the access control model, and the formalization of service chain and information flow. Section 3 presents a motivating example to illustrate various issues and considerations. Section 4 introduces the IFC rules, which will be used to guide the composition process. In Section 5, we discuss the three-phase composition protocol. The experimental study and results are presented in Section 6. Section 7 presents related works. Section 8 concludes this paper.
2 System Model

2.1 A Model of Web Service System

We consider a general web service system (Fig. 1), which consists of multiple domains and multiple service composers. Each domain includes a set of web services, a set of data resources, and a security authority (SA), which manages a set of access control policies to control the accesses to the data resources in the domain. The web service system is defined in Definition 2.1.

Definition 2.1. A web service system includes a set of domains \( \{d_1, d_2, \ldots\} \) and a set of service composers \( \{scomp_1, scomp_2, \ldots\} \). Each domain \( d_i \) is a tuple \( <d_i, S, d_i, R, d_i, sa> \), where \( d_i, S = \{d_i, s_1, d_i, s_2, \ldots\} \) is the set of all services in \( d_i, d_i, R = \{d_i, r_1, d_i, r_2, \ldots\} \) is the set of all data resources in \( d_i, \) and \( d_i, sa \) is the SA of \( d_i. \) \( d_i, sa \) manages a set of access control policies \( d_i, Pol = \{d_i, pol_1, d_i, pol_2, \ldots\} \) to control the accesses to \( d_i, R.\)

Data resource refers to the data/information itself and any entity that may store or receive data/information. Such an entity can be a data container, such as a file, a directory, a relation, a view, and so on, or an exhaustible resource, such as a printer, a scanner, the disk space, CPU cycle, and so on. We assume that all services are semihonest. They follow the protocol and conform to the access control policies, but some services may attempt to derive the sensitive information of others from the information they have received. Also, we do not consider the interoperability issues. There are techniques in the literature that can help resolve these issues [15].

For convenience, we use \( dom(x) \) to represent the domain of \( x, \) where \( x \) can be a service or a data resource. Also, we use \( Pol(r), Pol(r) \subseteq dom(r).Pol, \) to denote the set of all access control policies in \( dom(r) \) that are applicable to \( r, \) where \( r \) is a data resource.

2.2 Service and Service Chain

We consider an abstract dataflow model to model the flow of data/information in service chains. In this model, each service \( y \) takes the input data \( y, In \) from the end user or another service \( x, \) completes its own computation, and generates its output \( y, Out, \) which is delivered to another service or the end user \( z. \) The computation of \( y \) may use some data resources stored in \( dom(y), \) i.e., \( y, R. \) As we only consider the deterministic system, the set of output data of \( y, y, Out, \) can be expressed as a function of its input, \( y, In, \) and local data resources, \( y, R. \) We define web service as follows:

Definition 2.2. A web service \( s \) is a tuple \( <s, In, s, Out, s, R, s, F> \), where \( s, In = \{s, in_1, s, in_2, \ldots\} \) is the set of all input data of \( s, s, Out = \{s, out_1, s, out_2, \ldots\} \) is the set of all output data of \( s, s, R = \{s, r_1, s, r_2, \ldots\} \) is the set of all local data in \( dom(s) \) that are used in the computation of \( s, \) and \( s, F \) is the computation function of \( s \) that \( s, Out = s, F(s, In, s,R). \)

Generally, a composite service can be defined using a workflow, which is the composition of component services. We consider abstract and concrete workflows. In an abstract workflow, each component service is abstract and is to be grounded to a concrete service. In a concrete workflow, each component service is a concrete web service. Upon composition, the service composer is given the desired abstract workflow and instantiates each abstract component service by a concrete service. In this paper, we only consider a simplified workflow, a service chain. The simplification is for the convenience in defining the notations and algorithms. The solutions provided in this paper are applicable to general workflows with parallel composition and loop. We define the abstract and concrete service chains as follows:

Definition 2.3. An abstract service chain \( <s_0, a s_1, \ldots, a s_n, s_{n+1}> \) consists of two end users, \( s_0 \) and \( s_{n+1}, \) where \( s_0 \) is the user who sends the input data to \( a s_1 \) and \( s_{n+1} \) is the user who receives the output data from \( a s_n. \) A composite service chain \( <s_0, s_1, \ldots, s_n, s_{n+1}> \) consists of the two end users, \( s_0 \) and \( s_{n+1}, \) and a sequence of concrete services \( s_1, \ldots, s_n, \) that should be grounded to concrete services. A concrete service chain \( <s_0, a s_1, \ldots, a s_n, s_{n+1}> \) to a service composer and the composer returns a concrete service chain \( <s_0, s_1, \ldots, s_n, s_{n+1}> \) to the user, where \( a s_i \) is grounded to \( s_i, 1 \leq i \leq n. \) Note that we consider the two end users, \( s_0 \) and \( s_{n+1}, \) as services. They may be the same user or may be different. During composition, we only need to consider the selection of \( a s_i \) to \( a s_i. \) But when considering access control, all users/services in the concrete service chain, from \( s_0 \) to \( s_{n+1}, \) should be considered. The service composer explores various candidate concrete compositions \( ch_k, \) for all \( k, \) and selects the best solution to return to the user. We use \( ch_k, <s_0, a s_1, \ldots, a s_n, s_{n+1}> \) to represent the concrete service chain for \( ch_k \) and \( ch_k, s_i \) to represent the specific service \( s_i \) in this chain.

2.3 Attribute-Based Access Control

We consider a general attribute-based access control model [11], [24], [29]. A set of attributes is defined for each service/data resource. The attributes of a service may include service name, WSDL pointer, the permission granted to the service, reputation, and so on. The attributes of a data resource may include owner, security classification, and so on. The attributes of a data resource are included in the metadata and stored with the data. The attributes of a service must be asserted by a SA and included in a certificate, called the attribute certificate. The attribute...
certificate must be signed by its issuer. The attribute and attribute certificate are defined as follows:

**Definition 2.4.** Each service or data resource $x$ is associated with a set of attributes $\text{Attr}(x) = \{\text{attr}_1(x), \text{attr}_2(x), \ldots\}$. Each attribute $\text{attr}(x) \in \text{Attr}(x)$ is defined as a tuple $(\text{attr}(x).\text{name}, \text{attr}(x).\text{val})$ in which $\text{attr}(x).\text{name}$ is a string that uniquely specifies the name of the attribute, and $\text{attr}(x).\text{val}$ is the value of the attribute.

Each service $s$ owns a set of attribute certificates $s.\text{AC}$. Each attribute certificate $\text{ac} \in s.\text{AC}$ is issued by a SA to certify that $s$ owns certain attributes $s.\text{ac}.\text{Attr}$, where $s.\text{ac}.\text{Attr} \subseteq \text{Attr}(s)$.

We consider that the SA in each domain manages the attribute certificates of all services in the domain. The attributes of a service can be sensitive or nonsensitive. Attribute certificates containing only nonsensitive attributes can be freely exchanged among different parties. However, the release of an attribute certificate with sensitive attributes requires negotiation.

In attribute-based access control, an access control policy is a set of conditions defined over the set of all attributes used by a domain (the set of all attributes defined for services and resources in the domain). For simplicity, we consider a unified set of attributes defined across all domains. When a service $s$ accesses a data resource $r$, $s$ presents its attribute certificate $s.\text{ac}$, which contains a set of attributes $s.\text{ac}.\text{Attr}$ to $\text{dom}(r).\text{sa}$, $\text{dom}(r).\text{sa}$ verifies $s.\text{ac}$ from its issuer (may be $\text{dom}(s).\text{sa}$, $\text{dom}(r).\text{sa}$, or another SA trusted by $\text{dom}(r).\text{sa}$) and extracts $s.\text{ac}.\text{Attr}$ from $s.\text{ac}$ and evaluates against the access control policies $\text{Pol}(r)$, $\text{Pol}(r) \subseteq \text{dom}(r).\text{Pol}$.

Fig. 2 shows an example attribute certificate and attribute-based access control policy. To evaluate the policy, the SA substitutes the variables in the policy by $s$'s attributes (e.g., replace subscriptionType(s) by “regular”).

### 3 Motivating Example

We consider an example application workflow (Fig. 3) to motivate and demonstrate the need for the IFC in service composition and the benefit of considering access control at composition time. It is also used as a running example to help illustrate various concepts in our model. The workflow is used to help with screening of disease $x$ by first extracting association rules from medical data of patients with and without disease $x$. The association rules are then used to determine how likely a new patient does have disease $x$. The workflow consists of the following abstract services, a client program CLN, a medical database MDB, a template image database TDB, an image enhancement service IES, an image registration service IRS, an object recognition service ORS, an association rule mining service ARM, and a classifier CLS. CLN first searches MDB (with keyword $x$) for the medical records of the patients who are diagnosed to have the disease $x$, and searches TDB (with keywords such as “bone,” “polyp,” “nodule,” and so on) for the template images for object recognition. Each medical record stored in MDB includes the alphanumeric medical data, for example, the patient’s medical history, family history, personal data (e.g., gender, age, height, weight, living area, etc.), and the medical images (e.g., CT, X-ray, Nuclear, etc.). The alphanumeric medical data of MDB are sent to ARM. The template images are sent to ORS. The medical images are first sent to IES, which performs image enhancement (e.g., noise cancellation, etc.). The enhanced images are sent to IRS, which performs image registration to align different images into one coordinate system. The aligned images are sent to ORS, which detects and recognizes the objects in the images (e.g., bones, polyps, nodules, etc.) using the template images. After recognition, it assigns labels to the recognized objects in the image. The labeled images are sent to ARM, which uses these images together with the alphanumeric medical data received from MDB to extract association rules (in the form of $(y_1, \ldots, y_n) \Rightarrow x$, where $y_i$, $1 \leq i \leq n$ is an object label or a string extracted from the alphanumeric medical data and $x$ is the disease name) that are sent to CLS.

Now consider the sensitivity of the data that are used by the composite service (Note that this abstract composite service includes three abstract service chains). We assume that the search keywords that CLN sends to MDB and TDB are not sensitive and, hence, require no protection. The alphanumeric medical data that MDB send to ARM and the medical images that MDB send to IES are sensitive and the recipients are required to have read permissions to these data. (Note that the recipient rather than the invoker needs to have the proper privilege. For example, IES needs to have read permission to the medical images in MDB, but CLN does not.) The template images are used in a payer-use manner and, hence, require the recipients to present proper privilege.

Next, consider the concrete services that can be used to instantiate the abstract services and the privileges they have (Fig. 4). For simplicity, we assume that CLN, MDB, TDB, IES, IRS, and CLS are already concretized by $\text{cls}_1$, $\text{mdb}_1$, $\text{tdb}_1$, $\text{ies}_1$, $\text{irs}_1$, and $\text{cls}_1$, respectively. $\text{cls}_1$ and $\text{cls}_1$ are served by hospital A (domain $d_A$). $\text{mdb}_1$ and $\text{tdb}_1$ are served by hospital B (domain $d_B$) and research institute C (domain $d_C$), respectively. $\text{ies}_1$ and $\text{irs}_1$ are served by research institute D (domain $d_D$). ORS can be instantiated by $\text{ors}_1$, $\text{ors}_2$, and $\text{ors}_3$. Note that $\text{ies}_1$ does not modify the content of the medical image received from $\text{mdb}_1$, and $\text{irs}_1$
does not modify the content of the images received from ics1. Hence, the medical images of mdb1 are essentially delivered to the ORS service (ors1, ors2, or ors3) in their raw forms. ARM can be instantiated by arm1 and arm2. We consider that ors1 and arm1 are hosted by institute D, ors2 is hosted by research institute E (domain dE), and ors3 and arm2 are hosted by university F (domain dF).

For simplicity, we assume that all the services can be invoked by anyone and we only define the resource-based access control policies here. Consider that service x invokes service y. If y does not read/write any sensitive local data (e.g., a table, etc.) in its computation, then the invocation can be directly granted. If y reads some sensitive local data resources r, then the invocation is granted when x has the read permission to r. Similarly, if y writes to r, then the invocation is granted when x has the write permission to r. Fig. 4 depicts the resource access rights. We consider that institutes D and E are federated with hospital B and, hence, services ies1, irs1, ors1, arm1, ors2 have the read permission to the medical data in domain dB. Also, we consider that institute E has purchased the service of tdb1 from institute C, and hence, ors2 has the read permission to the template images in domain dc. No other read/write accesses to the medical data or the template images are allowed.

ICF illustration. Conventional access control models cannot address the security needs in this application. For example, the medical images of mdb1 are delivered to the ORS service in raw forms and ors3 does not have the read permission for the medical images of mdb1, the selection of ors3 should be prohibited. Also, as arm2 does not have the privilege for the template images in tdb1, arm2 should not be selected either. However, conventional models do not consider IFC and will allow ORS to be grounded to ors3 or ARM to be grounded to arm2, resulting in undesirable information flows.

Benefit of composition-time access control validation. Assume that access control is not validated at composition time. Suppose that the workflow (Fig. 3) is first concretized by the composite services \{chn1, mdb1, tdb1, ies1, irs1, ors3, arm2, cls1\}. During execution, chn1, mdb1, and tdb1 are invoked and executed successfully. But when mdb1 sends out the alphanumeric medical data to arm2, because arm2 does not have read permission to the alphanumeric medical data of mdb1, mdb1 cannot send its output to arm2 and the execution fails. All the execution of chn1, mdb1, and tdb1 are wasted. After this failure, the execution coordinator needs to pass the execution information to the service composer and the composer records the access control violation between mdb1 and arm2 and then replaces arm2 with arm1. The execution of the new composite service \{chn1, mdb1, tdb1, ies1, irs1, ors3, arm1, cls1\} also fails as ors3 has neither the read permission to the medical images of mdb1 nor the permission to the template image of tdb1. Again, chn1, mdb1, ies1, irs1, and tdb1 have completed their execution and the efforts are wasted. Such failure may continue until the only feasible composition \{chn1, mdb1, tdb1, ies1, irs1, ors3, arm1, cls1\} is selected and executed.

If access control validation is performed at composition time, then the selected composition is likely to succeed, avoiding wasting unsuccessful execution efforts. (Note that there may still be policies that can only be evaluated at execution time, which can still result in failures).

4 IFC Rules

In a service chain, the output data of service \(s_i\) may be computed from some sensitive information of \(s_i\)’s prior services. When delivered to the subsequent services \(s_j\), \(j > i, s_j\) may be able to derive the sensitive information of \(s_i\) or of \(s_i\)’s prior services from the data it has received. Thus, during composition, it is desirable that the service composer can make sure that the information flow from \(s_i\) to \(s_j\) does not violate any security constraints. To ensure the secure information flow, each service in the service chain needs to define detailed policies to control the flow of its sensitive information. Also, it is desired to define rules to govern the composition process such that the service chain generated by the service composer does not violate any IFC policies specified by the services in the chain. In this section, we define the IFC rules.

4.1 Basic IFC Rules

The goal of IFC is to guarantee that the sensitive information of each service in a service chain is not only secured from direct accesses but also secured after it flows (in its raw or processed form) to the subsequent services. Note that, the output data of service \(s_i\)

\[
\begin{align*}
  s_i, \text{Out} &= s_i, F(s_i, \text{Out}, s_i, R) \\
  &= s_i, F(s_{i-1}, F(s_{i-2}, \text{Out}, s_{i-1}, R), s_i, R) = \cdots \\
  &= s_i, F(s_{i-1}, F(\ldots(s_i, F(s_0, F(s_0, R), s_1, R), \ldots), s_{i-1}, R), s_i, R).
\end{align*}
\]

Thus, to achieve IFC in service chain \(<s_0, \ldots, s_{n+1}>\), the service composer needs to ensure that, for each service pair in the service chain \((s_i, s_j), 0 \leq i \leq n, i < j \leq n + 1\), \(s_j\) is authorized to access the sensitive information contained in \(s_i, R\). This principle is specified by the basic IFC rules given as follows:

\(\text{BIFC}_1\). valid\((s_i, s_j) \iff \text{auth}(\text{Attr}(s_i), \text{Attr}(s_i, R), \text{Pol}(s_i, R)), \) for all \(i, j, 0 \leq i \leq n, i < j \leq n + 1\).

\(\text{BIFC}_2\). valid\(<s_0, \ldots, s_{n+1}> \iff \forall (s_i, s_j), \) for all \(i, j, 0 \leq i \leq n, i < j \leq n + 1\).
SHE ET AL.: SECURITY-AWARE SERVICE COMPOSITION WITH FINE-GRAINED INFORMATION FLOW CONTROL

TABLE 1

<table>
<thead>
<tr>
<th>TF Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NR (No risk)</td>
<td>Impossible to derive the sensitive information in the input or local data from the output.</td>
</tr>
<tr>
<td>LR (Low risk)</td>
<td>Difficult to derive the sensitive information in the input or local data from the output.</td>
</tr>
<tr>
<td>MR (Medium risk)</td>
<td>Some sensitive information in the input or local data is derivable from the output.</td>
</tr>
<tr>
<td>HR (High risk)</td>
<td>A majority or all of the sensitive information in the input or local data is derivable from the output, or the output contains the raw information of the input or local data.</td>
</tr>
</tbody>
</table>

Here, \(\text{auth}(\text{Attr}(s_j), \text{Attr}(s_i), \text{Pol}(s_i, R))\) denote the IFC decision (true, false, or unknown) made by evaluating the attributes of the requested service \(s_j\), \(\text{Attr}(s_j)\), and the attributes of the requested data resource \(s_i, R\), \(\text{Attr}(s_i, R)\), against the IFC policies that are applicable to \(s_i, R\), \(\text{Pol}(s_i, R)\). \(\text{valid}(s_i, s_j)\) denotes the validity of the service pair \((s_i, s_j)\). If \(\text{valid}(s_i, s_j)\) is true, then the selection of service pair \((s_i, s_j)\) is valid. If all the selections are valid, then the service composition is valid.

4.2 TF and Advanced IFC Rule

To validate a service chain based on the basic IFC rules, we need to evaluate \(s_i\) against the IFC policies of \(s_j\), for all \(i, j\), \(0 \leq i < n, i < j \leq n + 1\). In practice, some of the policy evaluation tasks may be unnecessary. First, there may exist some service \(s_{k_0}, 0 < k < n + 1\), such that the sensitive information contained in its input \(s_{k_0}.\text{In}\) is not derivable from its output, \(s_{k_0}.\text{Out}\). In this case, for any pair of services \((s_i, s_j)\), where \(0 \leq i < k, k < j \leq n + 1\), \(\text{valid}(s_i, s_j)\) is always true. On the other hand, if all the intermediate services between \(s_i\) and \(s_j\) do not transform their inputs when generating their outputs, \(s_j\) and \(s_j\) are as though interacting directly and, hence, \(s_j\) should be validated against the IFC policies of \(s_i, R\) (similar to the approach in [7], [27]). If the intermediate services between \(s_i\) and \(s_j\) transform \(s_i.\text{Out}\) to some extend when generating \(s_j.\text{Out}\) but there is still potential of deriving some partial information of \(s_i.\text{Out}\) from \(s_j.\text{Out}\), then the potential risk should be taken into consideration when validating \(s_j\) against \(s_i\)’s IFC policies. To recognize these situations, we introduce TF to estimate the risk of \(s_j\) deriving \(s_i\)’s sensitive information from \(s_j\)’s input data, \(s_j.\text{In}\), and consider the risk as a major factor in computing \(\text{valid}(s_i, s_j)\). We define multilevel TF to measure how a service processes its input and local data to generate its output in Definition 4.1.

**Definition 4.1.** In the service chain \(<s_0, \ldots, s_{n+1}>\), the TF of a service \(s_i\), \(tf(s_i), 0 \leq i \leq n + 1\), specifies how much transformation service \(s_i\) makes when generating \(s_i.\text{Out}\) using \(s_i.\text{In}\) and \(s_i.\text{R}\).

Table 1 lists four TF levels in our model. Note that \(\text{HR} < \text{MR} < \text{LR} < \text{NR}\).

Next, we define the TF for partial service chains, based on the TF of each individual service in the chain in Definition 4.2.

**Definition 4.2.** In the service chain \(<s_0, \ldots, s_{n+1}>\), the TF of a partial service chain \(<s_i, \ldots, s_j>\), \(0 \leq i \leq n, i < j \leq n + 1\) specifies how much transformation the partial service chain \(<s_i, \ldots, s_j>\) makes when generating \(s_j.\text{Out}\) using \(s_i.\text{In}\) and \(s_i.\text{R}\). \(tf(<s_i, \ldots, s_j>) = \max\{tf(s_i), \ldots, tf(s_j)\}\).

TF of a partial service chain can impact the IFC decisions of indirectly interacting services. It also has a major effect on the naive pairwise validation process. The flow of sensitive information in a service chain may be “broken” by a service with NR TF. According to Table 1, if there exists a service \(s_{k_0}, 0 < k < n + 1\), s.t. \(tf(s_{k_0}) = \text{NR}\), then we consider that it is very difficult to derive the sensitive information contained in \(s_{k_0}.\text{In}\) and \(s_{k_0}.\text{R}\) from \(s_{k_0}.\text{Out}\). This breaks the information flow, and the service chain can be considered as two partial service chains, \(<s_0, \ldots, s_{k_0}>\) and \(<s_{k_0}+1, \ldots, s_{n+1}>\). When considering IFC, such partial service chains can be considered separately, as the data generated by any two services in different partial chains can be considered as unrelated. Based on this observation, we define an advanced IFC rule to specify the situation under which the policy evaluation for some service pairs \((s_i, s_j)\) can be fully ignored (always true). Also, we rewrite the basic IFC rules to include TF as a factor in making IFC decisions. The advanced IFC rules are specified as follows:

**AIFC1.** \(\text{valid}(s_i, s_j) \equiv (tf(<s_i, \ldots, s_{j-1}>) = \text{NR}), \forall i, j, 0 \leq i < n, i < j \leq n + 1\).

**AIFC2.** \(\text{valid}(s_i, s_j) \equiv \text{auth}(\text{Attr}(s_j), \text{Attr}(s_i, R), tf(s_i, \ldots, s_{j-1}), \text{Pol}(s_i, R)), \forall i, j, 0 \leq i < n, i < j \leq n + 1\).

**AIFC3.** \(\text{valid}(<s_0, \ldots, s_{n+1}) \equiv \land \text{valid}(s_i, s_j), \forall i, j, 0 \leq i \leq n, i < j \leq n + 1\).

4.3 Examples of TF Settings

For some services (e.g., alphanumeric input/output, the local data access is static), static program analysis may help determine their TFs [20], [21]. However, in case that a service takes in or generates nonalphanumeric data (e.g., image, audio, video, etc.), the TF may only be determined based on its functionality. For simplicity, we consider that a security officer decides the TFs of services in each domain.

Consider the example system in Section 3. The image enhancement service $I_E S$ takes in a set of input images, removes the noise in them, and reconditions them for object recognition. Since the enhanced image contains raw data in the original image, its TF can be HR.

In some cases, part of the sensitive information contained in the input may be derivable from its output. The search engine $M D B$ searches for the medical records of the patients diagnosed to have disease $x$. Though the search keyword $x$ may not be directly seen in the search result, it may be possible to guess the value of $x$ from the common information in these records. If the likelihood of making a successful guess (e.g., when the output always contains very few records) is very low, then the TF can be set to LR. If the likelihood is moderate, then the TF is set to MR.
The association rule mining service ARM takes in a set of medical images with object labels and the associated alphanumeric medical data and derives association rules. As it is impossible to derive the sensitive information, such as the patient medical history, and so on, from the output association rules, its TF can be set to NR.

5 Security-Aware Service Composition

In service composition, the service composer takes an abstract service chain \(<s_0, a_{s_1}, \ldots, a_{s_n}, s_{n+1}>\) from the user and selects concrete services to instantiate \(a_{s_i}, \ldots, a_{s_n}\) while satisfying the IFC constraints. The composer first retrieves a set of concrete services for each abstract service \(a_{s_i}\) from UDDI and generates a set of candidate concrete compositions \(CH_0\). For each \(ch_k\) in \(CH_0\), the composer follows the advanced IFC rules and verifies whether \(valid(ch_k, <s_0, \ldots, s_{n+1}>)\) is true.

The service composer may have to explore \(O(c^n)\) candidate concrete compositions, where \(n\) is the number of abstract services in the abstract service chain and \(c\) is the average number of candidate concrete services per abstract service. For each candidate composition \(ch_k\), the service composer needs to verify whether \(valid(ch_k, s_i, s_j)\) is true, for all \(i, j, 0 \leq i \leq n, i < j \leq n + 1\). This requires generating \(O(n^2)\) IFC decisions. The decision making may also involve retrieving \(Pol(ch_k, s_i, R)\) from \(dom(ch_k, a_{s_i})\) and \(ch_k, a_{s_i}\) from \(dom(ch_k, s_i)\). Even if \(Pol(ch_k, a_{s_i})\) and \(ch_k, a_{s_i}\) are cached by the service composer, the policy evaluation may still be expensive, as the service composer may have to evaluate many rules in \(Pol(ch_k, s_i, R)\) and combine the results. In case that \(Pol(ch_k, s_i, R)\) and/or some attributes in \(ch_k, a_{s_i}\) are protected, the service composer needs to interact with \(dom(ch_k, a_{s_i})\) and/or \(dom(ch_k, s_i)\) to compute \(auth(Attr(ch_k, s_i), Attr(ch_k, a_{s_i}, R), tf(\langle ch_k, s_{i+1}, \ldots, s_{j-1}\rangle), Pol(ch_k, s_i, R))\). If this policy evaluation process is applied to each candidate composition, the composition cost can be very high.

We consider a three-phase mechanism to achieve efficient security-aware service composition. In the first phase (Section 5.1), there are many candidates and an efficient method is used to quickly evaluate candidate concrete compositions and the most promising candidates are selected for further analysis. Instead of actually computing \(valid(ch_k, s_i, s_j)\) using \(auth(Attr(ch_k, s_i), Attr(ch_k, a_{s_i}, R), tf(\langle ch_k, s_{i+1}, \ldots, s_{j-1}\rangle), Pol(ch_k, s_i, R))\), the service composer uses the validation results of historical composition transactions to compute the likelihood of \(valid(ch_k, s_i, s_j)\) being true (denoted as \(LL(ch_k, s_i, s_j)\)). Accordingly, the fitness value of each candidate composition \(ch_k\), \(fit(ch_k)\) is computed, and the top \(L_1\) is the percentage that ranges from 0 to 1) candidates are selected and included in \(CH_1\).

In the second phase (Section 5.2), a more accurate but potentially more time consuming process is used to evaluate the candidates in \(CH_1\). For each candidate in \(CH_1\), \(ch_k\), the service composer uses the cached or newly downloaded (if the information is not cached or is stale) policies \(Pol(ch_k, s_i, R)\) and/or certificates \(ch_k, a_{s_i}\) to compute \(auth(Attr(ch_k, s_i), Attr(ch_k, a_{s_i}, R), tf(\langle ch_k, s_{i+1}, \ldots, s_{j-1}\rangle), Pol(ch_k, s_i, R))\), for all \(i, j, 0 \leq i \leq n, i < j \leq n + 1\), locally.

5.1 First-Phase Analysis

We develop a set of likelihood computation (LLC) rules to compute \(LL(ch_k, s_i, s_j)\). First, we consider the case when there is an information flow break between \(ch_k, s_i\) and \(ch_k, s_j\).

\[
\text{LLC}_1: \quad \text{if } tf(\langle ch_k, s_{i+1}, \ldots, s_{j-1}\rangle) = NR, \text{ then } LL(ch_k, s_i, s_j) = 1.
\]

If rule LLC_1 is not applicable, then we need to use the historical validation results to estimate \(LL(ch_k, s_i, s_j)\). To facilitate the estimation of the likelihood, the service composer maintains a database VDB to store the validation results of all service pairs in historical composition transactions. The TF between the two services also impacts the IFC decisions. Thus, each record in VDB (for service pair \((x, y)\)) records \(\langle x, y, res\text{ult}, tf\rangle\). Here, \(\langle x, y, res\text{ult}, tf\rangle\) is the final result of \(valid(x, y)\), \(x, y, tf = tf(\langle x, \ldots, pre(y)\rangle), \) where \(pre(y)\) denotes the service right before \(y\) in

---

Fig. 5. Three-phase service composition algorithm.

In some cases, the value of \(valid(ch_k, <s_0, \ldots, s_{n+1}>)\) may be determined. In case that \(valid(ch_k, <s_0, \ldots, s_{n+1}>)\) is false, \(ch_k\) is removed from \(CH_1\). If \(valid(ch_k, <s_0, \ldots, s_{n+1}>)\) is true, \(ch_k\) (a valid composition) is directly returned to the user and the third-phase analysis is skipped. As some policies and attributes may be protected and cannot be downloaded, the validity of some candidate concrete compositions may not be verifiable by the service composer. In this case, the fitness value for each candidate \(ch_k\) in \(CH_1\), \(fit(ch_k)\), are recomputed, and the top \(L_2\) is the percentage that ranges from 0 to 1) candidate concrete compositions are selected and included in \(CH_2\).

In the third phase (Section 5.3), any previously selected compositions in \(CH_2\) should be fully validated. For each service pair \((ch_k, s_i, s_j)\) in a candidate concrete composition \(ch_k\) in \(CH_2\), if \(Pol(ch_k, s_i, R)\) is protected, then the service composer needs to forward the attributes of \(ch_k, s_i\), \(Attr(ch_k, s_i)\), to \(dom(ch_k, s_i)\) for remote policy evaluation. If the policy evaluation requires some protected attributes of \(ch_k, s_j\), the service composer needs to initiate a negotiation session in which, \(dom(ch_k, s_i)\) retrieves the protected attributes of \(ch_k, s_j\) from \(dom(ch_k, s_j)\).

In case that the third-phase analysis does not yield a solution from \(CH_2\), the process will rewind to the second phase to choose the next best \(L_2\) candidates in \(CH_1\) and perform third-phase analysis again. If \(CH_1\) does not include a valid composition, then the process will rewind to the first phase and choose the next best \(L_1\) candidate compositions in \(CH_0\) and perform the whole process again. The overall three-phase composition protocol is given in Fig. 5.
the original service chain. Note that $(x, y).tf$ is computed based on the original service chain but the record does not need to keep the original chain information.

We first consider retrieving strongly matched records for $(chk:si, chk:sj)$ to estimate $LL(chk:si, chk:sj)$.

**LLC2.** If there exist service pairs $(x_i, y_i)$ in VDB, s.t. for all $l$, $chk:si = x_i, chk:sj = y_i$, and $tf(<chk:si, ..., chk:sj-1>) = (x_i, y_i).tf$, then $(chk:si, chk:sj)$ and $(x_i, y_i)$, for all $l$, have strong matches, and $LL(chk:si, chk:sj) = Avg\{ (x_i, y_i).vresult \}$. $Avg\{ (x_i, y_i).vresult \}$ computes the average over $(x_i, y_i).vresult$. Note that, we convert true to 1 and false to 0.

If rules LLC1 and LLC2 are not applicable, then we consider retrieving weakly matched records in VDB.

**LLC3.** If there exist service pairs $(x_i, y_i)$ in VDB, s.t. for all $l$, $chk:si = x_i, chk:sj = y_i$, and $tf(<chk:si, ..., chk:sj-1>) = (x_i, y_i).tf$, then $(chk:si, chk:sj)$ and $(x_i, y_i)$, for all $l$, have weak matches with matching level $ml_l = (1 - \Delta tf / max tf)$, where $\Delta tf = tf(<chk:si, ..., chk:sj-1>) - (x_i, y_i).tf$, and $LL(chk:si, chk:sj) = (\sum_{l=1}^{n} (x_i, y_i).vresult: ml_l) / (\sum_{l=1}^{n} ml_l)$. $\Delta tf$ is computed in the first phase, for all $i < j$.

Here, we convert the TF levels into numerical values (HR = 0, MR = 1, LR = 2, NR = 3). Note that max tf is 4.

If rules LLC1 through LLC3 are not applicable, then there is no record in VDB that matches $(chk:si, chk:sj)$. In this case, we set $LL(chk:si, chk:sj) = 1$ (rule LLC4). This way, $chk:si$ and $chk:sj$ are likely to be selected in the first and second phases and validated in the third phase, and hence, the validation results are generated, which may be used in future composition transactions. Note that, if we have selected a concrete composition without service pair $(x, y)$ in the second phase, then we can replace the corresponding service pair with $(x, y)$ and the fitness of the new service chain will be greater than or equal to the fitness of the previous one. We also consider that the service composers may exchange their historical information offline or when there is insufficient information to help evaluate the likelihoods and use the historical information of other service composers to help compute the fitness of candidate compositions. Therefore, it is very unlikely that the rules LLC1 through LLC3 are not applicable.

**LLC4.** If rules LLC1 through LLC3 are not applicable, then set $LL(chk:si, chk:sj) = 1$.

We define $fit(chk) = \Pi_{l=1}^{n} LL(chk:si, chk:sj)$, for all $i, j$, $0 \leq i \leq n, i < j \leq n + 1$. The service composer will rank all candidate compositions based on their fitness and select the top $L_1$ candidates and include them in $CH_1$. The detailed protocol for the first t-phase analysis is given in Fig. 6.

### 5.2 Second Phase Analysis

In the second phase, the service composer computes auth$(Attr(chk:si), Attr(chk:sj), tf(<chk:si, ..., chk:sj-1>), Pol(chk:si,R), Pol(chk:sj,R))$, for all $i, j$, $0 \leq i \leq n, i < j \leq n + 1$, using its cached attributes $Attr(chk:si)$ and/or policies $Pol(chk:si,R)$. If the required information is not in cache, then the service composer downloads $Pol(chk:si,R)$ from $dom(chk:si).sa$ and $chk:sj.ac$ (containing $Attr(chk:sj)$) from $dom(chk:sj).sa$.

**INPUT:** $CH_0, L_1$. **OUTPUT:** $CH_1, LS$.

1. For each candidate composition $chk_i \in CH_0$.
   1) For all $i, j$, $0 \leq i \leq n, i < j \leq n + 1$.
      2) Compute $LL(chk:si, chk:sj)$ using LLC1, ..., LLC4.
   2) Compute $fit(chk) = \Pi_{l=1}^{n} LL(chk:si, chk:sj)$, for all $i, j$, $0 \leq i \leq n, i < j \leq n + 1$.
   3. Set $LS = \{ LL(chk:si, chk:sj) \}$ for all $i, j, k$, $0 \leq i < j < n + 1$.

Fig. 6. First-phase protocol.

To avoid potential frauds, the download of IFC policies and/or attribute certificates needs to be properly controlled. We consider each service composer is associated with a set of attributes. These attributes may include the name of the service composer, the domain of the composer, the trust level of a SA on the composer, the permission granted to the service composer for downloading some policies/certificates from a domain, and so on. The attributes of a service composer must be asserted by a SA. For authentication and nonrepudiation purpose, the asserted attributes are included in an attribute certificate and signed by the issuer. In Definition 5.1, we define the attributes and attribute certificates of service composers.

**Definition 5.1.** Each service composer $scomp$ is associated with a set of attributes $Attr(scomp)$. $scomp$ owns a set of attribute certificates $scomp.AC$. Each attribute certificate, $scomp.ac \in scomp.AC$, is a certificate issued by a SA to certify that $scomp$ holds certain attributes $scomp.ac.Attr \subseteq Attr(scomp)$.

We consider each SA $d_{i,sa}$ defines a set of policies to control the download of the policies and/or certificates of $d_{i,sa}$. Such a policy is called the disclosure policy. We use $d_{i,PolD} = \{d_{ipolD1}, d_{ipolD2}, \ldots \}$ to represent the set of all disclosure policies in domain $d_i$. When a service composer $scomp$ requests for downloading certain protected policies and/or certificates from $d_{i,sa}$, $scomp$ must present one of its attribute certificates, $scomp.ac$, to $d_{i,sa}$. After receiving the download request of $scomp$, $d_{i,sa}$ extracts $scomp.ac.Attr$ from $scomp.ac$ and evaluates $scomp.ac.Attr$ against $d_{i,PolD}$. If the decision is true, $d_{i,sa}$ sends the requested policies and/or certificates to $scomp$. If the decision is false, then the requested policies/certificates cannot be disclosed to $scomp$ and the policy evaluation has to be performed remotely at $d_{i,sa}$.

In the second phase, for each candidate concrete composition $chk_i$ in $CH_1$, the composer refines $LL(chk:si, chk:sj)$ computed in the first phase, for all $i, j$, $0 \leq i < j \leq n + 1$, using cached or downloaded policies $Pol(chk:si,R)$ and/or attributes certificates $chk:sj.ac$ to compute $auth(Attr(chk:si), Attr(chk:sj), tf(<chk:si, ..., chk:sj-1>), Pol(chk:si,R), Pol(chk:sj,R))$ locally. If it is evaluated to true, then $LL(chk:si, chk:sj)$ is set to 1. If it is evaluated to false, then $chk_i$ is removed from $CH_1$. If it cannot be fully evaluated, then the same $LL(chk:si, chk:sj)$ computed in
the first phase will be used. Based on the updates, the service composer reranks the candidates in \( C_{H1} \), and selects the top \( L_{2} \) candidates. The second-phase protocol is shown in Fig. 7.

### 5.3 Third-Phase Analysis

In the third phase, the service composer \( scomp \) contacts the security authorities \( dom(ch_{i}, s_{i}).sa \), for all \( i, 0 \leq i \leq n, \) to perform remote policy evaluation, that is, to compute \( auth(Attr(ch_{i}, s_{i})), Attr(ch_{i}, s_{i}), \) \( tf(<ch_{i}, s_{0}, ..., ch_{i}, s_{j-1}>) \), \( Pol(ch_{i}, s_{i}) \) that have not been validated in the second phase. In this phase, the service composer \( scomp \) may send the cached attributes of \( ch_{i}, s_{i} \), \( Attr(ch_{i}, s_{i}) \), included in the attribute certificate \( ch_{i}, s_{i}.ac \), to \( dom(ch_{i}, s_{i}).sa \). However, some attributes in \( Attr(ch_{i}, s_{i}) \) may be protected by \( dom(ch_{i}, s_{i}).sa \) and cannot be revealed to \( scomp \). In this case, \( dom(ch_{i}, s_{i}).sa \), for all \( i, 0 \leq i \leq n, \) where the evaluation of \( Pol(ch_{i}, s_{i}) \) requires some protected attributes of \( ch_{i}, s_{j} \), for some \( j, i < j \leq n + 1 \), need to negotiate with \( dom(ch_{i}, s_{j}).sa \) to retrieve the protected attributes [13], [17].

A negotiation session includes a startup round, and \( m \) negotiation rounds. The service composer decides the maximal number of negotiation rounds \( M \), and may terminate the negotiation session if \( m \) exceeds \( M \). Note that, \( m \) may be 0, indicating that all attributes required by the remote policy evaluation are already provided by the service composer and, hence, no negotiation is required.

In the startup round, the service composer initiates the negotiation by sending the cached attributes (included in attribute certificates) required by the remote policy evaluation within a special message to all \( dom(ch_{i}, s_{j}).sa \).

In each round of the negotiation, each SA identifies the missing attributes required for the policy evaluation and the security authorities who own these attributes, and sends an attribute request to the service composer. The service composer routes the attribute requests to the designated authorities. On receiving the attribute request from the service composer, each SA evaluates its disclosure policies and may return the requested attributes in an attribute response. The attribute response is also routed by the service composer.

In this paper, we consider a synchronized negotiation protocol in which the service composer acts as the negotiation broker between all security authorities. This is to avoid direct negotiation between all \( dom(ch_{i}, s_{j}).sa, 0 \leq i \leq n \), and all \( dom(ch_{i}, s_{j}).sa, i < j \leq n + 1 \), which results in \( O(n^2) \) negotiation channels. During the negotiation, the service composer reorganizes the packages received from different security authorities, and consolidates the packages with the same destination into one message. This approach can reduce the number of negotiation channels into \( O(n) \). Note that protected attributes should be encrypted during communication. Thus, although the message transfer is through the service composer, the service composer does not know the actual attribute values.

By the end of the negotiation session, either all \( dom(ch_{i}, s_{j}).sa \) retrieves the required attributes for policy evaluation, or the negotiation fails. For the latter case, the service composer will set \( valid(ch_{i}, <s_{0}, ..., s_{n+1}>) \) to false. For the former case, \( auth(Attr(ch_{i}, s_{j}), Attr(ch_{i}, s_{j}), tf(<ch_{i}, s_{0}, ..., ch_{i}, s_{j-1}>) \), \( Pol(ch_{i}, s_{j}) \), for all \( i, j, 0 \leq i \leq n, i < j \leq n + 1 \), will be either true or false and, hence, \( valid(ch_{i}, <s_{0}, ..., s_{n+1}>) \) can be fully decided.

In the third phase, the service composer takes the top candidate concrete composition \( ch_{k} \) in \( C_{H2} \) and may start negotiation for the service pairs in \( ch_{k} \) that have not been fully validated. Then, \( valid(ch_{k}, <s_{0}, ..., s_{n+1}>) \) will have a definite result. If the result is true, \( ch_{k} \) is returned to the user. If the result is false, \( ch_{k} \) is removed from \( C_{H2} \) and the next highest ranked candidate composition will be selected to go through the same validation process. The third-phase protocol is shown in Fig. 8.

### 6 Performance Evaluation

To validate the effectiveness and evaluate the performance of the three-phase composition mechanism, we design a set
of experiments and compare the three-phase approach with the conventional single-phase composition approach. To facilitate the performance comparison and validation, we set up a simulation system to simulate the composition of service chains.

6.1 Experimental Setup

The simulation system includes 80 domains and 400 concrete services. Each domain has a domain ID (1 to 80) and the longitude and latitude. The longitude and latitude are used to generate the communication latency between the service composer and the SA of the domain. We use WS-Sim toolset [22] to generate the longitude and latitude for each domain and use the correlation between communication latency and distance collected from the Internet to generate the simulated latency.

For each concrete service, we first uniformly randomly select its domain (with equal probability of being in any of the 80 domains). Each concrete service instantiates an abstract service and we consider 200 different abstract services (each abstract service is realized by two concrete services). Each concrete service has its own concrete service ID, the ID of the corresponding abstract service, its domain ID, the size of the output, and the service latency.

We also use WS-Sim [22] toolset to generate the service output size and the service latency. The generation of TF for each service is based on the characteristics of the web services in each category in WS-Sim. We first consider two main categories of web services, including the data-centric service category (e.g., data center, digital library, video hosting sites, etc.) and functional service category (e.g., image processing, object recognition, etc.), based on their major functions. Then, we divide each main category into subcategories based on the type of data they process, including video/audio, image, document, and alphanumeric data. For each subcategory, we choose several sample web services and generate TF for each of these services. For example, for the subcategory of functional service with image data type, we consider image enhancement service, image registration service, feature extraction service, and object recognition service. We assume that a functional service with image input/output has equal probability to be in one of these four sample services. For each sample web service, we derive its TF based on its functionality. For example, for image enhancement service and image registration service, the TF is HR. The TF of feature extraction service is MR. The TF of object recognition service is LR. Hence, the TF of a service in the category of functional service with image data type has 50 percent to be HR, 25 percent to be MR, and 25 percent to be LR.

The policy download flag and policy evaluation time are randomly generated to facilitate the simulation of the policy evaluation process. The set of all security policies in each domain is associated with a single policy download flag, which is uniformly distributed between 0 (not downloadable) and 1 (downloadable). We use MatLab to analyze the result of a rule engine scalability test [28] to generate the policy evaluation time (i.e., the time required to evaluate an access control policy). The result shows that the policy evaluation time follows a Gaussian distribution, where \( \mu = 106.61 \) (in milliseconds) and \( \lambda = 169.33 \). This distribution is used to generate the policy evaluation time. To simulate the policy download time, we randomly generate policy sizes.

The generation of policy size \( SP \) follows the equation:

\[
SP = \sum_{1 \leq i \leq K} SR_i \text{, where } K \text{ is the number of policy rules,}
\]

and \( SR_i, 1 \leq i \leq K \) is the size of the \( i \)th policy rule. For simplicity, we assume that \( K \) is uniformly distributed between 1 and 50, and \( SR_i \) is uniformly distributed between 64 and 512 bytes.

Though we simulate the policy sizes and evaluation time, we do not actually perform policy evaluation. Instead, we use a simple rule to generate the validation results to avoid inconsistent validation outcomes in the simulation. The rule is that the access is granted only if the clearance level of the requesting service is greater than or equal to the security class of the requested data object.

One factor that has significant impact on performance is the success rate of the composition tasks. When the success rate is high, the conventional single-phase composition mechanism can easily find out a valid composition, while the three-phase composition process needs to spend extra time on the first and second phases and, hence, the advantage it has cannot pay off the extra overhead. On the other hand, when the success rate is low, the single-phase approach may spend much longer time on finding out a valid composition and, hence, the three-phase method may perform much better. We intend to study the impact of the success rate, but it is difficult to directly control the success rate. Instead, we generate different percentages of public services to indirectly control the success rate.

For simulation purpose, we consider an attribute-based access control system in which each data resource is assigned a security class and each service is assigned a clearance level by each domain. The security class \( sc(r) \) measures the sensitivity level of data \( r \) and also the level of security protection required by \( r \). The security class is defined by multiple levels, including No Protection (NP), Low Protection (LP), Medium Protection (MP), Medium-high Protection (MHP), and High Protection (HP), where \( NP < LP < MP < MHP < HP \). We use \( cl(s, d) \) to represent the clearance level of service \( s \) in domain \( d \). We consider a simple access control policy, that is, a request is only granted if the clearance level (ranging from NP to HP) of the requesting service is greater than or equal to the security class of the requested data. In the simulation, we ignore data resources that do not need protection and generate the security classes for sensitive data resources following a uniform distribution between LP and HP. We use public services ratio (PSR, \( 0 \leq PSR \leq 1 \)) as a parameter, and generate the clearance levels of nonpublic services following a uniform distribution between LP and HP. Note that the simple model here is to ease the simulation system as it will be very difficult to simulate real attribute-based policies and certificates.

The client generates a length-\( n \) abstract service chain by randomly selecting \( n \) different abstract services (uniform distribution) and submits it to the service composer. The composer uses a conventional single-phase method and the three-phase composition process to select concrete services for the abstract service chain.
We also design a caching system to store the downloaded policies and the validation results of historical composition transactions. We consider a simple first-in first-out cache replacement policy.

### 6.2 Experimental Results

We conduct experiments to study the performance of the three-phase composition protocol (P1), the conventional single-phase composition protocol (P2), and the protocol without composition-time access control validation (P3), under different success rates, different service chain lengths, and variant $L_1$ and $L_2$.

In Fig. 9, we compare the performance of P1, P2, and P3 under different success rates. As can be seen, when the success rate decreases, the time required by P3 increases dramatically because the wasted execution efforts become more severe. When the success rate is below 68 percent, the cost of P3 is higher than the single-phase protocol (P2) (Note that in P1 and P2, we need to perform access control validation at composition time, and redo it at execution time to assure security). The performance gain of P1 becomes significant even at a high success rate. When the success rate is 97 percent, P1 is only 3 percent faster than P2 and performs slightly worse than P3. When the success rate is 53 percent, P1 is 24 percent faster than P2 and 37 percent faster than P3.

In Fig. 10, we compare the composition time of P1 and P2 under different service chain lengths at a relatively high success rate (ranging from 88 percent ($\text{length} = 6$) to 74 percent ($\text{length} = 14$)). For shorter service chains ($\text{length} \leq 10$), both methods can find a valid composition quickly. In some cases, the three-phase protocol (P1) may even perform a little worse than P2 due to the extra time spent for fitness calculation. For longer service chains ($\text{length} > 10$), P1 performs much better than P2, from 30 percent improvement at length 11 to 70 percent improvement at length 14. From the growing trend, the performance improvement in the three-phase scheme can be much more significant with increasing problem size.

Fig. 11 shows how the selection of top candidate ratios in the first and second phases, $L_1$ and $L_2$, can impact the performance of the three-phase composition protocol. The result shows that $L_1$ and $L_2$ do not significantly impact the performance. With different $L_1$, the composition time of the three-phase protocol oscillates between 2.3 and 3 sec.

### 7 Related Work

The existing works in web service security generally fall into two research directions. One research direction is the development of execution time access control schemes to secure web services [1], [2], [4], [8], [16], [19], [25], [30]. Wonohoesodo and Tari [25] consider a role-based access control for web services. It treats web service as a special type of data resource, and applies both action-based (control the accesses to service operations) and resource-based (control the accesses to data resources accessed through web services) access control. Bhatti et al. [5] consider dynamically changing security requirements of services and defines contextual parameters, for example, time, location, and so on, to capture such requirements. An access request includes a set of contextual parameters, which is evaluated together with the user information against the policies. Bertino et al. [4] consider an adaptive access control model for web services characterized by variant protection granularities. It allows to divide the domain of the service input into multiple value ranges and to specify different access control policies for different ranges. It also allows to group multiple services into a service class and to specify access control policies for the service class. Paci et al. [19] consider the controlled
dissemination of policy information to users in conversa-
tional web service. It models a service as a finite state
machine in which each transition is associated with a
service operation and a set of access control policies. At
each state, it determines the set of policies that may be
enforced afterwards (the policies for all service operations
that may lead to a final state) and allows only these policies
to be disseminated. Damiani et al. [8] focus on the
protection of SOAP messages, and considers the XPath
(a path in the tree representation of XML-like document
such as SOAP) as the first-class object in the access control
system. Ardagna et al. [2] consider the credential-based
access control with the abstraction of complex concepts, for
example, a set, a disjunction/conjunction, and so on, into a
single concept in policy specification. It also supports
the recursive reasoning on credentials and negotiation.
Agarwal and Sprick [1] use DAML-S to specify access
control lists and includes them in an SPKI/SDSI credential
to facilitate a credential-based access control. The authors
of [1], [30] also consider the composition of access control
policies of individual component services into composite
service level policies and use a centralized entity (e.g., the
workflow execution engine) to enforce the policies. They
focus on the control of the user’s accesses to the component
services but do not consider securing the interactions
between individual component services.

There have been a few works that consider the
information flow problem in composite services [7], [23],
[27]. In [7], [27], interactions between nonconsecutive
services are validated in exactly the same way as that of
consecutive services. The computation effects of intermedi-
ate services are disregarded. For example, in the service
chain \(<s_0, s_1, s_2, s_3, s_4>\), \(s_2, s_3, s_4\) and \(s_4\) are validated as if they
are directly interacting with \(s_0\) (i.e., \(s_1\)). This approach,
though capable of ensuring information flow security, poses
overly restrictive constraints. Without proper privileges, the
composition of \(s_2, s_3, s_4\) and \(s_4\) is prohibited even if \(s_0\)'s output
does not really flow into \(s_2, s_3,\) and \(s_4\). In [23], an extreme
approach is considered. When specifying policies, it needs
to consider all intermediate services. Consider the same
example service chain. To validate the accesses between \(s_0\)
and \(s_1\), it is necessary to consider all possible compositions
of intermediate services between them. As can be seen, this
approach is almost infeasible, considering its complexity for
policy specification.

Another research direction is security-aware service
composition [3], [6], [9], [10], [18]. These systems treat
security as a set of quantitatively measurable attributes, for
example, the type of encryption scheme, the type of
authentication protocol, trust and reputation, and so on.
The security properties of each service are specified in
terms of these attributes. The users and service providers
may specify their security constraints, also in terms of these
attributes, to ensure the secure use of their services and/or
data resources. To achieve secure composition, the service
composer ensures that the security properties of all the
selected concrete services satisfy all the security constraints.
Denker et al. [9] focus on the definition of security
ontologies in DAML+OIL to facilitate the specification of
security properties and constraints. It uses Java Theorem
Prover to match security properties with security con-
straints. It also identifies several situations in which
different levels of match may be achieved, and suggests
using negotiation when an exact match between the
security constraints and security properties cannot be
found. In [6], security properties of services are evaluated
by a trusted authority and certified by SAML assertions
issued by the authority. The validated security properties
are stored in the WSDL document of the corresponding
service. Also, the user-specified security constraints are
included in the SOAP requests, and used to prune the
candidate services prior to the composition process. On
the other hand, the security constraints specified by the
service providers are included in the WSDL documents,
and are validated when allocating a concrete service to an
activity in the workflow. Also, AI planning techniques
are used to achieve service selection and composition. In [10],
the negotiation process is introduced in secure service
composition. Each concrete service has multiple sets of
security properties (which are protected rather than
publicly available), each with a preference level. Prior to
the negotiation, the most preferable sets of security prop-
erties are always loaded to the negotiation agent. When a
mismatch is detected, some services may provide their less
preferable sets of properties to achieve composition. The
recomposition issue is also considered for the case when
the security constraints posed by the service providers or
the user are changed. In [18], each web service is modeled
as a finite state machine, where each transition arc is
associated with a service operation and defined a precondi-
tion (an access control policy). It also models each
composite service as a finite state machine, where each
state includes the states of all its component services and
each transition records the invoker/invitee information
and the service operation to be invoked. It considers that
each service defines a set of access control policies to
specify the credentials required to grant the access and a set
of credential disclosure policies to specify how a specific
credential can be released. The verification of a composition
is to verify, for each transition (in the composite service),
whether the invitee’s credential disclosure policies comply
with the access control policies of the invitee. In [3], the
composition/matchmaking problem is mapped to the type
system of an enriched \(\lambda\)-calculus. It considers security
constraints and properties from the perspective of events.
With a predefined set of events, both security constraints
and properties can be expressed as a temporally ordered
sequence of events. To decide whether a security property
matches a security constraint, one only needs to verify
whether the order of events in the security property
conforms to the security constraint. However, this work
provides very little information about how actual access
control policies can be abstracted in this manner.

8 Conclusion
We have developed an innovative security-aware service
composition protocol with composition-time IFC, which can
reduce the execution-time failure rate of the composed
composite services due to IFC violations. We define IFC
rules based on the concept of transformation factor to guide
the composition process. A preliminary version of this work has been published in [31]. We also develop a three-phase composition protocol, which can quickly eliminate invalid concrete compositions and identify the composition that satisfies the IFC policies. Experimental study confirms the efficiency of the mechanism.

Our approach can greatly reduce the potential access control violations at execution time but cannot guarantee that there are no access control violations at runtime and a suitable execution-time access control model will be needed to provide further protection. Consider a web service with a back-end database. The data resources that may be accessed during the service execution may not be determined at the composition time (nor even at the service invocation time). To control the accesses to the data resources that are further protected, we are going to consider a hybrid information flow analysis approach. Specifically, we plan to use static program analysis techniques to find the data accesses within services that can be determined statically and perform composition-time access control validation with the policies defined for these data resources. For the accesses that can only be determined at runtime, we consider dynamic information flow tracking techniques to track down the flow of sensitive information both inside the service and between services to perform execution-time access control.

ACKNOWLEDGMENTS

This research was supported by the US Air Force Office of Scientific Research under Award No. FA-9550-08-1-0260, the US National Science Foundation (NSF) Industry and University Collaborative Research Center (IUCRC) on Net-Centric Systems and its industrial membership, and the NSF Fundamental Research Program under Award No. IIP-1128270. The authors thank their research sponsors for their support.

REFERENCES

Wei She received the BS and MS degrees from Tsinghua University, China, in 2001 and 2004, respectively, and is working toward the PhD degree from the University of Texas at Dallas. His research interests include web services, access control, and information flow control. He is a student member of the IEEE.

I-Ling Yen is a professor in the Computer Science Department at the University of Texas at Dallas. Her research interests include cloud computing, service computing, high-assurance systems, and security. She had published many technical papers and received many research awards in these areas. She is a member of the IEEE.

Bhavani Thuraisingham is the Louis A. Beecherl, Jr. Distinguished Professor and the Director of the Cyber Security Research Center at the University of Texas at Dallas. She is the recipient of numerous awards and has more than 30 years of experience in the industry, MITRE, the US National Science Foundation, and Academia. Her work has resulted in 100+ journal articles, 200+ conference papers, three US patents, and 12 books. She is an elected fellow of the IEEE, the AAAS, the British Computer Society, the Society for Design and Process Science, and the Society of Information Reuse and Integration (a subcommittee of IEEE Systems, Man and Cybernetics Society).

Elisa Bertino is a professor in the Computer Science Department at Purdue University and the director of the Center for Education and Research in Information Assurance and Security. Her main research interests include security, privacy, database systems, digital identity management systems, distributed systems, and multimedia systems. She has served on the editorial boards of several scientific journals. She received the 2002 IEEE Computer Society Technical Achievement Award and the 2005 IEEE Computer Society Tsutomu Kanai Award. She is a fellow of both the IEEE and the ACM.