Research Article

Distributed Information Flow Verification Framework for the Composition of Service Chain in Wireless Sensor Network

Ning Xi, Jianfeng Ma, Cong Sun, Yulong Shen, and Tao Zhang

School of Computer Science and Technology, Shaanxi Key Laboratory of Network and System Security, Xidian University, China

Correspondence should be addressed to Ning Xi; xining@stu.xidian.edu.cn

Received 28 February 2013; Revised 13 April 2013; Accepted 15 April 2013

Academic Editor: Liangmin Wang

Copyright © 2013 Ning Xi et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

DynamicservicecompositionprovidesuswithapromisingapproachtocooperatedifferentsensornodesinWSNtobuildcomplexapplicationsbasedontheirbasicfunctions.Usuallymultiplenodeslocatedindifferentregionsprovidedatawithdifferentsecuritylevels,andovertimeitiscriticaltoensurethesecurityoftheinformationflowinthecompositeservices.However,thenergylimitednatureofsensor nodes in WSN poses a significant challenge for the centralized information flow verification with which the verification node needs to consume lots of computation and network resources. In this paper, wespecifythesecurityconstraintsforeachservice participant to secure the information flow in a service chain based in the lattice model and then present a distributed verification framework that cooperates different service participants to verify their information flow policies distributively. The evaluation results show a significant decrease on the verification cost of the single verification node, which provides a better load balance in each sensor node.

1. Introduction

WSN is the key enablers for the development of the Internet of Things (IoT), which is responsible for collecting surrounding context and environment information. In a service-oriented WSN [1, 2], multiple sensor nodes with different basic services, for example, data aggregation, data processing, and decoding, can cooperate with each other to develop new applications rapidly. However, because of the variety and regional characteristics of WSN, the data provided by the sensor nodes have different security levels. When services are composed together, data are transmitted among these nodes, respectively, where an operation in a node assigning high-level data to a low-level object would cause the information leakage with a serious impact on the public safety or personal privacy.

For example, a personal-health helper service can be provided for the healthy advice according to the body status and environments data. Most of the former work, mainly focus on the access control of the individual services [3, 4]. But in a service chain, data may be computed from its prior services which can result in the undesired information leakage. When the collection service is completed, the data collected by the wearable sensors and environmental sensors are delivered to the data processing node, such as mobile phone. Healthy information may leak to untrusted third party through the illegal operations during data processing. So the information flow security is one of the major concerns about the service composition in sensor network environments.

One issue in information flow security of the composite service is the dynamic dependence among various objects in different service participants. Accorsi and Wonnemann [5] use Petri nets to represent the workflow and detect information leaks in workflow descriptions based on static information flow analysis. But this work can only validate the information flow in fixed workflow with static input and output dependences. In service-oriented WSN, there are several candidate services with same functions where the dependences between input and output are different from each other. It is necessary for user to select appropriate service for the secure composition of the service chain. She et al. [6, 7] define transformation factors to measure how likely the output depends on the input data in different candidate services. But it is hard to define the LR, MR, and
HR transformation factors. Therefore, a suitable dependence model is required for the analysis of the information flow in different candidate services.

Another major issue for the information flow verification in WSN is the energy cost of the verification node. Zorgati and Abdellatif [8] and She et al. [9] propose the centralized verification approach against the information flow control policies to ensure an end-to-end security in wired network. However, in WSN, the sensor node is energy limited, and the centralized way consumes lots of energy of the verification node. Yildiz and Godart [10] propose an decentralized service composition approach considering the information flow policies in an inexpensive manner, but its policies are static. Based on the information flow type system, Hutter and Volkamer [11] specify the composition rules to control the security of dynamically computed data and their proliferation to other web services. But it costs extra energy of the sensor node to compile the service code again before the service execution.

In this paper, we present a distributed information flow verification approach applied on the composition of the service chain in wireless sensor network. Our contributions include the following. (1) For the dynamic dependences in service chain, we define the intra and inter dependences among different objects in composite service based on the PDG. (2) We specify the security constraints for each service participant based on the dependencies and lattice model. (3) We propose a decentralized information flow verification approach to execute the verification process distributively to provide a better load balance of the sensor nodes in WSN.

The rest of the paper is structured as follows. Section 2 presents the basic definitions of the wireless sensor service system. Section 3 specifies the security constraints for each service participant based on the analysis of the information flow in the service chain. In Section 4, we propose the distributed information flow verification framework based on the secure information flow model. Section 5 evaluates the proposed verification approach. Section 6 concludes the paper.

2. Wireless Sensor Service System

A wireless sensor service system (WSS) is a large-scale distributed environment which consists of multiple wireless sensor nodes, public data resources and security authorities, which is shown in Figure 1. Sensor nodes in WSN can collect these resources, and provide different basic functions, such as data analyzing or processing, which are treated as various services in WSN. There is also a security authority for each data resources for the management of these data security levels which are used for the security verification. The service on each sensor node can be defined as follows.

Definition 1. Each service \( s_i \) is a tuple \( s_i = \langle id_i, \text{In}_i, \text{Out}_i, F_i, C_i \rangle \), where \( id_i \) is the identifier of the service; \( \text{In}_i \) is the set of input of service; \( \text{Out}_i \) is the set of output of service; \( F_i \) is composed of a sequence of actions \( \langle a_1, a_2, \ldots \rangle \); \( C_i \) is the certificate of the service which specifies the security properties of service.

In WSS system, various services are provided by different sensor nodes. These individual services can also be combined together to generate a more powerful service. During the execution of composite service, each service node collects data from its local storage or the public resources, processes the input data, and finally provides results to the sink nodes. On the other hand, these nodes may also update the local storage or store to the public data resources in WSS. A composite service can be denoted as a directed graph, where the vertex is the service component and the edge represents an composition relationship from one service to another. In this paper, we investigate a simplified composite service, the service chain, which is defined as follows.

Definition 2. A service chain \( s_c \) can be represented as a tuple \( s_c = \langle CH, In, Out \rangle \) where \( CH \) is a sequence of services \( \langle s_1, s_2, \ldots \rangle \); \( In \) is the set of input of \( s_c \); \( Out \) is set of output of \( s_c \).

In a service chain \( s_c \), the predecessor of a service \( s_i \) can be denoted as \( s_{i-1} \), and the successor of a service \( s_i \) is denoted as \( s_{i+1} \). \( s_0 \) denotes the node who sends the initial request to \( s_1 \), and \( s_{n+1} \) denotes the sink node who receives the service result from \( s_n \). Figure 2 shows a simple service chain model.

Due to the dynamic and heterogeneous sensor network environment, it is necessary to select appropriate service to satisfy the different requirements including QoS and security. In this paper, we focus on the verification of the information flow security in composite service chain and providing support for the security enforced selection of services in WSN.

3. Secure Information Flow Model

3.1. Security Label Model. For the information with different sensitivities, we use multilevel security labels to describe the security properties of objects \( o \).
Definition 3. Security label model is defined as a lattice \((SL, \leq)\), where \(SL\) is a finite set of security levels that is totally ordered by \(\leq\).

The lattice model is widely used in government or military systems in which the security classes are determined solely from the four security levels: unclassified, confidential, secret, and top secret [12].

For a clear discussion, in this paper, we define that each object \(o\) has a provided and required security level, \(Pr(o)\) and \(Re(o)\), which specifies the read and write permissions possessed by \(o\). The provided security labels of the objects can be given by the data owners, which are specified in certificates. And the required security labels of data objects will be computed according to the dependence of the input and output data.

3.2. Information Flow in Service Component. In a service chain, the information flow through \(s_i\) is shown in Figure 3. We consider a data flow model in which each service may read from a set of input data objects and write to a set of output data objects. The set of input objects of a service \(s_i\) includes all the objects that \(s_i\) receives from its predecessor \(s_{i-1}\) and all data objects obtained from the public data resources or stored in the local storage in sensor nodes. The set of output objects of \(s_i\) includes all the objects that \(s_i\) sends to its successor \(s_{i+1}\) and all the data objects that \(s_i\) updates to the public data resources and the local storage.

For the input information for \(s_i\), there is \(In_i = \{In_i^M, In_i^D, In_i^L\}\), where

(i) \(In_i^M = \{In_{i,1}^M, In_{i,2}^M, \ldots, In_{i,n}^M\}\) is the set of all input objects that \(s_i\) receives from its predecessor \(s_{i-1}\);

(ii) \(In_i^D = \{In_{i,1}^D, In_{i,2}^D, \ldots, In_{i,n}^D\}\) is the set of all input objects from the public data resources;

(iii) \(In_i^L = \{In_{i,1}^L, In_{i,2}^L, \ldots, In_{i,n}^L\}\) is the set of all input objects located in local storage in sensor nodes.

For the output information for \(s_i\), there is \(Out_i = \{Out_i^M, Out_i^D, Out_i^L\}\), where

(i) \(Out_i^M = \{Out_{i,1}^M, Out_{i,2}^M, \ldots, Out_{i,n}^M\}\) is the set of all output objects that \(s_i\) sends to its successor \(s_{i+1}\);

(ii) \(Out_i^D = \{Out_{i,1}^D, Out_{i,2}^D, \ldots, Out_{i,n}^D\}\) is the set of all output objects that \(s_i\) updates to the public data resources;

(iii) \(Out_i^L = \{Out_{i,1}^L, Out_{i,2}^L, \ldots, Out_{i,n}^L\}\) is the set of all output objects that \(s_i\) writes to the local storage in sensor nodes.

In order to validate the information flow in \(s_i\), we need to analyze the relationships between the input and output objects. The output \(Out_i\) is computed from \(In_i\) during the execution of the service function \(F_i\). The syntax of \(F_i\) is defined as follows:

\[
\begin{align*}
  f &::= a; f, \\
  a &::= \text{skip} | \text{input}(in, e) | \text{var} := e | a; a \\
  &\quad | \text{if}(e) \then a \else a \\
  &\quad | \text{while}(e) a | \text{output}(out, e), \\
  e &::= \text{var} | e \text{Re} \\
  R &::= + | - | \leq < .
\end{align*}
\]

A service function consists of a collection of activities, some of which are the control and computation operations, while some of which are responsible for receiving the inputs from different sources \(in\) and producing outputs data to the required objects \(out\). We can establish the program dependence graph (PDG) [13] of \(F_i\) according to its syntax to analyze the relationships among different objects used in \(F_i\). The PDG is defined as follows.

Definition 4. Program dependence graphs (PDG) is a directed graph \((V, \tilde{E})\), where the expressions and the activities in \(F_i\) constitute the nodes of the graph and the edges express data and control dependences. A data dependence represented by an edge \(a \rightarrow a'\) means that the activity \(a\) assigns variable \(x\) which is used in activity \(a'\). A control dependence represented by an edge \(e \rightarrow e'\) means that the execution of \(a\) depends on the value of the expression \(e\), which is typically a branch and loop condition.

Once a program dependence graph PDG \(= \langle V, \tilde{E}\rangle\) has been constructed, program backward slice [14] is used to analyze the dependences among the different objects that are used in activities and expressions in PDG. Here we use \(\text{Dep}(o)\) to represent the obtained dependency set of an object \(o\).

Based on the dependency set \(\text{Dep}(o)\), we can compute output object required security level according to the following equations: for \(\forall u \in \text{Out}_i\),

\[
\text{Re}(u) = \bigcup_{\max} \text{Re}(v), \quad v \in \text{In}_i \land v \in \text{Dep}(u) .
\]

Based on the previous equation, we can obtain the following.

Each service $s_i$ has different levels of inputs and outputs. The value of the input objects with high-level security label may flow to the low-level output objects during the execution of the service and cause the information leak. Therefore, the definition of the secure information flow in service component is given as follows.

Definition 6. The information flow in service component $s_i$ is considered secure if it satisfies that for $\forall u \in Out^M_i$, there are

$$\Pr(u) \geq \max_v \Re(v), \quad v \in In_i \lor v \in \text{Dep}(u).$$

The previous condition provides that there are no lower level objects in public resources and local storage storing the data with higher security level during the execution of each service.

3.3. Secure Information Flow in Service Chain. Consider the service chain $(s_1, s_2, \ldots, s_n)$. The output data sent from $s_i$ to $s_{i+1}$, $Out^M_i$, may be dynamically computed from some data stored by sensor node and public data resources, $In^L_i$ and $In^D_i$, and some data received from $s_{i-1}$, $Out^M_{i-1}$. $Out^M_i$ may be further processed by $s_i, s_{i+1}, \ldots, s_{j-1}$ and computed into $Out^M_{j-1}$ which is delivered to service $s_j$, $j > i$. And the dependence between objects belonging to different service components is considered as the interservice dependence. The interservice dependence set of object $u$, $\text{Dep}_{\text{inter}}(u)$, is defined as follows.

Definition 7. For objects $v \in s_j$ and $u \in s_i$ where $j > i$, $v$ is in $\text{Dep}_{\text{inter}}(u)$ which satisfies one of the following two conditions:

1. $i = j - 1$:
   - $\exists w_1 \in Out^M_i, w_2 \in In^M_j, w_1 = w_2$
   - $(v \in \text{Dep}(w_1) \lor w_1 = v) \land (w_2 \in \text{Dep}(u) \lor w_2 = u)$

2. $i \neq j - 1$:
   - $\exists w \in s_k, i < k < j$
   - $v \in \text{Dep}_{\text{inter}}(w) \land w \in \text{Dep}_{\text{inter}}(u)$

For two adjacent services $s_i$ and $s_j$ where $i = j - 1$, there are four cases that need to be considered. (1) For $v \in Out^M_i = u \in In^M_j$, there is an interservice dependence between $u$ and $v$. (2) For $v \in Out^M_i$ and $u \notin In^M_j$, there is an interservice dependence between them if there are objects $w \in In^M_j$ and $v = w$ that $u$ depends on. (3) For $v \notin Out^M_i$ and $u \in In^M_j$, $u$ externally depends on $v$ if there are objects $w \in Out^M_i$ and $u = w$ that depends on $v$. (4) For $v \notin Out^M_i = u \notin In^M_j$, if there are two objects $w_1 \in Out^M_i, w_2 \in In^M_j$ that $w_1 = w_2$, while data object $u$ in $s_i$ depends on $w_2$, and $w_1$ depends on $v$ in $s_j$, we call $u$ that externally depends on $v$.

For two services $s_i$ and $s_j$ where $j > i + 1$, if there is an object $w$ in $s_k$ that $v$ externally depends on, while $u$ externally depends on $w$, the dependence between $u$ and $v$ is the interservice dependence.

For a service chain $s_1$ where $\text{In}_{s_1} = \bigcup (In^L_i \cup In^D_i)$ and $\text{Out}_{s_1} = \bigcup (Out^D_i \cup Out^M_i), 0 \leq i \leq n + 1$, we use $s_0$ that denotes the start node which sends the initial request to $s_1$, and $s_{n+1}$ denotes the sink node which receives the service results from $s_n$. And we assume that $In^M_0 = \emptyset, Out^D_0 = \emptyset, Out^L_0 = \emptyset, In^D_{n+1} = \emptyset, Out^M_{n+1} = \emptyset$.

Definition 8. The information flow in service chain $s_i$ is considered secure if and only if it satisfies that for $\forall u \in Out^M_i$, there are

$$\Pr(u) \geq \max_v \Re(v), \quad v \in In_i \lor (v \in \text{Dep}(u) \lor v \in \text{Dep}_{\text{inter}}(u))$$

$1 \leq j < i$.

According to the definition of the secure information flow in $s_i$ and $s_1$, we can obtain the following lemmas and theorems.
Lemma 9. In a service chain \( s_i(s_1, s_2, \ldots, s_m) \), \( \forall u \in \text{Out}_i^M \), \( 0 \leq i \leq m \) satisfies

\[
\text{Re}(u) \geq \text{Re}(v)
\]

\( v \in \text{In}_j \land (v \in \text{Dep}(u) \lor \text{Dep}_{\text{inter}}(u)) \),

\[
0 \leq j \leq i.
\]

Proof. First, let \( m = 1 \), then there are two service components \( s_0 \) and \( s_1 \).

For \( \forall u \in \text{Out}_0^M \), Theorem 5 provides that \( \forall v \in \text{In}_0 \land v \in \text{Dep}(u) \), and there is \( \text{Re}(u) \geq \text{Re}(v) \).

And there is no interservice dependence in \( s_0 \), so the lemma is proved.

For \( \forall u \in \text{Out}_1^M \), there are two cases to consider.

Case 1. \( j = 1 \), \( \forall v \in \text{In}_1 \land v \in \text{Dep}(u) \). Theorem 5 provides \( \text{Re}(u) \geq \text{Re}(v) \).

Case 2. \( j = 0 \), \( \forall v \in \text{In}_0 \land v \in \text{Dep}_{\text{inter}}(u) \). In this case, the definition of the interservice dependence provides \( \exists w_1 \) and \( w_2 \) where

\[
w_1 \in \text{Out}_0^M = w_2 \in \text{In}_1^M,
\]

\[
v \in \text{Dep}(w_1) \land w_2 \in \text{Dep}(u).
\]

Theorem 5 provides that

\[
\text{Re}(u) \geq \text{Re}(w_2),
\]

\( \forall u \in \text{Out}_0^M \), \( 0 \leq i \leq n - 1 \); that is, for \( \forall u \in \text{Out}_0^M \), \( 0 \leq i \leq n - 1 \), there are

\[
\text{Re}(u) \geq \text{Re}(v),
\]

\( v \in \text{In}_j \land (v \in \text{Dep}(u) \lor v \in \text{Dep}_{\text{inter}}(u)) \),

\[
0 \leq j \leq i.
\]

And the case that \( m = n \) is proved as follows: for \( \forall u \in \text{Out}_n^M \), there are also two cases to consider.

Case 1. \( j = i \), \( \forall v \in \text{In}_i \land v \in \text{Dep}(u) \). In this case, Theorem 5 provides \( \text{Re}(u) \geq \text{Re}(v) \).

Case 2. \( 0 \leq j < i \), \( \forall v \in \text{In}_j \land v \in \text{Dep}_{\text{inter}}(u) \). In this case, the definition of the interservice dependence provides \( \exists w_1, w_2 \) where

\[
w_1 \in \text{Out}_i^M = w_2 \in \text{In}_n^M,
\]

\[
(\forall v \in \text{Dep}(w_1) \lor v \in \text{Dep}_{\text{inter}}(w_1)) \land w_2 \in \text{Dep}(u).
\]

Theorem 5 provides that

\[
\text{Re}(u) \geq \text{Re}(w_2).
\]

The previous assumption provides that for \( 0 \leq i \leq n - 1 \), there is

\[
\text{Re}(w_1) \geq \text{Re}(v),
\]

and there is

\[
\text{Re}(w_2) = \text{Re}(w_1).
\]

Based on (14), (15), and (16), we can get

\[
\text{Re}(u) \geq \text{Re}(v).
\]

In a conclusion, when \( m = n \), the lemma is proved.

Lemma 10. If the information flow of each service in first \( m \) step of \( s_i \) is secure, \( \forall u \in \text{Out}_i^D \cup \text{Out}_i^L \), \( 0 \leq i \leq m \) satisfies

\[
\text{Pr}(u) \geq \text{Re}(v),
\]

\( v \in \text{In}_j \land (v \in \text{Dep}(u) \lor v \in \text{Dep}_{\text{inter}}(u)) \),

\[
0 \leq j \leq i.
\]

Proof. For \( \forall u \in \text{Out}_i^D \cup \text{Out}_i^L \), there are also two cases to consider.

Case 1. \( j = i \), \( \forall v \in \text{In}_i \land v \in \text{Dep}(u) \). In this case, the secure information flow Definition 6 provides \( \text{Pr}(u) \geq \text{Re}(v) \).

Case 2. \( 0 \leq j < i \), \( \forall v \in \text{In}_j \land v \in \text{Dep}_{\text{inter}}(u) \). In this case, the definition of the interservice dependence provides \( \exists w_1, w_2 \) where

\[
w_1 \in \text{Out}_i^M = w_2 \in \text{In}_n^M,
\]

\[
(\forall v \in \text{Dep}(w_1) \lor v \in \text{Dep}_{\text{inter}}(w_1)) \land w_2 \in \text{Dep}(u).
\]

The secure information flow Definition 6 provides

\[
\text{Pr}(u) \geq \text{Re}(v).
\]

Theorem 5 provides that

\[
\text{Re}(u) \geq \text{Re}(w_2).
\]

And the Lemma 9 provides that

\[
\text{Re}(w_1) \geq \text{Re}(v).
\]

And there is

\[
\text{Re}(w_2) = \text{Re}(w_1).
\]

Based on (20), (21), (22), and (23), we can get

\[
\text{Pr}(u) \geq \text{Re}(v).
\]

In a conclusion, the lemma is proved.

Theorem 11. For a service chain \( s_i \), if the information flow in each service component \( s_i \) is considered secure, the flow in the service chain is secure.

Proof. Let \( m = n + 1 \), and the theorem is proved based on Lemma 10.
4. Distributed Information Flow Verification Framework for Wireless Service Composition

4.1. Information Flow Verification Framework. For a service chain $s = (s_1, s_2, \ldots, s_n, s_{n+1})$, there are several candidate services but different implementations by developers for each service step $s_i$. In the distributed information flow verification framework, each sensor node is only responsible for validating its next-step candidate service node $s_{i+1}$, which can balance the energy cost on a single verification node. The distributed information flow verification framework is shown in Figure 4.

In our framework, Service Authorization Centre (SAC) is a trusted third party for service certificate generation before the deployment of the sensor node. There are two phases for the verification of the information flow: service certificate setup and service verification phase. The service certificate that specifies the security properties of the service, that is, the dependence between the service input and output, is first generated and signed by a SAC. During the service composition procedure, the service composer obtains the required service certificates, and verifies the information flow in candidate nodes. These two phases are detailed in the following sections.

4.2. Service Certificate Setup. Service certificate setup is the preparation phase of the verification process, which is shown in Figure 5. In this phase, service developer submits authorization request containing service function code in service node to SAC. And then the generated service certificate $Ce$ is installed on the sensor node with the service. Considering the complexity and security of the service code transmission, the authorization process is executed by the offline mode between the service developer and SAC, which does not need to consume extra energy of the sensor node.

Definition 12. A service certificate $Ce$ is a tuple $(CA, s, De)$, where $CA$ is the issuer, that is, SAC; $s$ is the service identifier; $De$ is the set of statements that describe the output data dependence.

The service certificate $Ce$ specifies the attributes of the service including the service identifier, the dependence between input and output objects in the service function. According to the PDG construction of service function, SAC uses the algorithms presented in [13] to generate the PDG. Once a program dependence graph $PDG = (V, E)$ has been computed, a dependence set can be established for each node $x \in V$ by using intraprocedural backward slice [14], written $De(x)$ containing the set of all nodes in PDG from which $x$ can be reached as follows: $De(x) = \{ y \mid y \rightarrow^* x \}$. In this paper we mainly consider the dependences between the input and output objects in the PDG nodes; that is, $De_{in}(Out_{ik}) = \{ v \mid v \in De(Out_{ik}) \land v \in In_{i+1}, 0 \leq i \leq n, 0 \leq k \leq |Out_{i+1}|$.

For each $Out_{ij} \in Out_i$, its input dependence is written into the certificate. Finally, the certificate is signed by SAC and sent to the service node. Then the service certificate setup phase is complete. The Algorithm 1 is shown as follows.

When there is a request for the service, the node needs to send its certificate to the composer for its information flow verification. The provided security levels of the public and local input data and output objects are also required to be sent to the verification node. If the realization of the service is changed, for example, a new version service is published, the service needs to be authorized by SAC again and reinstalled on the sensor node.

4.3. Service Verification. Service verification is a vital phase in which the verification node requires the service certificates and validates the candidate nodes against the information flow control policies. The verification procedure is shown in Figure 6. During the verification process, service composer $s_{i-1}$, required for the service certificate and the provided security levels of the public and local data and objects first. Then the composer computes the required security levels of the output objects and then validates whether they satisfy the security constrains.

4.3.1. Required Security Level Computation. According to the secure information flow definition in service chain, the required security levels of the data objects need to be computed first. The required security levels of the objects in each service $s_{i,j}$ are computed according to the following three computation rules (CR):

- $CR_1$ For $\forall u \in In_{i,j}^D \cup In_{i,j}^L$, $Re(u) = Pr(u)$;
- $CR_2$ For $\forall u \in In_{i,j}^M$, $Re(u) = Pr(v)$ where $v \in Out_{i-1,j} \land v = u$;
- $CR_3$ For $\forall u \in Out_{i,j}$, $Re(u) = \bigcup_{v \in In_{i}} Re(v) \land v \in Dep(u)$.

$CR_1$ specifies that the required security levels of the input objects from public and local storage are equal to their provided security levels. $CR_2$ specifies that the required security levels of the input objects from predecessor are determined by that of the output objects in $s_{i-1}$. $CR_3$ specifies that the required security levels of the output objects are computed from that of the input objects that the output depends on.

4.3.2. Service Verification. During the service verification, the information flow control policy (IFCP) specifies how to validate a candidate service $s_{i,j}$. Based on the security label model and the definition of the secure information flow in each service, we define the information flow control policies in each service $s_i$ as follows:

- $IFCP_1$ For $\forall u \in Out_{i,j}^D$, $Pr(u) \geq Re(u) \Rightarrow true$, $Pr(u) < Re(u) \Rightarrow false$;
- $IFCP_2$ For $\forall u \in Out_{i,j}^L$, $Pr(u) \geq Re(u) \Rightarrow true$, $Pr(u) < Re(u) \Rightarrow false$.

Based on the required security level computation rules and information flow control policies, verification node can validate the candidate sensor node $s_{i,j}$ in a service chain. The Algorithm 2 is shown as follows.
4.4. Decentralized Information Flow Verification Algorithm for the Service Chain. For each step verification, verification node obtains the passed candidate service set \( S_{p,i} \), then the verification node will notice these passed sensor nodes to verify the following candidate services. And there are three types of messages for the synchronization of the verification procedure, that is, \textit{start message}, \textit{success message}, and \textit{fail message}. \textit{start message} is used to allow the candidate service \( s_{i,j} \) to execute the \textit{Verify_ServiceChain()} procedure. When the nodes in service chain all pass the service verification process, \textit{success message} with the executable path is sent to inform its requestor \( s_0 \). During each step verification \textit{fail message} will be sent to the predecessor of the verification node when there are no candidate services passed.

**Algorithm 1: Service_Certificate_Set_Up().**

- **Input:** Service \( s \langle id_i, dom_i, In_i, Out_i, F_i, Ce_i \rangle \).
- **Output:** Service certificate of \( Ce_i \langle CA, s, De \rangle \).

1. \( Var (x) \) represents the variables objects in \( x \) statement
2. \( Ce_i \cdot s = id_i \)
3. PDG \( \langle V, E \rangle \)
4. generatePDG(\( F_i, G \))
5. for each output node \( x \in G \wedge Var(x) \in Out_i \) do
6. \( BS(x) = \text{backwardSlice}(x) \)
7. for each \( y \in BS(x) \) do
8. if \( Var(y) \in In_i \) then
9. pushInto(\( Ce_i \cdot De(Var(x), Var(y)) \))
10. end if
11. end for
12. end for
13. signature(\( Ce_i \cdot CA, SAC \))
14. return \( Ce_i \)

**Algorithm 2: Service_Verification().**

- **Input:** \( \text{Re}(Out_{i+1}^M), \text{Candidate Service Set } S_i, \text{Pr}(In_i^D), \text{Pr}(In_i^L), \text{Pr}(Out_i^D), \text{Pr}(Out_i^L) \)
- **Output:** Passed Service Set \( S_{p,i} \)

1. \( \text{exOutput}_i \) represents \( In_{i,j}^M \)'s corresponding output in its predecessor
2. \( \text{filterService}(S_{p,j}, S_i) \) represents filtering the unsatisfied candidate service \( s_{i,j} \) from \( S_p \)
3. \( S_{p,i} = S_i \)
4. for each \( s_{i,j} \in S_i \) do
5. requestCert(\( s_{i,j}, Ce \))
6. for each \( u \in In_{i,j}^D, v \in In_{i,j}^L, w \in In_{i,j}^M \) do
7. \( \text{Re}(u) = \text{Pr}(u) \)
8. \( \text{Re}(v) = \text{Pr}(v) \)
9. \( \text{Re}(w) = \text{Re}(\text{exOutput}(w)) \)
10. end for
11. for each \( u \in Out_i^D, v \in Out_i^L \) do
12. for each \( w \in De(u) \) do
13. \( \text{Re}(u) = \cup_{\text{max}} \text{Re}(w) \)
14. end for
15. if \( \text{Pr}(u) < \text{Re}(u) \) then
16. \( \text{filterService}(S_{p,j}, S_i) \)
17. break;
18. end if
19. for each \( w \in De(v) \) do
20. \( \text{Re}(v) = \cup_{\text{max}} \text{Re}(w) \)
21. end for
22. if \( \text{Pr}(v) < \text{Re}(v) \) then
23. \( \text{filterService}(S_{p,j}, S_i) \)
24. break;
25. end if
26. end for
27. end for
28. return \( S_{p,i} \)
Input: $\text{Re}(\text{Out}_{i-1}^M)$, Candidate Service Set $S_{i+1}$.
Output: Secure Execution Path $P$

(1) if $i \neq 1$ then
(2) wait start message
(3) end if
(4) if $i = n$ then
(5) send success message with secure execution path $P$ to the requestor
(6) else
(7) push $S_{i,j}$ into the Execution Path $P$
(8) $S_{p,i+1} = \text{Service Verification}(\text{Re}(\text{Out}_{i-1}^M), S_{i+1})$
(9) if $S_{p,i+1} = \emptyset$ then
(10) send fail message to its predecessor
(11) else
(12) for each $s_{i+1,k} \in S_{p,i+1}$ do
(13) send start message to $s_{i+1,k}$
(14) end for
(15) end if
(16) end if

Algorithm 3: Verify_ServiceChain().

5. Experiments and Evaluations

This paper studies distributed information verification framework for the service composition in WSN. Through the security analysis in Section 3, the information flow security can be ensured by the Theorem 11. In this section, we investigate the impact of distributed service verification on the sensor node’s cost including verification time and communication effort. A centralized approach implements the service verification work by a single sensor node. We test both approaches with NS-3 [15] in multiple scenarios. Table 1 shows further details about the simulation configuration.

Figure 7 shows the computation time on the verification node. In the centralized way, time rises vastly with the increase of the candidate service number. That is because the execution paths that need to be verified are increased at an exponential rate. However, time increases slowly in the distributed way because there is no significant variations on the candidate nodes that each sensor node needs to verify.

Figure 8 shows the communication effort on the verification node. In Figure 8, the communication effort in the centralized way is evidently higher than that used the distributed way. That is because the single verification node needs to
Service composer $S_{i-1}$

Service certificate

Data security authority

Service certificate

Service certificate

Sensor node

(i) Authorization request

(ii) Cert generation

(iii) Service certificate

(iv) Service installation

(vi) Send result to $S_{i,j}$

(v) Verification request

(vector)

Candidate service $S_{i,j}$

(vi) Send result to $S_{i,j}$

Figure 6: Service Verification Phase.

Table 1: Simulation Configuration.

<table>
<thead>
<tr>
<th>General</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulator</td>
<td>NS-3</td>
</tr>
<tr>
<td>Field ($m^2$)</td>
<td>$100 \times 100$</td>
</tr>
<tr>
<td>Radio type</td>
<td>Zigbee</td>
</tr>
<tr>
<td>Service step</td>
<td>4</td>
</tr>
<tr>
<td>Simulation duration (s)</td>
<td>1000</td>
</tr>
<tr>
<td>Random</td>
<td></td>
</tr>
<tr>
<td>Node placement</td>
<td></td>
</tr>
<tr>
<td>Node movement</td>
<td></td>
</tr>
<tr>
<td>Security level</td>
<td>Unclassified, confidential, secret, and top secret</td>
</tr>
<tr>
<td>Controlled</td>
<td></td>
</tr>
<tr>
<td>Candidate number</td>
<td>5–10</td>
</tr>
<tr>
<td>Verification mode</td>
<td>Centralized, decentralized</td>
</tr>
</tbody>
</table>

Figure 7: Computation time on the verification node(s).

Figure 8: Communication effort on the verification node(s).

6. Conclusion

In this paper, we specify the security constraints for each service participant based on the partial order model and propose a decentralized information flow verification approach that cooperates each sensor node to verify the information flow security distributively and builds up secure service chains in wireless sensor environments. Through the simulation on NS-3, the result shows that this approach can decrease the cost of the sensor nodes effectively.

Acknowledgments

This work is supported by Program for the Key Program of NSFC-Guangdong Union Foundation (U1135002), Major National S&T Program (2011ZX03005-002), National Natural Science Foundation of China (60872041, 61072066), and the Fundamental Research Funds for the Central Universities (JY10000903001, JY10000901034, and KS051203010).
References


