entropy

MDPI

Article

Secure Service Composition with Quantitative Information Flow Evaluation in Mobile Computing Environments

Ning Xi *, Jing Lv, Cong Sun and Jianfeng Ma

School of Cyber Engineering, Xidian University, Xi'an 710071, China
* Correspondence: nxi@xidian.edu.cn

Received: 31 May 2019; Accepted: 30 July 2019; Published: 1 August 2019

Abstract: The advances in mobile technologies enable mobile devices to cooperate with each other to perform complex tasks to satisfy users' composite service requirements. However, data with different sensitivities and heterogeneous systems with diverse security policies pose a great challenge on information flow security during the service composition across multiple mobile devices. The qualitative information flow control mechanism based on non-interference provides a solid security assurance on the propagation of customer's private data across multiple service participants. However, strict discipline limits the service availability and may cause a high failure rate on service composition. Therefore, we propose a distributed quantitative information flow evaluation approach for service composition across multiple devices in mobile environments. The quantitative approach provides us a more precise way to evaluate the leakage and supports the customized disciplines on information flow security for the diverse requirements of different customers. Considering the limited energy feature on mobile devices, we use a distributed evaluation approach to provide a better balance on consumption on each service participant. Through the experiments and evaluations, the results indicate that our approach can improve the availability of composite service effectively while the security can be ensured.

Keywords: quantitative information flow; secure information flow model; service composition; mobile computing

1. Introduction

With the development of intelligent terminal, 5G and IoT technologies, various mobile applications enrich our daily lives with more flexible and convenient IT services delivery [1,2]. Moreover, high speed processors and stable connections enable the efficient service interactions among different mobile devices. Based on service-oriented architecture, service composition across multiple mobile devices provides a promising way for integrating several distributed services to satisfy users' complex requirements [3]. Most works focus on improving the efficiency and availability of composite services in mobile computing [4–6]. However, various data with different sensitivities and heterogeneous systems with diverse security policies pose a great challenge on information flow security during the service composition across multiple devices [7]. In particular, if one service component contains malicious code or vulnerabilities, customers' sensitive data may be leaked. In addition, illegal providers or attackers may collude together to eavesdrop private data more effectively based on feedback from different components, in which private data may also be leaked even if individual service is protected by an access control mechanism [8,9].

In order to prevent the data leakage during service composition, types of information flow mechanisms are proposed including type of system [10], model checking [11,12], program static

analysis [13,14] and real-time monitoring [15,16]. Considering the limited energy and dynamic composition relationships, we propose a distributed information flow verification framework for secure service composition in mobile computing environments [7]. Although these approaches provide a solid assurance on information flow security of composite service, implementing them in a real application is still a challenge. These approaches are based on a qualitative discipline, i.e., non-interference [17], which strictly limits the complete absence of any causal flow from high-level sources to low-level sinks. Too strict discipline causes the loss of service availability on account of the security limitations on the cross-level operations in program. It may also cause a high failure rate on service composition because few services can satisfy the discipline. In fact, it is usually permitted in practice to tolerate some leakage for a better service availability. For example, the area of our location may be allowed to be observed by mobile service providers for a customized and more precise route planning. Therefore, for a better balance on service security and availability, it is important for us to measure "how much" information is leaked and "how many" leaks are allowed by customers during the service composition.

In order to quantifying the leakage, many quantitative information flow approaches are proposed based on Shannon's information theory [18]. The authors in [19,20] propose the approach to quantify interference in a simple imperative language for the information flow verification. The authors in [21] present an automatic method for information-flow verification that discovers what information is leaked and computes its comprehensive quantitative interpretation. The authors in [22] establish a tight bound on the maximum leakage from repeated independent runs. However, these approaches mainly focus on a single program that works in a centralized way. During the service composition, there may be several services with similar functions but developed by different mobile service providers, which requires us to select appropriate services for optimized performance [6,23]. It would be a resource-consuming work to evaluate all possible services by a single piece of equipment, which is hard to be implemented due to the energy-limited features of mobile terminals. In addition, all candidate services must be reevaluated even if a small change of one service occurs, which also increases the evaluation load on mobile devices.

In this paper, we present a distributed quantitative information flow evaluation approach applied on the service composition in a mobile computing environment. Our contributions mainly include: (1) we make the quantifying rules on information flows based on the static analysis; (2) we propose the quantitative definition on secure information flow in composite services and specify the security constraints on each service component for distributed evaluation; (3) we design a distributed quantitative information flow evaluation framework and approach for secure service composition in mobile environment, which can provide a better service availability and load balance with affordable costs.

The rest of the paper is structured as follows. Section 2 presents the basic models of the mobile service system. Section 3 details the quantifying rules and the security theorems based on the static analysis. In Section 4, the distributed quantitative information flow evaluation approach is proposed according to the security theorems. Section 5 evaluates our proposed approach. Section 6 concludes the paper.

2. Mobile Service System

2.1. System Model

As shown in Figure 1, the Mobile Service System (MSS) is a distributed IT system consisting of multiple network domains. A domain can have various types of resources, such as data, information, and other physical resources. Mobile terminals in the domain can use these resources and its application functions to provide various services to users, e.g., s_1, s_2, etc. These services can be composed together for a more complex users' application. Moreover, there are several candidate services that can execute the similar functions for a given service. These services can be developed by different service providers.

For example, s_1 can be provided by A, B or other service providers, i.e., $s_{1|A}$, $s_{1|B}$ and so on. In addition, there is also a security authority in each domain for the security management in the domain.

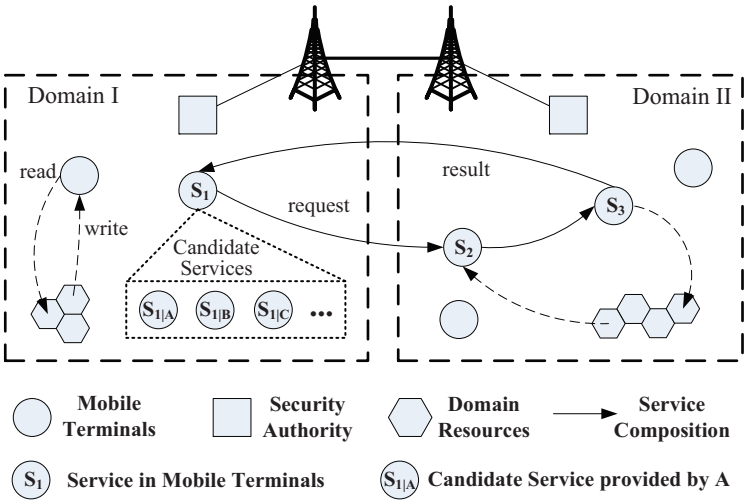

Figure 1. Mobile service system.

Referring to the system model in [7], each domain D can be represented as $D = \langle S, R, SA \rangle$, where S is the set of various services, i.e., $S = \{s_0, s_1, \dots\}$; R is the set of physical resources that can be collected by mobile services in the domain, e.g., environment data, traffic data and so on; SA is the security authority. Each service s_i in S is defined as a tuple $s_i = \langle id_i, dom_i, In_i, Out_i, Pg_i, Ce_i \rangle$, where id_i is the identifier of the service provider; dom_i is the domain that s_i belongs to; In_i is the set of inputs in s_i; Out_i is the set of the outputs in s_i; Pg_i is the program of s_i, which describes the execution procedure of s_i; Ce_i is the certificate of the service which specifies the security properties.

Due to the user's complex requirements, different services s_i in multiple domains may be composed together to achieve the service goal. In this paper, we investigate a typical composite service, i.e., the service chain S_{ch} [7], as shown in Figure 2. A service chain is widely used in service composition because of its simplified composition structure which is easy to deploy and control. In service chain S_{ch}, s_0 receives the request from user and starts the composition procedure. Then, each service s_i, $0 < i < n$, receives the intermediate result as the inputs from its unique predecessor s_{i-1}, executes the service program Pg_i and outputs the intermediate result to its unique successor s_{i+1}. Finally, the last one s_n sends the final results to the user. During the execution, service providers (SP) can input and obtain some data according to the service request, which may cause the leakage of a user's private information.

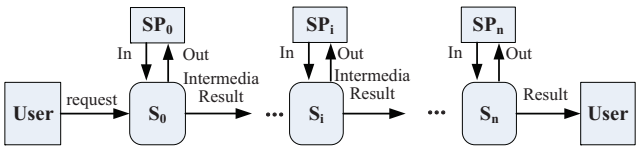

Figure 2. Service chain model in mobile service system.

2.2. Threat Model

Based on our system model, we make the following assumptions about the security capabilities of the participants in MSS.

• **User**: The user is the data owner who has access to all inputs and outputs including public and private information. We partition them into two sets: $L(low)$ for public data and $H(high)$ for private data. In addition, users don't intentionally collude to leak the private data.

• **Service Providers**: Service providers from different mobile devices are honest but curious. They execute the service functions in accordance with their descriptions. They can not access users'

high-level data directly due to the privacy policies, but they can freely observe the public data including all low-level inputs and outputs before and after (but not during) the service's execution. Some of them may try to analyze the value of the private data based on users' low-level inputs and outputs on purpose. In addition, different services may collude together to analyze a user's private data more effectively.

For a clear description, we define LIn_i and $LOut_i$ as the public inputs and outputs with low-level security L. HIn_i and $HOut_i$ are defined as the private inputs and outputs with high-level security H. Then, we can obtain that $In_i = LIn_i \bigcup HIn_i$ and $Out_i = LOut_i \bigcup HOut_i$.

• **SA**: Security Authority is the trusted third party that executes the security function honestly without any interception and manipulation.

3. Quantitative Information Flow Model for Service Composition in a Mobile Computing Environment

3.1. Quantitative Information Flow Model Based on Information Theory

Shannon's theory provides a standard measurement on information quantity known as self-information or entropy. For random variable X for storing different data $x \in X$ in service program Pg_i, its entropy can be defined as [24]

$$\mathcal{H}(X) = \sum_x p(x) \log \frac{1}{p(x)}, \tag{1}$$

where X is a random variable, $p(x)$ is shorthand for $P(X = x)$, which is the probability of $X = x$, and the sum is over the range of X. In Equation (1), the base for log is conventional to use base 2 for the analysis in computer program.

The conditional entropy can be used to represent the amount of information carried by X give the knowledge of variable Y, which is defined as:

$$\mathcal{H}(X|Y) = \sum_y p(y)\mathcal{H}(X|Y = y), \tag{2}$$

where $\mathcal{H}(X|Y = y) = \sum_x p(x|y) \log \frac{1}{p(x|y)}$, and $p(x|y)$ is the probability that random variable $X = x$ given that random variable $Y = y$.

Based on the information quantity on each variable, the mutual information provides a general way of measuring the amount of information stored by X that can be learned by observing another random variable Y, which is defined as:

$$\begin{aligned} \mathcal{I}(X;Y) &= \sum_x \sum_y p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \\ &= \mathcal{H}(X) + \mathcal{H}(Y) - \mathcal{H}(X,Y) \\ &= \mathcal{H}(X) - \mathcal{H}(X|Y) \\ &= \mathcal{H}(Y) - \mathcal{H}(Y|X). \end{aligned} \tag{3}$$

According to our system model, LIn and $LOut$ are the low-level inputs and outputs that service providers can observe during the service composition. Based on Equation (3), for each $X \in In$, the leakage through the flow from X to $Y \in Out$ can be defined as

$$\mathcal{F}_{LIn}(X \rightsquigarrow Y) = \mathcal{I}(X; YlIn) = \mathcal{H}(XlIn) - \mathcal{H}(X|Y, LIn). \tag{4}$$

We use $\mathcal{F}_{LIn}(X \rightsquigarrow LOut)$ to represent the overall leakage of X through all different flows from X to any Y in $LOut$. For a clear description, we assume all the inputs and outputs are k bits variables and

the inputs are uniformly distributed and independent from each other in the following calculation. Then, we can derive the first basic Quantifying Rule (**QR**) as follows:

QR 1. $\forall X \in In, Max(\mathcal{F}_{LIn}(X \rightsquigarrow LOut)) = \mathcal{H}(X) = k$ *and* $Min(\mathcal{F}_{LIn}(X \rightsquigarrow LOut)) = 0$.

Then, we can obtain the quantitative definition on information flow security in a service as follows:

Definition 1. $\forall X \in HIn$ *in a service, the flows in service are K-secure, for* $0 \leq K \leq k$, *if*

$$\mathcal{F}_{LIn}(X \rightsquigarrow LOut) < K,$$

where LIn and LOut are the low-level input and output observations, and K is the security threshold that depends on a user's requirement and the system running environment.

According to Definition 1, we can derive the following two facts: (1) if $K = 0$, it requires that there is no flow from X to any Y in $LOut$, which becomes the qualitative definition of standard non-interference as shown in [25]. (2) if $0 < K \leq k$, it is considered secure if there is at least $k - K$ unknown bits for service providers. A user can choose different thresholds for diverse security requirements in different running environments.

3.2. Quantifying the Information Flow in Service Components

Clark et al. [20] propose the basic analysis rules to quantify leakage for a software program with sequence, branch and loop structures. These rules are specified for the single program analysis in a centralized way, which aim at quantifying the overall leakage instead of each flow's leakage. Therefore, they don't support distributed quantifying across multiple services. Based on some basic rules in [20], we design the improved quantifying rules on different information flows through the static analysis. Our rules are based on the worst case assumption for guaranteeing the security. First, for the s_i's program Pg_i, its syntax can be defined as follow by referring to [20]. It is a simple imperative language including all basic notions, operations and structures in a program:

$$\begin{aligned}
&C \in Com \quad var \in Var \quad E \in Exp \quad B \in BExp \quad const \in \mathbb{N}\\
&P ::= P; C \mid C\\
&C ::= \textbf{skip} \mid var := E\\
&\quad \mid \textbf{if}(B)\ \textbf{then}\ C\ \textbf{else}\ C'\\
&\quad \mid \textbf{while}(B)\ C\\
&E ::= var \mid const \mid E + E' \mid E * E'\\
&B ::= E\ R\ E' \mid \neg B \mid B \wedge B' \mid B \vee\ B'\\
&R ::= < \mid > \mid ==.
\end{aligned}$$

There are two kinds of flows to consider, i.e., explicit and implicit flow [17]. The explicit flow occurs as a result of executing the assignment statement. For example, for the statement $var' := E$, if E contains variable var, there is an explicit flow between var and var', namely $var \rightarrow var'$. Implicit flow occurs as a result of executing a statement C or not when this statement C is conditioned on the value of an binary expression B. This type of flow usually exists in the branch and the loop structures. If B contains var and var' appears in C or C' as an objective variable, there is an implicit flow between var and var', namely $var \overset{B}{\dashrightarrow} var'$. Based on the basic dependence and its transitivity, we can define the intra flows from var to var' as follow, which is represented as $\delta(var, var')$.

Definition 2. $\forall var, var' \in Var$ *in* s_i, *and there are four cases to consider:*
(1) $\exists v \in Var$ *that satisfies* $var \rightarrow v$ *and* $v \rightarrow var'$, *then* $\delta(var, var') = var \rightarrow var'$.

(2) $\exists v \in Var$ *that satisfies* $var \rightarrow v$ *and* $v \overset{B}{\dashrightarrow} var'$*, then* $\delta(var, var') = var \overset{B}{\dashrightarrow} var'$.
(3) $\exists v \in Var$ *that satisfies* $var \overset{B}{\dashrightarrow} v$ *and* $v \rightarrow var'$*, then* $\delta(var, var') = var \overset{B}{\dashrightarrow} var'$.
(4) $\exists v \in Var$ *that satisfies* $var \overset{B}{\dashrightarrow} v$ *and* $v \overset{B'}{\dashrightarrow} var'$*, then* $\delta(var, var') = var \overset{B}{\dashrightarrow} var'$.

Here, we use $var \rightsquigarrow var'$ to represent all the flows from var to var'. According to our attacker model, the attacker can observe the low-level inputs and outputs before and after the execution of service. Thus, we mainly focus on the flows between the inputs and outputs. In addition, we can also obtain that $In_i \subseteq Var$ and $Out_i \subseteq Var$. In order to analyze these flows among the different inputs and outputs in s_i, we construct the PDG (Program Dependence Graph) first [26], and then use the program slicing [27] to obtain all the flows from inputs to outputs based on Definition 2. Then, we define $\boldsymbol{F}_i = \{\delta(X, Y) | X \in In_i, Y \in Out_i\}$ as the set of all intra flows in s_i for the following calculation.

Based on the definition of $\boldsymbol{F}_i$, we can derive the following quantifying rules on the leakage of each flow in s_i. In this paper, we consider the worst case assumptions in which we focus on the computation of the upper bound of leakage for a strong security assurance.

QR 2. $\forall X \in HIn_i$ *and* $Y \in LOut_i$ *satisfy* $X \rightarrow Y$*, then we have*

$$\mathcal{F}_{LIn_i}(X \rightarrow Y) = \mathcal{H}(X) = k, \tag{5}$$

where LIn_i *is the low level input observations in* s_i.

QR 3. $\forall X \in HIn_i$ *and* $Y \in LOut_i$ *satisfy* $X \overset{B}{\dashrightarrow} Y$*, then we have*

$$\mathcal{F}_{LIn_i}(X \overset{B}{\dashrightarrow} Y) = \begin{cases} 1 & B ::= E < E' \mid E > E' , \\ \mathcal{F}^{Eq} & B ::= E == E' , \\ \mathcal{F}^{B} & B ::= \neg B \mid B \wedge B' \mid B \vee B', \end{cases} \tag{6}$$

where $\mathcal{F}^{Eq} = \mathcal{F}_{LIn_i}(X \rightsquigarrow E) + \mathcal{F}_{LIn_i}(X \rightsquigarrow E')$, $\mathcal{F}^{B} = \mathcal{F}_{LIn_i}(X \overset{B}{\dashrightarrow} Y) + \mathcal{F}_{LIn_i}(X \overset{B'}{\dashrightarrow} Y)$.

QR 2 is used to analyze the leakage of the explicit flow. It is easy to follow that, when there is an explicit flow from X to Y, we regard this as all the information of X having been delivered to Y based on knowledge of LIn_i, i.e., $\mathcal{H}(X|Y, LIn_i) = 0$.

QR 3 is used to analyze the leakage of the implicit flow, which includes the following three cases.

(1) For the basic boolean expressions $(E < E')$ or $(E > E')$, we consider the worst case in which the value of B and E' can be observed based on the knowledge of Y and Z. Then, attackers can deduce one more bit information about X in E at most after the service execution, which complies with $1 - Bit$ rule in [20].

For example, for the following program in which x is the 5 bits high level input ranging from -16 to 15, y is the low level output,

$$\{\text{state } \mathbf{s}\}\mathbf{if}(x < z) \textbf{ then } y = 0 \textbf{ else } y = 1\{\text{state } \mathbf{s'}\}.$$

If attackers know the value of z, then he can deduce if the value of x is greater or less than z through the output value of y. Based on information theory, the entropy of x in state s and s' can be calculated as follows:

$$\begin{aligned} &\mathcal{H}(x_s|z_s) = 5, \\ &\mathcal{H}(x_s|z_s, y_{s'}) = P_{x<z}\mathcal{H}(x_s|z_s, y_{s'} = 0) + \\ &\qquad\qquad (1 - P_{x<z})\mathcal{H}(x_s|z_s, y_{s'} = 1). \end{aligned}$$

In addition, we can also get that $\mathcal{H}(x_s|z_s, y_{s'})$ is minimum when $z = 0$, i.e., $Min(\mathcal{H}(x_s|z_s, y_{s'})) = (1/2)log(16) + (1/2)log(16) = 4$. Then, the attacker can obtain one bit of information about x at most through this flow.

(2) For the equality expression $(E == E')$, it is a special case in which service providers may obtain all bits of X in E or E' when this expression is true. In this case, the leakage depends on how much information leaked from X to E and E', i.e., $\mathcal{F}_{LIn_i}(X \rightsquigarrow E) + \mathcal{F}_{LIn_i}(X \rightsquigarrow E')$.

(3) For the complex expressions $(\neg B)$, $(B \wedge B')$ and $(B \vee B')$, the leakage of X depends on the quantity of leakage on each condition B and B', i.e., $\mathcal{F}_{LIn_i}(X \overset{B}{\dashrightarrow} Y) + \mathcal{F}_{LIn_i}(X \overset{B'}{\dashrightarrow} Y)$.

These quantifying rules are consistent with the rules in [20]. Based on the quantifying on explicit and implicit flows, we can calculate the overall leakage from X to Y through different flows by the following rules.

QR 4. *$\forall X \in HIn_i$ and $Y \in LOut_i$ satisfy $X \rightsquigarrow Y$, then we have*

$$\mathcal{F}_{LIn_i}(X \rightsquigarrow Y) = \sum_{\delta(X,Y) \in \boldsymbol{F}_i} \mathcal{F}_{LIn_i}(\delta(X, Y)). \tag{7}$$

For **QR** 4, we also consider the worst case in which the leakage of information through each flow is different. Then, the overall quantity on leakage from X to Y is the sum of the leakage in each flow $\delta(X, Y)$. Based on the above quantifying rules and Definition 1, we can derive the following theorem on information flow security in s_i.

Theorem 1. *$\forall X \in HIn_i$ in s_i, the flows in s_i are K-secure if they satisfy that*

$$\sum_{Y \in LOut_i} \mathcal{F}_{LIn_i}(X \rightsquigarrow Y) < K, \quad 0 \leq K \leq k,$$

where K is the security threshold.

Proof. Based on the information entropy and the quantifying rules, it is easy to deduce that

$$\mathcal{F}_{LIn_i}(X \rightsquigarrow LOut_i) \leq \sum_{Y \in LOut_i} \mathcal{F}_{LIn_i}(X \rightsquigarrow Y) < K.$$

According to Definition 1, the flows in s_i are secure. □

3.3. Quantifying the Information Flow in the Service Chain

In our threat model, different service providers may collude together to analyze s user's private data. It means that different providers may share their knowledge on the low-level inputs and outputs during the service composition, which causes more leakage on a user's private data. In order to quantify the additional leakage of private data across different services, we design the quantifying rules based on the analysis of the inter-service flows.

For service chain $S_{ch} = \langle s_0, s_1, s_2, \ldots, s_n \rangle$ where $In_{ch} = \bigcup_{0 \leq i \leq n} In_i = In_{0,1,\ldots,n}$ and $Out_{ch} = \bigcup_{0 \leq i \leq n} Out_i = Out_{0,1,\ldots,n}$, the inter-service flows may occur between the inputs and outputs across multiple services, which is shown as Figures 3 and 4.

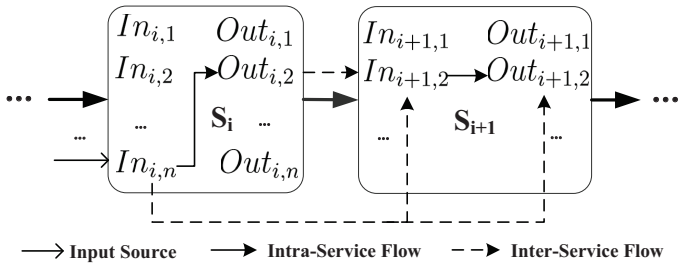

Figure 3. Information flow between adjacent services.

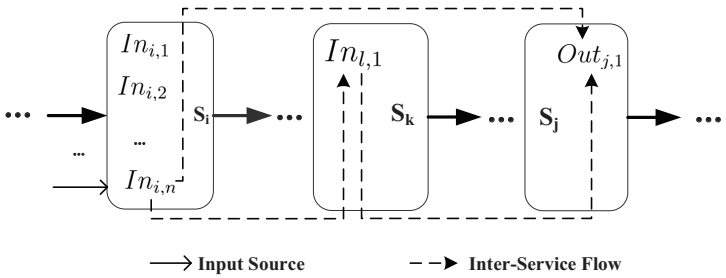

Figure 4. Information flow across multiple services.

Adjacent-service flow is the basic inter-service flow, which occurs because of the transmission on the intermediate result between the outputs and inputs across the adjacent services, such as the inter-service flow between $Out_{i,2}$ and $In_{i+1,2}$. Based on the adjacent-service flows, more inter-service flows occur due to the transitivity of the information flow, such as the inter-service flow between $In_{i,n}$ and $Out_{j,1}$, $0 \leq i < j$. Therefore, we can formally define the inter-service flows as follows [7]:

Definition 3. *$\forall X \in In_i$ and $\forall Y \in Out_j$ where $0 \leq i < j \leq n$, there are following two cases.*
(1) $j = i + 1$: $\exists W_1 \in Out_i$, $W_2 \in In_j$, $W_1 \to W_2$ that satisfy $X \rightsquigarrow W_1$ and $W_2 \rightsquigarrow Y$, then $X \rightsquigarrow W_2$ and $X \rightsquigarrow Y$.
(2) $j > i + 1$: $\exists W \in In_l \cup Out_l$, $i < l < j$ that satisfy $X \rightsquigarrow W$ and $W \rightsquigarrow Y$, then $X \rightsquigarrow Y$.

Based on Definition 3, we can obtain all the inter-service flows. Here, we define $\boldsymbol{F}_{ch} = \{X \rightsquigarrow Y | X \in In_i, Y \in Out_j, 0 \leq i < j \leq n\}$. According to the composition structure of service chain model, the intermediate result is the only method that passes the value of input source across multiple services. Then, we can obtain the following proposition.

Proposition 1. *$\forall X \in In_i$ and $\forall Y \in Out_j$, $0 \leq i < j \leq n$, if $X \rightsquigarrow Y$, $\exists W \in In_j$ satisfies that $X \rightsquigarrow W$ and $W \rightsquigarrow Y$.*

On the basis of Proposition 1, for each inter-service flow $X \rightsquigarrow Y$, its leakage from X to Y depends on the quantity of information that X passes to W and how much information is leaked through the intra-service flow $W \rightsquigarrow Y$. Then, we use $\mathcal{F}_{LIn_{i,i+1,\ldots,j}}(X \rightsquigarrow Y)_W$ to represent the additional leakage of X to Y through W, which can be calculated based on the following rule.

QR 5. *$\forall X \in HIn_i$, $Y \in LOut_j$ and $W \in In_j$ satisfy $X \rightsquigarrow W$ and $W \rightsquigarrow Y$, then we have*

$$\mathcal{F}_{LIn_{i,i+1,\ldots,j}}(X \rightsquigarrow Y)_W = \begin{cases} 0, & W \in LIn_j, \\ \mathcal{F}_{LIn_j}(W \rightsquigarrow Y), & W \in HIn_j. \end{cases} \tag{8}$$

For the inter-service flow $X \rightsquigarrow Y$ through W, there are two cases to consider in **QR** 5.

(1) $W \in LIn_j$: Because $W \in LIn_j$, the information of X is leaked through W. Then, service providers can not obtain additional information about X through the flow $W \rightsquigarrow Y$. For this type of flow, $\mathcal{F}_{LIn_{i,i+1,\ldots,j}}(X \rightsquigarrow Y)_W = 0$.

(2) $W \in HIn_j$: During the service composition, it is considered secure that private data are delivered between high-level sources and sinks. Thus, the explicit flows usually occur between the high-level inputs and outputs. In this case, we also consider the worst assumption that all the information of X is delivered to W based on **QR** 2. Because $W \in HIn_j$, the information of X can not be leaked through W. The leakage of X depends on how much information of W leaks through the flow $W \rightsquigarrow Y$. In addition, we assume that the leakage from X to Y is different from the previous flows. Then, we can obtain that $\mathcal{F}_{LIn_{i,i+1,\ldots,j}}(X \rightsquigarrow Y)_W = \mathcal{F}_{LIn_j}(W \rightsquigarrow Y)$.

In addition, for each inter-service flow $X \rightsquigarrow Y$, the information of X may leak to Y through different W. Then, we can deduce the following lemma.

Lemma 1. *$\forall X \in Hin_i$ and $Y \in LOut_j$, $0 \leq i < j \leq n$, satisfy $X \rightsquigarrow Y$, then*

$$\mathcal{F}_{LIn_{i,i+1,\ldots,j}}(X \rightsquigarrow Y) \leq \sum_{W \in HIn_j} \mathcal{F}_{LIn_j}(W \rightsquigarrow Y),$$

where W satisfies $X \rightsquigarrow W$ and $W \rightsquigarrow Y$.

Proof. According to the above analysis and the service chain model, we can deduce that

$$\mathcal{F}_{LIn_{i,i+1,\ldots,j}}(X \rightsquigarrow Y) \leq \sum_{W \in In_j} \mathcal{F}_{LIn_{i,i+1,\ldots,j}}(X \rightsquigarrow Y)_W = \sum_{W \in HIn_j} \mathcal{F}_{LIn_j}(W \rightsquigarrow Y)$$

lemma is proved. □

Based on Lemma 1, we can obtain that

Lemma 2. *In a service chain $S_{ch} = \{s_0, s_1, \ldots, s_n\}$, $\forall X \in Hin_i$ and $\forall Y \in LOut_{ch}$ satisfy $X \rightsquigarrow Y$, then*

$$\mathcal{F}_{LIn_{i,i+1,\ldots,n}}(X \rightsquigarrow LOut_{ch}) \leq \mathcal{L}(X)_i + \mathcal{L}(X)_{i+1,\ldots,n},$$

where $\mathcal{L}(X)_i$ is the leakage of X to $\forall Y \in LOut_i$ in service s_i, namely,

$$\mathcal{L}(X)_i = \sum_{Y \in LOut_i} \mathcal{F}_{LIn_i}(X \rightsquigarrow Y), \tag{9}$$

and $\mathcal{L}(X)_{i+1,\ldots,n}$ is the additional leakage of X to $\forall Y \in LOut_j, i < j \leq n$ in following services $s_{i+1}, \ldots, s_j$, namely,

$$\mathcal{L}(X)_{i+1,\ldots,n} = \sum_{j=i+1}^{n} \sum_{Y \in LOut_j} \sum_{W \in HIn_j} \mathcal{F}_{LIn_j}(W \rightsquigarrow Y), \tag{10}$$

and W satisfies $X \rightsquigarrow W$ and $W \rightsquigarrow Y$ in Equation (10).

Proof. The proof is shown in Appendix A. □

Based on Lemma 2, we can derive the following information flow security theorem.

Theorem 2. *For a service chain $s_{ch} = \{s_0, s_1, \ldots, s_n\}$, the information flows are K-secure if each service component $s_j, 0 \leq j \leq n$, satisfies the following two conditions:*
(1) Flows in each service component s_j are secure.
(2) $\forall X \in HIn_i$, $0 \leq i < j$; it satisfies that

$$\mathcal{L}(X)_{i,\ldots,j-1} + \mathcal{L}(X)_j \leq K, \tag{11}$$

where K is the security threshold. $\mathcal{L}(X)_{i,\ldots,j-1}$ *is the overall leakage of X from* s_i *to* s_{j-1}*, namely,*

$$\mathcal{L}(X)_{i,\ldots,j-1} = \mathcal{L}(X)_i + \mathcal{L}(X)_{i+1,\ldots,j-1} \tag{12}$$

and $\mathcal{L}(X)_j$ *is the leakage of X in* s_j*, namely,*

$$\mathcal{L}(X)_j = \sum_{Y \in LOut_j} \sum_{W \in HIn_j} \mathcal{F}_{LIn_j}(W \rightsquigarrow Y) \tag{13}$$

and W satisfies $X \rightsquigarrow W$ *and* $W \rightsquigarrow Y$ *in Equation (13).*

Theorem 2 can be proved based on Lemma 2 and Definition 1. The security constraints on each service are given in Theorems 1 and 2, which makes a basis for the decentralized evaluation in mobile computing environment. Each service requires that the leakage of high level data through the intra and inter flows can not exceed the threshold K.

4. Distributed Quantitative Information Flow Evaluation for Service Composition in a Mobile Computing Environment

In MSS, services may be composed together to accomplish a user's complex service requirement. For a service chain $S_{ch} = \{s_0, s_1, s_2, \ldots, s_n\}$, there are several candidate service components with the same functions but different providers for each service step s_i. In order to efficiently evaluate the leakage for the service composition in a mobile computing environment, we propose a distributed quantitative information flow evaluation approach based on Theorems 1 and 2.

By referring to Figure 1, candidate services and security authorities will be involved in the evaluation procedure. The procedure includes two phases, i.e., intra-service evaluation and inter-service evaluation. First, each candidate service is evaluated by its local SA, and SA generates a security certificate for the following inter-service evaluation. When these candidate services are going to be composed together, the inter-service evaluation process will be executed for the evaluation on leakage by inter-service flows.

4.1. Intra-Service Evaluation

The intra-service evaluation is executed by SA before the service composition. SA evaluates each candidate service s_i based on the quantifying rules and Theorem 1, and generates security certificates Ce_i for secure ones. This phase can be executed in an offline way to reduce the evaluation cost during the composition.

During the intra-service evaluation, SA first obtains the PDG of s_i, then computes the quantity of leakage from $\forall X \in HIn_i$ to $\forall Y \in LOut_i$ based on the above QRs. After that, SA validates the flow in s_i. For secure services, a certificate Ce_i specifying the quantity of leakage on each high-level inputs $\mathcal{L}(X)_i$ is generated for the following evaluation. Insecure ones without certificates are not allowed to be composed during the service composition. The intra-service evaluation procedure is presented as Algorithm 1.

In the computation of the leakage on $X \rightsquigarrow Y$, we record the value in the certificate which can be used in the inter-service evaluation phase. It can avoid the repeated work on quantifying leakage in a same service component. At the end of the procedure, we record the flows between high level inputs and outputs in certificate instead of computing its leakage. It is based on our worst assumption that information has been passed to high level outputs if there is a flow, which usually happens. In the meantime, it can save lots of efforts on computation of leakage during inter-service evaluation without loss on security.

For a clear description on our intra-service evaluation algorithm, consider the following service's program:

```
public static int Compare(int hin, int lin){
      int hout, lout;
      hout=hin;
      lout=-1;
      if(hin>lin)
         lout=0;
      else
         lout=1;
}
```

In the above example, *hin* and *hout* are high-level inputs and outputs while *lin* and *lout* are low-level ones. First, the code needs to be sent to SA. Then, SA constructs the PDG of 'Compare' service and obtains the intra-service flow set $\boldsymbol{F}_i = \{hin \rightarrow hout, hin \overset{hin>lin}{\dashrightarrow} lout, lin \overset{hin>lin}{\dashrightarrow} lout\}$. After that, we compute the leakage from *hin* to *lout* through the flow $hin \overset{hin>lin}{\dashrightarrow} lout$ based on **QR** 3. The leakage is validated according to security threshold K. If it is considered secure, the leakage of *hin* through each flow, current overall leakage of *hin* and the intra-service flows between *hin* and *hout* will be recorded in certificate Ce_i for the inter-service evaluation. Finally, certificate Ce_i is signed by SA for the protection against manipulation.

Algorithm 1 *Intra_Eval*()

Input: s_i, K
Output: **True** or **False**, Ce_i.

1: generate the s_i's PDG and obtain $\boldsymbol{F}_i$
2: **for** each $X \in HIn_i$ **do**
3: **for** each $Y \in LOut_i$ **do**
4: **for** each $\delta(X, Y) \in \boldsymbol{F}_i$ **do**
5: compute $\mathcal{F}_{LIn_i}(\delta(X, Y))$ based on **QR** 2 and **QR** 3
6: $\mathcal{F}_{LIn_i}(X \rightsquigarrow Y) = \mathcal{F}_{LIn_i}(X \rightsquigarrow Y) + \mathcal{F}_{LIn_i}(\delta(X, Y))$
7: **end for**
8: record the $\mathcal{F}_{LIn_i}(X \rightsquigarrow Y)$ into service certificate Ce_i
9: $\mathcal{L}(X)_i = \mathcal{L}(X)_i + \mathcal{F}_{LIn_i}(X \rightsquigarrow Y)$
10: **end for**
11: **if** $\mathcal{L}(X)_i \geq K$ **then**
12: **return** **False**
13: **end if**
14: record the $\mathcal{L}(X)_i$ into service certificate Ce_i
15: **for** each $Y \in HOut_i$ **do**
16: **if** $\exists \delta(X, Y) \in \boldsymbol{F}_i$ **then**
17: record the flow from X to Y into service certificate Ce_i
18: **end if**
19: **end for**
20: **end for**
21: signature(Ce_i,SA)
22: **return** **True**

4.2. Inter-Service Evaluation

Inter-Service evaluation is a vital phase to evaluate the leakage of high level data during service composition. In this phase, s_i firstly retrieves current leakage on high level data $L(X)_{0,1,\ldots,i}$ and their inter flows $\boldsymbol{F}_{0,1,\ldots,i}$. Then, s_i requires s_{i+1}'s intra flow and leakage through the certificate Ce_{i+1}, and it updates the inter-service flow set and evaluates the candidate service s_{i+1} according to Theorem 2. The inter-Service evaluation procedure is shown as Algorithm 2.

During the evaluation on candidate service s_{i+1}, the additional leakage $L(X)_{i+1}$ is first calculated based on **QR** 5 and Lemma 1. After that, the overall leakage of each high-level input, $L(X)_{0,1,\ldots,i+1}$, is computed and validated. If the overall leakage on any high-level input exceeds security threshold K, it means that this candidate service s_{i+1} is not secure for composition.

Algorithm 2 *Inter_Eval*()

Input: s_{i+1}, K, $\mathcal{L}(X)_{0,1,\ldots,i}$, $\boldsymbol{F}_{0,1,\ldots,i}$
Output: **True** or **False**, $\mathcal{L}(X)0,1,\ldots,i+1$, $\boldsymbol{F}_{0,1,\ldots,i+1}$.

retrieve cert Ce_{i+1}
update the flows $\boldsymbol{F}_{0,1,\ldots,i+1}$ based $\boldsymbol{F}_{0,1,\ldots,i}$ and Ce_{i+1}
for each $X \in HIn_j, j < i+1$ **do**
  for each $Y \in LOut_{i+1}$ **do**
    for each $W \in HIn_{i+1}$ **do**
      if $(X \rightsquigarrow W) \wedge (W \rightsquigarrow Y) \in \boldsymbol{F}_{0,1,\ldots,i+1}$ **then**
        get the leakage $\mathcal{F}_{LIn_{i+1}}(W \rightsquigarrow Y)$ from the certificate Ce_{i+1}
        $\mathcal{L}(X)_{i+1}$=$\mathcal{L}(X)_{i+1}$+ $\mathcal{F}_{LIn_{i+1}}(W \rightsquigarrow Y)$
      end if
    end for
  end for
  $\mathcal{L}(X)_{0,1,\ldots,i+1}$ =$\mathcal{L}(X)_{0,1,\ldots,i}$+$\mathcal{L}(X)_{i+1}$
  if $\mathcal{L}(X)_{0,1,\ldots,i+1} \geq K$ **then**
    return **False**
  end if
end for
return **True**

4.3. Distributed Quantitative Information Flow Evaluation Algorithm for the Service Composition in Mobile Computing Environments

Based on the intra-service and inter-service evaluation procedure, we propose a distributed quantitative information flow evaluation algorithm for service composition across multiple mobile devices. The evaluation algorithm on each high level input is presented as Algorithm 3.

Algorithm 3 *Eval_SC*()

Input: s_i, s_{i+1} K
Output: $\mathcal{L}(X)0,1,\ldots,n$

wait **start_message**
if $X \in HIn_i$ **then**
  \\ Initiate the X's leakage in its first service
  $\mathcal{L}(X)_{i,i+1,\ldots,n} = \mathcal{L}(X)_i$
  $\boldsymbol{F}_{i,i+1,\ldots,n} = \boldsymbol{F}_i$
  send **start_message** to $s_i + 1$'s SA
else
  get $\mathcal{L}(X)_{0,\ldots,i}$ and $\boldsymbol{F}_{0,\ldots,i}$ from **start_message**
  if *Inter_Eval*(s_{i+1},K,$\mathcal{L}(X)_{0,\ldots,i}$, $\boldsymbol{F}_{0,\ldots,i}$)=**Fail then**
    send **fail_message** to the user
  else
    if i=n **then**
      send **success_message** to the user
    else
      send **start_message** to s_{i+1}
    end if
  end if
end if

The algorithm is deployed on each service node in a mobile computing environment. Then, it works in a step-by-step way through the cooperation among multiple services in different mobile devices. For each possible service chain, a user sends a start message to the first service s_0 to start the evaluation procedure. During each step evaluation, each candidate service s_{i+1} is evaluated by its predecessor s_i. If it returns true, s_i will send a start message with current leakage and flows to its successors to continue the evaluation procedure. Otherwise, s_i will send a failure message to a user to check whether this service chain is not secure, and the evaluation on this chain will stop. In addition, for each high-level input in the service chain, it needs to be initiated in its first service based on the certificate. When the final service s_n passes the evaluation, then it will send a success message with the overall leakage $\mathcal{L}(X)_{0,1,\ldots,n}$ on each high level input to user. The leakage can be used as a security criterion on different candidate service chains.

5. Experiments and Evaluations

The information flow security can be ensured by Theorem 2, and the security proof and analysis are shown in Appendix A. The basic comparisons of related approaches are shown in Table 1.

Table 1. Basic comparison.

	Approach	**Mode**	**Service Composition**
Our Approach	Quantitative	Distributed	$\surd$
She et al. [13,14]	Qualitative	Centralized	$\surd$
Xi et al. [25]	Qualitative	Distributed	$\surd$
Clark et al. [20,24]	Quantitative	Centralized	$\times$
Smith et al. [22]	Quantitative	Centralized	$\times$

According to Table 1, traditional approaches validate the information flow across multiple services based on non-interference, i.e., qualitative verification. Comparing with quantitative approaches for a program, our approach supports the distributed quantifying on the flows across multiple services, which is more appropriate for the service composition in a mobile computing environment. Although we refer to the rules in [20], these rules are used to quantify the overall leakage of high-level inputs instead of each flow's leakage, which is more suitable for the centralized evaluation on a single program.

We also implemented our approach in Huawei mobile phones and ran the evaluation procedure in a WLAN network with the speed of 150 Mbps. The basic configuration is shown in Table 2.

Table 2. Configuration.

Mobile Environment	
Network Type	WLAN
Network Speed	150 Mbps
Mobile Mode	random walk
Mobile Devices	Huawei nova 3
Device's CPU and RAM	2.8 GHz, 6 G
Mobile Device's Operation System	Android 9.0
Data Set	
Service Step	1–10
Candidate Number	1–10
Security Level	H, L
High Level Input and Output	2, 2
Low Level Input and Output	2, 2
Flows between Input and Output	randomly generated

In order to evaluate the performance of our approach, we construct a data set of android applications. These applications support two security levels, i.e., H (High) and L (Low). Each application has two high-level inputs, two low-level inputs, two high-level outputs and two low-level outputs. The flows between different inputs and outputs in each application are randomly generated. These applications can be regarded as basic services in mobile computing environments, which can be composed together as a composite service by network communication. For the composite service, the number of service step N_s is from 1 to 10. The number of candidate service N_c for each step is also from 1 to 10. N_s means this composite service is composed by N_s types of applications. N_c means there are N_c applications having similar functions but different implementations for each type service. In our experiments, we focus on the evaluations on service availability, time cost and energy cost.

(1) Service Availability: we use success number and success rate to evaluate the availability of the composite service. Success number N_{suc} is the number of the composite services that successfully pass the validation. In addition, success rate R_{suc} is the percentage of successful composite services in all possible ones, which can be calculated as the following equation:

$$R_{suc} = \frac{N_{suc}}{N_{all}}, \tag{14}$$

where N_{all} is the number of all possible composite services composed of different candidate applications in our data set.

In this test, we execute the quantitative approaches with different security threshold (K = 8, 16, 32) and qualitative approaches 100 times separately. Figure 5 shows the average success number and success rate on service composition with different approaches.

Figure 5a shows the variation on average success number of service composition with fixed service steps ($N_s = 5$) but different number of candidate services N_c. With the increase in N_c, the number is rising because it is easier to be successfully composed with more candidate services. Figure 5b shows the variation on average success rate of service composition with fixed candidate services ($N_c = 5$) but a different number of service steps N_s. With the increase in N_s, the rate is declining because it is harder to find an appropriate service that can satisfy the security constraints.

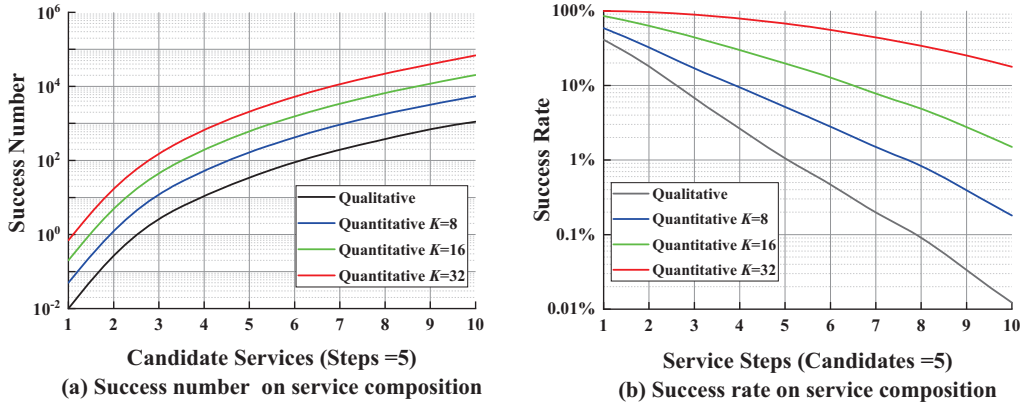

Figure 5. Success number and success rate on service composition.

Throughout both figures, the quantitative approaches have better performance compared to the qualitative approach. Especially for the success rate when $N_s = 10$ in Figure 5b, few services could pass the qualitative validation which may cause failure on service composition. On the contrary, the success rate is still high in quantitative approaches. It indicates that the performance of the quantitative approach is apparently superior to that of the qualitative approach. Moreover, it is easier to be successful with a higher security threshold.

(2) Time Cost: we focus on the time cost on different types of information flow validation approaches. The time cost mainly includes two types of costs, i.e., computation and communication. Figure 6a shows the overall time cost including computation and communication on qualitative and quantitative approaches. With the increase in the number of candidate services, the time cost is rising because of the increase in the complexity and the number of possible service chains. It also costs more time due to the additional computations and communications in quantitative evaluation approaches. However, we can minimize the cost by precomputation on the quantity of leakage in each candidate service.

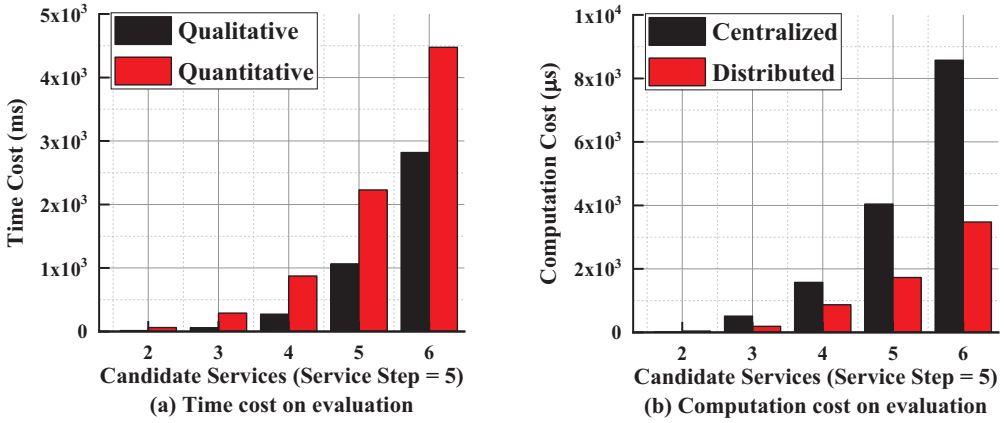

Figure 6. Time cost on information flow evaluation.

Figure 6b shows the average computation cost on each mobile device involved in the distributed and centralized quantitative approaches. We use the time cost on computation to represent the computation cost in this test. Instead of executing all of the evaluation work on a single device, the distributed way coordinates all participants to accomplish the evaluation together, which provides a better balance on the computation cost on mobile devices.

(3) Energy Cost on User's device: During the evaluation, the energy cost is mainly caused by the computation and the communication cost on a user's device. In our distributed approach, the user starts the evaluation with little computation cost. The computation cost is also evaluated in the above 'time cost' test. Thus, we focus on the communication cost on the user's mobile device in this test.

Figure 7 shows the communication cost on the user's mobile device in the distributed and centralized evaluation approaches. For the centralized evaluation approach, all candidate services will be evaluated by user's device. For our distributed approach, a user's device only needs to send the request to its following services and receives the final results from the last-step services. Therefore, the communication cost in a centralized approach is higher than that in our distributed approach. By combining the evaluation on the computation cost, they indicate that the energy cost on a user's device can be reduced by our distributed approach.

Based on the above experiments, the results show that our approach can provide better service availability with a small increase in time cost, provide a better load balance on computation and reduce the overload on the users' mobile phones effectively.

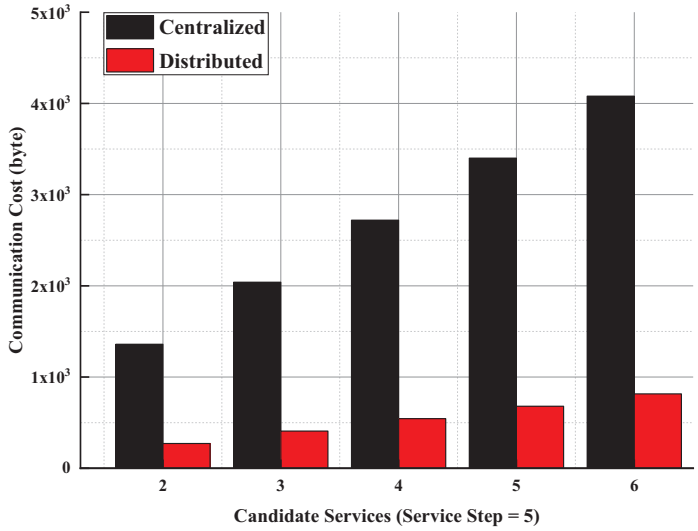

Figure 7. Communication cost on information flow evaluation.

6. Conclusions

Strict qualitative disciplines decrease the availability of the composite service and may cause a high failure rate on service composition. In this paper, we propose a distributed quantitative information flow evaluation approach for secure service composition in mobile computing environments. Our approach first evaluates the intra-service leakage between different inputs and outputs in each service, and then ensures the inter-service flow security based on the constraints specified in Theorem 2. Our framework and approach works in a distributed way which is quite suitable for the evaluation executed by energy-limited devices in mobile computing environments. Through experiments and evaluations, the results show that our approach can improve the service availability effectively and provide a better load balance on each device.

Author Contributions: Conceptualization, N.X. and J.M.; methodology, N.X., C.S. and J.M.; software, J.L. and N.X.; validation, J.L. and N.X.; formal analysis, N.X.; writing—original draft preparation, N.X.; writing—review and editing, C.S.; supervision, J.M.

Funding: This research is funded by the Natural Science Basis Research Plan in Shaanxi Province of China (Grant No. 2016JM6034) and the National Natural Science Foundation of China (61502368 and U1405255).

Acknowledgments: The authors would like to thank the administrative and technical support provided by Yang Xiang, Jun Zhang and Chao Chen from Swinburne University.

Conflicts of Interest: The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Appendix A. Proof of Lemma 2

In this section, we are going to prove Lemma 2 by using mathematical induction.

Proof. First, let $n = 1$, then $S_{ch} = \{s_0, s_1\}$ where $In_{ch} = \{In_0, In_1\}$ and $Out_{ch} = \{Out_0, Out_1\}$. In addition, we consider the following two cases:
(1) $\forall X \in HIn_0$, based on the service chain model, $LOut_{ch} = LOut_0 \bigcup LOut_1$. Then, we can get that

$$\mathcal{F}_{LIn_{0,1}}(X \rightsquigarrow LOut_{ch}) \leq \mathcal{F}_{LIn_0}(X \rightsquigarrow LOut_0) + \mathcal{F}_{LIn_{0,1}}(X \rightsquigarrow LOut_1). \tag{A1}$$

For $F_{LIn_0}(X \rightsquigarrow LOut_0)$, we can infer that

$$\mathcal{F}_{LIn_0}(X \rightsquigarrow LOut_0) \leq \sum_{Y \in LOut_0} \mathcal{F}_{LIn_0}(X \rightsquigarrow Y). \tag{A2}$$

For $\mathcal{F}_{LIn_{0,1}}(X \rightsquigarrow LOut_1)$,

$$\mathcal{F}_{LIn_{0,1}}(X \rightsquigarrow LOut_1) \leq \sum_{Y' \in LOut_1} \mathcal{F}_{LIn_{0,1}}(X \rightsquigarrow Y'). \tag{A3}$$

Based on Lemma 1, we can obtain that

$$\mathcal{F}_{LIn_{0,1}}(X \rightsquigarrow Y') \leq \sum_{W \in HIn_1} \mathcal{F}_{LIn_1}(W \rightsquigarrow Y'), \tag{A4}$$

where W satisfies $X \rightsquigarrow W$ and $W \rightsquigarrow Y$.

According to Equations (A1), (A2) and (A4),

$$\mathcal{F}_{LIn_{0,1}}(X \rightsquigarrow LOut_{ch}) \leq \sum_{Y \in LOut_0} \mathcal{F}_{LIn_0}(X \rightsquigarrow Y) + \sum_{Y' \in LOut_1} \sum_{W \in HIn_1} \mathcal{F}_{LIn_1}(W \rightsquigarrow Y').$$

(2) $\forall X \in HIn_1$, there is no inter flow because s_1 is the last service. Then,

$$\mathcal{F}_{LIn_1}(X \rightsquigarrow LOut_1) \leq \sum_{Y \in LOut_1} \mathcal{F}_{LIn_1}(X \rightsquigarrow Y).$$

Therefore, when $n = 1$, lemma is proved.

Then, we suppose that the Lemma is true when $n = m$. Then, the case that $n = m + 1$ is proved as follows:

(1) $\forall X \in HIn_i$, $0 \leq i \leq m$, we can obtain that $LOut_{ch} = LOut_{i,i+1,\ldots,m} \bigcup LOut_{m+1}$. Then,

$$\mathcal{F}_{LIn_{i,..m+1}}(X \rightsquigarrow LOut_{ch}) \leq \mathcal{F}_{LIn_{i,..m}}(X \rightsquigarrow LOut_{i,i+1,\ldots,m}) + \mathcal{F}_{LIn_{i,..m+1}}(X \rightsquigarrow LOut_{m+1}) \tag{A5}$$

Based on our assumption on $n = m$, we can get that

$$\mathcal{F}_{LIn_{i,\ldots,m}}(X \rightsquigarrow LOut_{i,i+1,\ldots,m}) \leq \sum_{Y \in LOut_i} \mathcal{F}_{LIn_i}(X \rightsquigarrow Y) + \sum_{j=i+1}^{m} \sum_{Y' \in LOut_j} \sum_{W \in HIn_j} \mathcal{F}_{LIn_j}(W \rightsquigarrow Y'), \tag{A6}$$

where W satisfies $X \rightsquigarrow W$ and $W \rightsquigarrow Y'$.

Based on Lemma 1, we can deduce the following equation:

$$\mathcal{F}_{LIn_{i,\ldots,m+1}}(X \rightsquigarrow LOut_{m+1}) \leq \sum_{Y' \in LOut_{m+1}} \sum_{W \in HIn_{m+1}} \mathcal{F}_{LIn_{m+1}}(W \rightsquigarrow Y'), \tag{A7}$$

where W satisfies $X \rightsquigarrow W$ and $W \rightsquigarrow Y'$.

Based on Equations (A6), (A5) and (A7),

$$\mathcal{F}_{LIn_{i,\ldots,m+1}}(X \rightsquigarrow LOut_{ch}) \leq \sum_{Y \in LOut_i} \mathcal{F}_{LIn_i}(X \rightsquigarrow Y) + \sum_{j=i+1}^{m+1} \sum_{Y' \in LOut_j} \sum_{W \in HIn_j} \mathcal{F}_{LIn_j}(W \rightsquigarrow Y'),$$

(2) $\forall X \in HIn_{m+1}$, we can prove that a lemma is true similar to case (2) in $n = 1$.

Therefore, when $n = m + 1$, Lemma 2 is proved. □

References

1. Agiwal, M.; Roy, A.; Saxena, N. Next, Generation 5G Wireless Networks: A Comprehensive Survey. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 1617–1655, doi:10.1109/COMST.2016.2532458.
2. Beshley, H.; Kyryk, M.; Beshley, M.; Panchenko, O. Method of Information Flows Engineering and Resource Distribution in 4G/5G Heterogeneous Network for M2M Service Provisioning. In Proceedings of the 2018 IEEE 4th International Symposium on Wireless Systems within the International Conferences on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS-SWS), Lviv, Ukraine, 20–21 September 2018; pp. 229–233, doi:10.1109/IDAACS-SWS.2018.8525680.
3. Ngoc, N.C.H.; Lin, D.; Nakaguchi, T.; Ishida, T. QoS-Aware Service Composition in Mobile Environments. In Proceedings of the 2014 IEEE 7th International Conference on Service-Oriented Computing and Applications, Matsue, Japan, 17–19 November 2014; pp. 97–104, doi:10.1109/SOCA.2014.51.
4. Ridhawi, Y.A.; Karmouch, A. Decentralized Plan-Free Semantic-Based Service Composition in Mobile Networks. *IEEE Trans. Serv. Comput.* **2015**, *8*, 17–31, doi:10.1109/TSC.2013.2297114.
5. Palade, A.; Clarke, S. Stigmergy-Based QoS Optimisation for Flexible Service Composition in Mobile Communities. In Proceedings of the 2018 IEEE World Congress on Services (SERVICES), San Francisco, CA, USA, 2–7 July 2018; pp. 27–28, doi:10.1109/SERVICES.2018.00027.
6. Deng, S.; Huang, L.; Taheri, J.; Yin, J.; Zhou, M.; Zomaya, A.Y. Mobility-Aware Service Composition in Mobile Communities. *IEEE Trans. Syst. Man Cybern. Syst.* **2017**, *47*, 555–568, doi:10.1109/TSMC.2016.2521736.
7. Xi, N.; Ma, J.; Sun, C.; Zhang, T. Decentralized Information Flow Verification Framework for the Service Chain Composition in Mobile Computing Environments. In Proceedings of the 2013 IEEE 20th International Conference on Web Services, Santa Clara, CA, USA, 27 June–2 July 2013; pp. 563–570, doi:10.1109/ICWS.2013.81.
8. Bertino, E.; Squicciarini, A.C.; Mevi, D. A fine-grained access control model for web services. In Proceedings of the IEEE International Conference on Services Computing (SCC 2004), Shanghai, China, 15–18 September 2004; pp. 33–40.
9. Bhatti, R.; Bertino, E.; Ghafoor, A. A trust-based context-aware access control model for web-services. *Distrib. Parallel Databases* **2005**, *18*, 83–105.
10. Hutter, D.; Volkamer, M. *Information Flow Control to Secure Dynamic Web Service Composition*; SPC; Springer: Berlin/Heidelberg, Germany, 2006; Volume 3934, pp. 196–210.
11. Nakajima, S. *Model-Checking of Safety and Security Aspects in Web Service Flows*; ICWE; Springer: Munich, Germany, 2004; Volume 3140, pp. 488–501.
12. Rossi, S. *Model Checking Adaptive Multilevel Service Compositions*; FACS; Springer: Guimaraes, Portugal, 2010; pp. 106–124.
13. She, W.; Yen, I.L.; Thuraisingham, B.; Huang, S.Y. Rule-Based Run-Time Information Flow Control in Service Cloud. In Proceedings of the 2011 IEEE International Conference on Web Services, Washington, DC, USA, 4–9 July 2011; pp. 524–531, doi:10.1109/ICWS.2011.35.
14. She, W.; Yen, I.L.; Thuraisingham, B.; Bertino, E. Security-aware service composition with fine-grained information flow control. *Serv. Comput. IEEE Trans.* **2013**, *6*, 330–343.
15. Schwartz, E.J.; Avgerinos, T.; Brumley, D. All You Ever Wanted to Know about Dynamic Taint Analysis and Forward Symbolic Execution (but Might Have Been Afraid to Ask). In Proceedings of the 2010 IEEE Symposium on Security and Privacy, Oakland, CA, USA, 16–19 May 2010; pp. 317–331, doi:10.1109/SP.2010.26.
16. Schuette, J.; Brost, G.S. LUCON: Data Flow Control for Message-Based IoT Systems. In Proceedings of the 2018 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/12th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE), New York, NY, USA, 1–3 August 2018; pp. 289–299, doi:10.1109/TrustCom/BigDataSE.2018.00052.
17. Denning, D.E. A Lattice Model of Secure Information Flow. *Commun. ACM* **1976**, *19*, 236–243, doi:10.1145/360051.360056.
18. Smith, G. Recent Developments in Quantitative Information Flow (Invited Tutorial). In Proceedings of the 2015 30th Annual ACM/IEEE Symposium on Logic in Computer Science, Kyoto, Japan, 6–10 July 2015; pp. 23–31, doi:10.1109/LICS.2015.13.

19. Clark, D.; Hunt, S.; Malacaria, P. Quantitative Information Flow, Relations and Polymorphic Types. *J. Log. Comput.* **2005**, *15*, 181–199, doi:10.1093/logcom/exi009.
20. Clark, D.; Hunt, S.; Malacaria, P. A static analysis for quantifying information flow in a simple imperative language. *J. Comput. Secur.* **2007**, *15*, 321–371, doi:10.3233/JCS-2007-15302.
21. Backes, M.; Kopf, B.; Rybalchenko, A. Automatic Discovery and Quantification of Information Leaks. In Proceedings of the 2009 30th IEEE Symposium on Security and Privacy, Berkeley, CA, USA, 17–20 May 2009; IEEE Computer Society: Washington, DC, USA, 2009; pp. 141–153, doi:10.1109/SP.2009.18.
22. Smith, D.M.; Smith, G. Tight Bounds on Information Leakage from Repeated Independent Runs. In Proceedings of the 2017 IEEE 30th Computer Security Foundations Symposium (CSF), Santa Barbara, CA, USA, 21–25 August 2017; pp. 318–327, doi:10.1109/CSF.2017.18.
23. Zhou, B.; Shi, Q.; Yang, P. A Survey on Quantitative Evaluation of Web Service Security. In Proceedings of the 2016 IEEE Trustcom/BigDataSE/ISPA, Tianjin, China, 23–26 August 2016; pp. 715–721, doi:10.1109/TrustCom.2016.0130.
24. Clark, D.; Hunt, S.; Malacaria, P. Quantitative Analysis of the Leakage of Confidential Data. *Electron. Notes Theor. Comput. Sci.* **2002**, *59*, 238–251, doi:10.1016/S1571-0661(04)00290-7.
25. Xi, N.; Sun, C.; Ma, J.; Shen, Y. Secure service composition with information flow control in service clouds. *Future Gener. Comput. Syst.* **2015**, *49*, 142–148, doi:10.1016/j.future.2014.12.009.
26. Ferrante, J.; Ottenstein, K.J.; Warren, J.D. The Program Dependence Graph and Its Use in Optimization. *ACM Trans. Program. Lang. Syst.* **1987**, *9*, 319–349, doi:10.1145/24039.24041.
27. Snelting, G.; Robschink, T.; Krinke, J. Efficient Path Conditions in Dependence Graphs for Software Safety Analysis. *ACM Trans. Softw. Eng. Methodol.* **2006**, *15*, 410–457, doi:10.1145/1178625.1178628.

© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).