Adaptive Secure Access to Remote Services in Mobile Environments

Hanping Lufei, Weisong Shi, *Member*, *IEEE*, and Vipin Chaudhary, *Member*, *IEEE*

Abstract—Since the inception of the service-oriented computing paradigm, we have witnessed a plethora of services deployed across a broad spectrum of applications, ranging from conventional RPC-based services to SOAP-based Web services. Likewise, the proliferation of mobile devices has enabled the remote "on the move" access of these services from anywhere at any time. Secure access to these services is challenging, especially in a mobile computing environment with heterogeneous modalities. Conventional static access control mechanisms are not able to accommodate complex secure access requirements. In this paper, we propose an *adaptive secure access* mechanism to address this problem. Our mechanism consists of two components: *an adaptive access control module* and *an adaptive function invocation module*. It not only adapts access control policies to diverse requirements but also introduces function invocation adaptation during access, which is the missing part of existing access control models. We have successfully applied the proposed adaptive secure access mechanism to a computer-assisted surgery application called UbiCAS. Performance evaluation shows that with limited overhead, our technique enforces secure access to the services provided by the UbiCAS system in a flexible way.

Index Terms—Distributed applications, mobile environments, access control in services systems.

1 Introduction

Service-oriented computing [5], [6], [35] is one of the main approaches to build distributed applications on the Web. At the same time, the Internet has been evolving into a more heterogeneous environment with a variety of devices (i.e., end hosts) connected by different edge networks. With the growth of heterogeneity in mobile computing environments, secure access to services is becoming more challenging in the design of these applications. We abstract the requirements of secure access to remote services as follows:

1. **Adaptation.** In a heterogeneous environment, like the Internet, it is very difficult, if not impossible, to build a one-size-fits-all approach that accommodates all diverse requirements. Adaptation has been considered as a general approach to address the mismatch problem between clients and servers [24], [34]. For secure access, there are two diverse requirements. On the user side, different configurations such as diverse devices and network bandwidths are coexisting. On the service side, there are different data formats, security requirements, and so on. Hence, we have to adapt access control policies to such diverse requirements.
2. **Efficiency.** For some applications, secure access enforcement incurs negligible system overhead. However, a poorly designed technique will not scale well and perform poorly with increasing system size. Hence, we need to meticulously design the enforcement algorithm that introduces minimum performance overhead.
3. **Evolvability.** System requirements usually keep evolving over time. A good secure access control framework should have the capability to extend current access policies. A user-friendly interface should also be defined for administrators who may need to add/update access control policies frequently to meet the evolved system requirements.

In this paper, we propose an adaptive secure access mechanism for remote accessible services that can be accessed from heterogeneous settings including mobile devices. Note that the services described in this paper is more general than conventional mobile services that are specifically designed for mobile devices like Smartphones and PocketPCs. For instance, Google Map is a typical remote service that can be accessed by both powerful destktops/laptops and Smartphones/PocketPCs, which are less powerful and have slower bandwidth.

Our approach contains two main modules: *an adaptive secure access control module* and *an adaptive secure function invocation module*. In the adaptive access control module, several definitions such as those of context term (CT) and context instance (CI) are formalized. An access control model is proposed to take application-related contexts into consideration in the design of access control policies. The enforcement algorithm is proven to be more efficient than the conventional access control enforcement algorithm in terms of time complexity.

We envision that secure access for remote services is a comprehensive process for access control and service invocation procedure, because the remote services we talk about here is more general and heterogeneous than

- *H. Lufei and W. Shi are with the Department of Computer Science, Wayne State University, 420 State Hall, 5143 Cass Ave., Detroit, MI 48084. E-mail: harrylufei@gmail.com, weisong@cs.wayne.edu.*
- *V. Chaudhary is with the Department of Computer Science and Engineering, University at Buffalo, State University of New York (SUNY), 201 Bell Hall, Buffalo, NY 14260. E-mail: vipin@cse.buffalo.edu.*

Manuscript received 27 Feb. 2008; revised 21 Oct. 2008; accepted 24 Oct. 2008; published online 27 Oct. 2008.
For information on obtaining reprints of this article, please send e-mail to: tsc@computer.org, and reference IEEECS Log Number TSC-2008-05-0051.
Digital Object Identifier no. 10.1109/TSC.2008.4.

conventional mobile services. For instance, even if a user is granted the access right to a service, he or she will not be able to access the service if the service's required encryption mechanism is not currently available on the user side. To remedy this situation, we introduce an adaptive function invocation module. We have applied the proposed adaptive secure access mechanism to a real-world computer-assisted surgery (CAS) application called UbiCAS [26] and evaluated the performance in a real mobile computing scenario. Compared with existing solutions, this paper has the following three contributions:

1. *Adaptive access control.* We propose a general model to integrate application-oriented contexts into the design. The computing time complexity of the access control enforcement algorithm is less than the traditional enforcement mechanism. Furthermore, the general design of the model enables it to dynamically evolve the access control policies to meet the future extended security requirements.
2. *Adaptive function invocation.* Besides adaptive access control, an adaptive function invocation module is also proposed. Users' diverse contexts sometimes mismatch the requirement of functions, and this in turn degrades the performance of functions or fails to execute the function at all. The adaptive function invocation module leverages appropriate components very efficiently.
3. *Implementation of secure access for the UbiCAS system.* We have successfully implemented our models in a distributed CAS system called UbiCAS. The integrated application enables different role members like doctors, surgeons, radiologists, and so on, using diverse devices, which are connected by different networks, to access multiple functions at any time from anywhere. The performance evaluation on different configurations shows that our approach provides efficient secure access to remote services with an acceptable extra overhead.

The rest of the paper is organized as follows: The system structure is introduced in Section 2. Then, we present the adaptive access control module and adaptive function invocation module in Sections 3 and 4, respectively. Section 5 depicts the details of implementation and evaluation. Finally, related work and concluding remarks are listed in Sections 6 and 7, respectively.

2 System Structure

With the proliferation of distributed applications, remote services are provided in different ways, such as Remote Procedure Call (RPC) [39] and Remote Method Invocation (RMI) [38] in Java and SOAP [44]. Although they have different names, the fundamental mechanism is similar. Usually, the server defines some functions including the implementation, the parameters, and the interfaces. If a user wants to use the remote service, he or she calls one of the functions by following the function interface. After the server finishes the task of the function, the result will be sent back to the user. In the following context, we use function and service interchangeably.

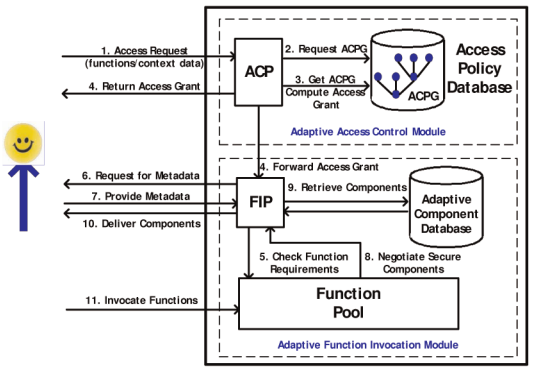


Fig. 1. An overview of the adaptive secure access mechanism.

Given this context, this section presents the system structure of the proposed adaptive secure access mechanism, which consists of two major components, *an adaptive access control module*, and *an adaptive function invocation module*, as shown in Fig. 1. The secure access procedure has two stages. First, the adaptive access control module decides which functions are allowed for clients to access. Second, the adaptive function invocation module deals with the adaptation of the function invocation.

2.1 Adaptive Access Control Module

The adaptive access control module enforces the access control policy. It constrains what a user can do, as well as what programs executing on behalf of the users are allowed to do. In this way, the access control module seeks to prevent the activity that could lead to a breach of security.

The adaptive access control module has two parts, *the access control point (ACP)* and *access policy database*, as shown in Fig. 1. When a client wants to access remote functions, first, he or she needs to provide his or her context information such as role, time, and location to ACP to acquire the access right to the desired functions. With the support of the access control policy database, ACP is able to do the access control enforcement using some algorithms, which will be discussed in Section 3. Access control policies are stored in the access policy database in the form of an ordered two-dimensional directed access control policy graph (ACPG), which will be explained also in Section 3.

2.2 Adaptive Function Invocation Module

After a user request passes the adaptive access control module, the adaptive function invocation module starts the procedure to adapt the invocation according to the user's dynamic contextual information. The motivation is that the same function call could experience totally different performances under different scenarios. As an example, calling an image download function could take intolerably long through a low-bandwidth network like dial-up while being fast via a T1 connection. Therefore, the priority of the adaptive function invocation module is to select appropriate adaptive components to augment the function call in different scenarios. Furthermore, if the selected components require user-side deployment, these components will be delivered to the user side and be plugged into the running space of the user-side program.

The adaptive function invocation module includes a *function invocation point (FIP)*, a *function pool*, and an *adaptive*

component database, as shown in the lower part of Fig. 1. FIP receives user context information and acquires the adaptive function invocation graph (AFIG), which will be described in Section 4, from the function pool. Then, FIP interacts with the user to decide the mandatory components for the function invocation and delivers the components from the database to the user if necessary.

2.3 Secure Access Procedure

Fig. 1 also shows the whole adaptive secure access procedure. After ACP receives the access request from a user in step 1, it will request and receive the ACPG from the access policy database (steps 2 and 3). Then, ACP will enforce the access control policy based on the user-provided context information. Next, the access control result will be forwarded to FIP and returned to the user by ACP (step 4). If the access is permitted, FIP will retrieve the AFIG from the function pool (step 5). In steps 6 and 7, FIP requests and receives the user metadata, which contains the contextual information of the user environment such as the network bandwidth, the CPU type, and so on. In step 8, FIP negotiates security-related components with the user based on the user *metadata* and the AFIG. If the user does not have the components, FIP will retrieve them from the database and deliver them to the user side (steps 9 and 10). Finally, the user application can dynamically deploy the components into the running space and start the function call (step 11).

To leverage the proposed adaptive secure access framework, we expect that the future applications will need to explicitly specify their policies, access control graph, FIPs, and, possibly, specific function module choices. The whole design space of the secure access control of future applications will include three components: 1) how to add/remove access control policies, 2) how to design multiple FIPs, and 3) how to choose different policies/functions to adapt to heterogeneous environments. Since the latter two components are very application specific, the focus of this paper is on the first component by providing system support for flexible access control and function invocation. In order to elaborate the second and third components, we present a specific UbiCAS application example to show the procedure we go through. We understand that a systematic mechanism of the latter two components in the design space is an important follow-up of the framework we proposed in the paper. Our future work will study the related topics.

3 Adaptive Access Control Module

Grimm et al. argue that *embracing contextual change* is the key to expose the dynamic changing context [18] to the application so that the adaptation can be conducted accordingly. Therefore, instead of using static access control policies, we introduce a general context in the design of the access control module. The context data represents all parameters related to the access control policy that an application defines, e.g., roles, locations, time, and so on. This design has two benefits. First, any context parameter that a real application requires can be regulated and enforced in the access control model. Second, the general context definition is evolvable to the extended access control requirement in the future. Next, we introduce the access control policy definitions and the policy graph and then present the enforcement algorithms.

3.1 Access Control Policy Definitions

Definition 1 (CT). *A CT is a tuple* $CT = (name, range, order)$, *where name is the name of the context, range is the value set of the context, and order is the method to order the values in the set.*

CT describes one context type from three aspects. First, CT^{name} is the name of the context, for example, the age of users. Second, CT^{range} is the range of the context's possible values, for example, from 10 years old to 70 years old, [10, 70]. Third, CT^{order} is the method for ordering the context values in the range. For instance, the ascending order of age as [10, 11, 12, ..., 68, 69, 70].

Definition 2 (CI). *A CI of CT* i, CT_i, *is a couple* $CI = (name, value)$, *where* $name = CT_i^{name}$, *and* $value \in CT_i^{range}$.

This definition shows one value of the CT. If the CT is represented as a vector, like the x-axis in a two-dimensional coordinate space, then the CI is a dot on the axis.

Definition 3 (function). *A function in the remote service is defined as* $F = (name, (input_i, i \in [1, n]), (output_i, i \in [1, m]), (com_i, i \in [1, k]))$, *where name is the signature of the function,* $input_i$ *is the* i*th input parameter of the function,* $output_i$ *is the* i*th output result of the function, and* com_i *is the* i*th necessary adaptive component for the function invocation. The number of inputs, outputs, and adaptive components are* n, m, *and* k, *respectively.*

This definition for function is similar to the traditional function definition except that we add the extra components that are necessary for the adaptive function invocation.

Definition 4 (service). *A service is the collection of related functions.* $Service = \bigcup Function_i$, $i \in [1, n]$.

Definition 5 (context space). *The context space is defined as* $CS = \bigcup CT_i$, $i \in [1, n]$. *For each function* f, *a context subspace (CSS) is* $CSS_f \subseteq CS$.

The context space is the whole space of the sum of all CTs. For each function, some of the CTs will be used to do the adaptive access control enforcement. These CTs form a subset of the context space, the CSS. Later in this section, we will present the data structure built upon CSS to represent the access control policies.

Definition 6 (CI node (CIN)). *A CIN is a data structure* $CIN = struct\{CI_i; Ptr(CI_j); Ptr(CI_k); \ldots\}$, *where* CI_i *is a CI of CT* i, *and* $Ptr(CI_j)$ *is a pointer directed to a CI of CT* j, *given* $i \neq j \neq k$.

Definition 7 (CI root node (CIRN)). *A CIRN is a* CIN. ϕ *denotes NULL.* $CIRN = \{CIN : \phi \rightarrow CIN\}$. *Therefore,* $CIRN \subseteq CIN$.

A CIN is built upon a CI. It has one or more pointers to point to other CIs of different CTs. For an access control policy, CIN marks each controlled value of each CT and connects them together to form the policy graph. CIRN is a special CIN that is not pointed by any other CIN. For

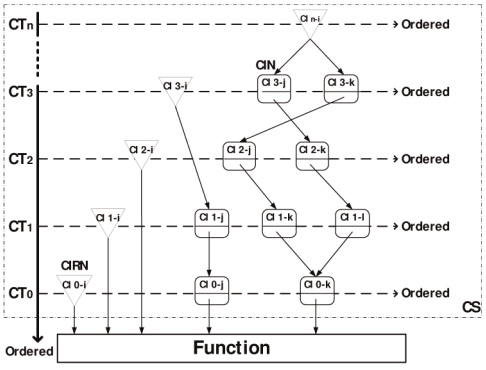


Fig. 2. The adaptive ACPG.

example, in Fig. 2, the inverted triangle shapes are CIRNs, and the rectangular shapes are CINs. Given these definitions, we can now present the structure of the ACPG.

Fig. 2 illustrates the access control policies for a function. There are $n+1$ types of CTs involved, from CT_0 to CT_n, which are ordered based on some defined standard. For example, using alphabet ascendent order for the *Role*, *Time*, *Location*, and *Speed* CTs, $CT_0 = Location$, $CT_1 = Role$, $CT_2 = Speed$, and $CT_3 = Time$. In the context space, the ordered CTs will help us locate a specific CT faster than the sequential search. The context space consists of not only the ordered CTs but also the ordered CIs of every CT. Each dotted horizontal arrow line represents the ordered CIs for the corresponding CT. For instance, CI_{0_i}, CI_{0_j}, and CI_{0_k} are three ordered CIs of CT_0. The CI and its possible pointers form a CIN, denoted as rectangular (CINs) and inverted triangle (CIRNs) shapes in the figure. A linked path from a start CIRN to the function (lower part in the figure) is an access path, formally defined as follows:

Definition 8 (access path). *An access path AP is an ordered sequence of CINs, CIN_i, $i \in [1, n]$. Symbol $\rightarrow$ means "point to," and ϕ means "none." In an access path, $CIN_n.Ptr \rightarrow Function$, and $\phi \rightarrow CIN_1$. In other words, CIN_1 is a CIRN, $CIN_i.Ptr \rightarrow CIN_{i+1}$.*

From Definition 8, we know that the access path is defined as a directed link list of CINs starting from a CIRN. For example, $CI_{3-i} \rightarrow CI_{1-j} \rightarrow CI_{0-j}$ is an access path, which means that if a user's CIs of CT_3, CT_1, and CT_0 match the values of CI_{3-i}, CI_{1-j}, and CI_{0-j}, respectively, this user is granted the access right to the function. The set of all the access paths form the access control policy of the function. For instance, in Fig. 2, the access control policy of the function has six access paths.

In order to support evolvability, we define an abstract policy class, as shown in Fig. 3, that acts as an interface to specifying the CTs required by the application access control policy. For a specific access control requirement, we need to materialize a concrete policy class that inherits the abstract class and extends the CTs. Based on these policy classes, the system will build an ACPG. A detailed example is given in Section 5. This evolvable design enables system administrators to easily and flexibly add/update access control policies. In the next section, we describe the access control enforcement algorithms.

```
abstract class Policy {
//Context term definitions
Context term      CT0;
Context term      CT1;
....
Context term      CTn;
// methods
....
}
```

Fig. 3. An abstract policy class.

3.2 Access Control Policy Enforcement

ACPG is predefined and saved in the access policy database, as shown in Fig. 1. Several related algorithms are proposed to perform the access control policy enforcement. First, when the ACP in Fig. 1 receives the context metadata from a user and retrieves ACPG from the access policy database, the CINs corresponding to the CIs of the metadata are located by the algorithm `FindCIN`, as shown in Fig. 5.

For each input CI, `FindCIN` locates the corresponding CT. Since the terms are sorted, the computational complexity is $\mathcal{O}(1)$. Then, `FindCIN` will find the input CI in this CT's ordered value space. It also takes $\mathcal{O}(1)$ steps. If the CI has a CIRN or CIN in it, the CIRN or CIN will be returned. If n is the total number of CTs and i is the number of input CIs, then the total complexity of `FindCIN` is $\mathcal{O}(n)$. Next, the ACP will use another algorithm called `AnyAccessPath`, as shown in Fig. 4. This algorithm is used to check if there is any access path existing among the CINs returned by the `FindCIN` algorithm.

```
1   AnyAccessPath ( IN: CIN1, CIN2, ... CINj
2                   CIRN1, CIRN2, ... CIRNk
3                 OUT: True or False )
4   {
5       for (m=1; m++; m<=k )
6       {
7           if (HavePath (CIRNm, CIN1, CIN2, ... CINj))
8               return True;
9       }
10      return False;
11  }


1   HavePath ( IN: CIN, CIN1, CIN2, ... CINj
2           OUT: True or False )
3   {
4       for (each Ptr of CIN )
5       {
6           if (
7                 (CIN.Ptr == Function)
8                 Or
9                 (CIN.Ptr ∈ CIN1, CIN2, ... CINj
10                And
11                HavePath(CIN.Ptr, CIN1, CIN2, ... CINj))
12            )
13                return True;
14      }
15      return False;
16  }
```

Fig. 4. The pseudocode of the `AnyAccessPath` algorithm.

```
1    FindCIN ( IN: CI1, CI2 ... CIi
2              OUT: CIN1, CIN2, ... CINj
3                   CIRN1, CIRN2, ... CIRNk )
4    {
5          for (m=1; m++; m<=i)
6          {
7                find corresponding context term CT
8                if (CT.CIm is a CIRN)
9                      return this CIRN
10               else if (CT.CIm is a CIN)
11                     return this CIN
12         }
13   }
```

Fig. 5. The pseudocode of the `FindCIN` algorithm.

For each CIRN found by the `FindCIN` algorithm, `AnyAccessPath` searches if an access path started from that CIRN exists. If one access path is found, `AnyAccessPath` returns with a true value, which means the access permission is granted to the user. Otherwise, the access is denied. Note that the core of `AnyAccessPath` is `HavePath`, which calls itself recursively to check the existence of a path starting from a CIN. Before we present the computational complexity of the algorithm, several facts are observed. Let k be the total number of CIRNs returned by the `FindCIN` algorithm. Let j be the total number of other CINs returned by the `FindCIN` algorithm. First, because a user only provides one CI for each CT, therefore, $k+j=n$, where n is the total number of CTs. Second, making a decision whether $CIN.Ptr$ is one of the CIN set $\{CIN_1, CIN_2, \ldots CIN_j\}$ is $\mathcal{O}(1)$ (line 9 of function `HavePath`) because the set is an ordered array. Let us assume that each CIN has m pointers and the height of each access path is h. Then, the best case of `AnyAccessPath` is $\mathcal{O}(n)$, while the worst case is $\mathcal{O}(mn^2)$.

Finally, the algorithm `GrantAccess` combines `FindCIN` and `AnyAccessPath` to decide the access permission (Fig. 6). Obviously, the complexity is between $\mathcal{O}(n)+\mathcal{O}(n)=\mathcal{O}(n)$ and $\mathcal{O}(n)+\mathcal{O}(mn^2)=\mathcal{O}(n(1+mn))$.

For the purpose of comparison, we also give the complexity of the conventional access control method. Usually, conventional access control methods do not utilize the complex data structure to organize the contexts. For example, it defines the access as

$$Permission := Clause_1 \bigcup Clause_2 \ldots \bigcup Clause_i,$$
$$Clause := Context_1 \bigcap Context_2 \ldots \bigcap Context_j.$$

Here, *Clause* is similar to the notion of access path in our approach. *Context* is like the CT. During the access control checking, the algorithm checks each *Clause* by comparing the *Context* value with the value received from the user. With the same parameters as we have had in the previous analysis, i.e., n CTs, k CIRNs, m pointers for each CIN, and h steps along an average path, we can easily see that there are $\mathcal{O}(km^h)$ paths or *Clauses*. The complexity of the best case is $\mathcal{O}(n)$, which happens when the first clause is satisfied. However, on the average, it requires checking $\mathcal{O}(nkm^h)$ steps, which is much worse than the worst case of our approach $\mathcal{O}(mn^2)$.

```
1    GrantAccess ( IN: CI1, CI2 ... CIi
2                  OUT: True or False )
3    {
4          if (AnyAccessPath (FindCIN (CI1, CI2 ... CIi)))
5                return True
6          else
7                return False
8    }
```

Fig. 6. The pseudocode of the `GrantAccess` algorithm.

4 Adaptive Function Invocation Module

In most of the previous efforts on access control [7], [14], [40], the procedure to access the relevant resources on the server side is not described or neglected after the access right is granted to a user. In reality, as we have argued in the introduction, the same service could have a variety of user-perceived behaviors (performance), depending on user-specific contexts such as client hardware/software configurations, network connections, and so on. For example, some applications require all Windows clients to have service pack 2 installed to access the resource even with a valid username and password. In this case, the user will either not be able to access the service or create some problems on the server side if his or her machine does not have the service pack 2 installed. To rectify this situation, we propose an adaptive function invocation approach that adjusts the function call according to the user's context. Combined with the adaptive access control module, our approach facilitates adaptive and secure access to remote services and provides the best possible performance/user experience to different users.

Specifically, adaptive components are utilized to adapt the function invocation. As shown in Fig. 1, all the adaptive components are stored in the database. According to the specific requirements of a function, FIP will request the metadata from the user. Then, FIP will decide which components are necessary for the function invocation. Finally, FIP retrieves those components from the database and delivers them to the user. Subsequently, user application can dynamically link the components into the running space and start the function invocation. Several techniques such as mobile code [20] and dynamic class loading can be used to plug in components on the fly.

The invocation requirements of a function are described as a function invocation graph, as shown in Fig. 7. Usually, a function has different requirements based on the context of access launched from a user, for example, secure and communication optimization requirements. Given the requirements, one or more components are provided to adapt the invocation according to the diverse user configurations. For instance, if one function needs content encryption, for each user hardware configuration, a specific encryption algorithm should be employed, since different encryption algorithms have various performances on different platforms [28]. The adaptive function invocation module leverages the AFIG in Fig. 7, which is similar to the access control graph, to define a directed path that connects a user's multiple metadata to access control components.

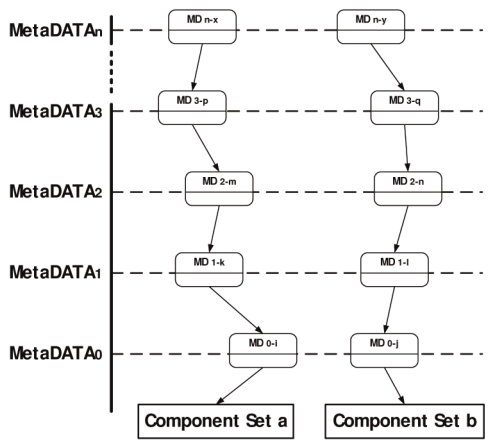


Fig. 7. The AFIG.

In Fig. 7, each metadata line represents one kind of user configuration. Many modalities could influence the function invocation, such as processor type, network bandwidth, screen resolution, and so on. For one type, there are multiple values. For instance, if *MetaData 0* represents processor type, then *MD 0-i* could refer to a PocketPC processor, and *MD 0-j* could refer to a Pentium Duo Core CPU. One linked list containing the different *MetaData* values directs to a set of components. The graph will guide FIP to find the component set for a specific user. We can see that the graph has similar structure as the ACPG in Fig. 2, so previously formalized definitions and algorithms can be easily applied here. Therefore, details about the graph definitions and search algorithms will not be repeated. Next, we will present an example of adaptive function invocation.

4.1 Adaptive Data Encryption in Function Invocation

Several functions in UbiCAS [26], such as the image load function and segmentation function, demand content encryption for patient information security/privacy specified by the HIPAA standard [21]. Although many symmetric or asymmetric encryption algorithms have been proposed, it is difficult, if not impossible, to build one single encryption protocol that performs well in such a dynamic environment as the Internet. The only way to support effective secure function invocation is to provide a flexible encryption adaptation mechanism.

Plenty of symmetric key encryption algorithms have been proposed. AES [2] and RC4 [37] are two of the most popular shared-key encryption algorithms.

In [27], we found that different encryption algorithms have different performances on various platforms. The experiment is repeated on laptop and PocketPC platforms, as shown in Fig. 8. The x-axis shows various encryption algorithms with different key lengths. The y-axis shows the total time for encrypting and decrypting 100 DICOM images. Each bar consists of two parts: the encryption time on a desktop server (lower part) and the decryption time on a laptop or PocketPC client (upper part). The hardware configurations of these devices are listed in Fig. 10. In this figure, we observe that for the laptop, the AES algorithm has better performance in terms of security (more key length) and efficiency (less total time). While for the PocketPC with constrained computing resources, the RC4 algorithm outperforms other algorithms in terms of encryption and decryption. Note that the performance on RC4 in Fig. 8a seems contrary to our intuition, this might be related to the implementation of RC4 [23]. However, even if the performance of RC4 is faster than AES on laptops, we should still choose AES from the perspective of security concerns.

Based on the above results and analysis, we choose AES and RC4 as the candidate encryption algorithms to adapt the function invocation. In the implementation, FIP will choose one encryption algorithm for a specific function invocation according to the operation system types, as shown in Fig. 9. Note that Windows XP usually runs on laptops or desktops, which normally has enough computing power to execute AES. However, for Smartphone- and PocketPC-like devices running the Windows CE operating system, previous data confirms that the AES algorithm is too heavy for them. Thus, RC4 might be a good choice for data encryption on this kind of platform. The adaptation focuses on how to choose different algorithms in the context of symmetric encryption. The procedure to set up the symmetric key(s) is beyond the scope of this paper. It is very easy to set up the symmetric keys using the Diffie-Hellman [10] key-exchange- or certificate-based authentication.

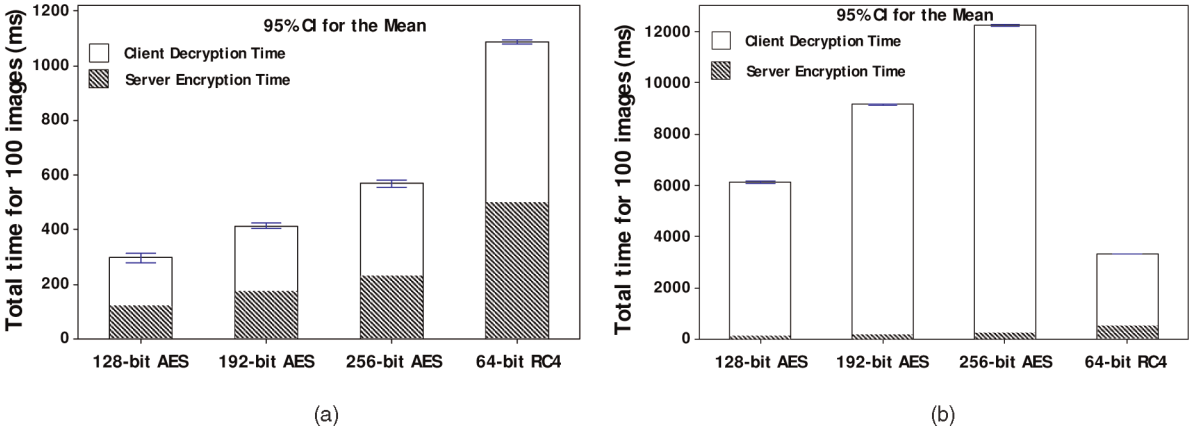


Fig. 8. Performance comparison of AES and RC4 on different platforms. (a) A laptop client. (b) A PocketPC client.

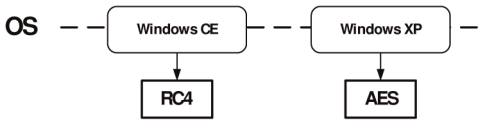


Fig. 9. Adaptive encryption function invocation graph.

5 Implementation and Evaluation

An adaptive secure access system is built in UbiCAS [26], which is a distributed CAS system. Two parts, the adaptive access control module and the adaptive function invocation module, are implemented using the Java 1.5 SDK platform. In this section, we first present a brief introduction about the UbiCAS system. Then, we show the implementation of the adaptive access control and function invocation modules. Finally, performance evaluation results are presented.

6 The UbiCAS System

CAS has broad applicability to human health. Traditional CAS systems are isolated solutions located in the operating room. Therefore, all the surgery data preparation, registration, segmentation, planning, and related operations are restricted to one physically fixed machine, which reduces the potential for telepresence and telesurgery in CAS systems. UbiCAS extends the stand-alone CAS system into distributed environments. UbiCAS allows surgeons to retrieve, review, and interpret multimodal medical images and to perform some critical neurosurgical procedures on heterogeneous devices from anywhere at any time. It has to handle several typical challenges in the mobile computing environment, such as security and privacy, multimodalities of diverse network connections and data formats, surgery-related function implementation and conciliation on heterogeneous devices, especially on resource-constrained devices like PocketPCs and Smartphones, and so on.

The adaptive secure access to the UbiCAS server deals with the above issues. The adaptive access control module provides the controlled access to the functions. Then, the adaptive function invocation module yields a secure UbiCAS function invocation. A simplified system deployment and configuration is shown in Fig. 10. The UbiCAS server, a laptop user, and a PocketPC user connect together in one local area network. The laptop user has two network interfaces: Ethernet and 802.11g wireless. The PocketPC user has an 802.11b wireless connection. The access control and function invocation modules are implemented in the UbiCAS server. They control the secure access to two functions: the DICOM image load function and the image segmentation function.

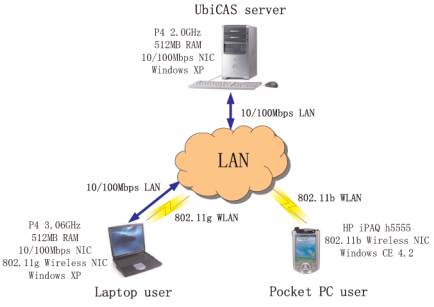


Fig. 10. A simplified UbiCAS system deployment and configuration.

```
class Policy {
//Context term definitions
Role role;
Location location;
Time time;
OS os;
// methods
....
}
```

```
class Policy-example extends Policy {
//Context term definitions
Role Doctor;
Location Office;
Time 0-24;
OS Any;
}
```

Fig. 11. The `Policy` class and an example.

6.1 Adaptive Access Control Implementation

Based on the adaptive access control module design, each function has its own ACPG. The class *Policy* is employed to describe the CTs in a policy as shown in the left side of Fig. 11. It defines four CTs as *Role*, *Location*, *Time*, and *OS* (Operating System types). The `Policy` class provides a template to define a policy for specific roles. An example policy class for doctor is shown in the right side of Fig. 11.

After all policy classes are defined and implemented, the ACPG can be formalized as shown in Fig. 12, which illustrates the overall ACPG for the segmentation function. The figure can be easily read as follows: For example, a *doctor* at *home* can only access the *segmentation* function from 0 A.M. to 8 A.M. and from 6 P.M. to 11 P.M. Following the graph, the access control enforcement algorithm can enforce the control based on each user's context data.

In the implementation, we also instrument the username and password authentication mechanism to work together with the access control module. CT instances like role, location, username, and password are provided by the user input. The operating system type is acquired by probing the system API without input from the user. Note that the location can be easily determined by other approaches such as GPS or even an 802.11 wireless network. Several attempts [4], [19], [36] address the location discovery for both indoor and outdoor scenarios. In the prototype implementation, we assume that the user provides the location automatically; however, it is trivial to leverage existing location discovery systems in a production system.

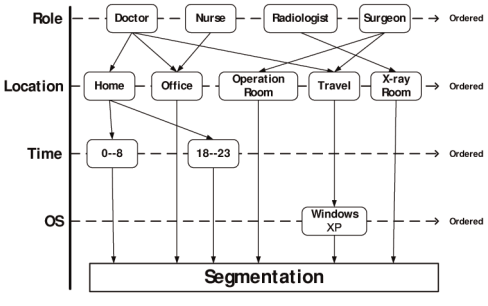


Fig. 12. Adaptive ACPG in UbiCAS.

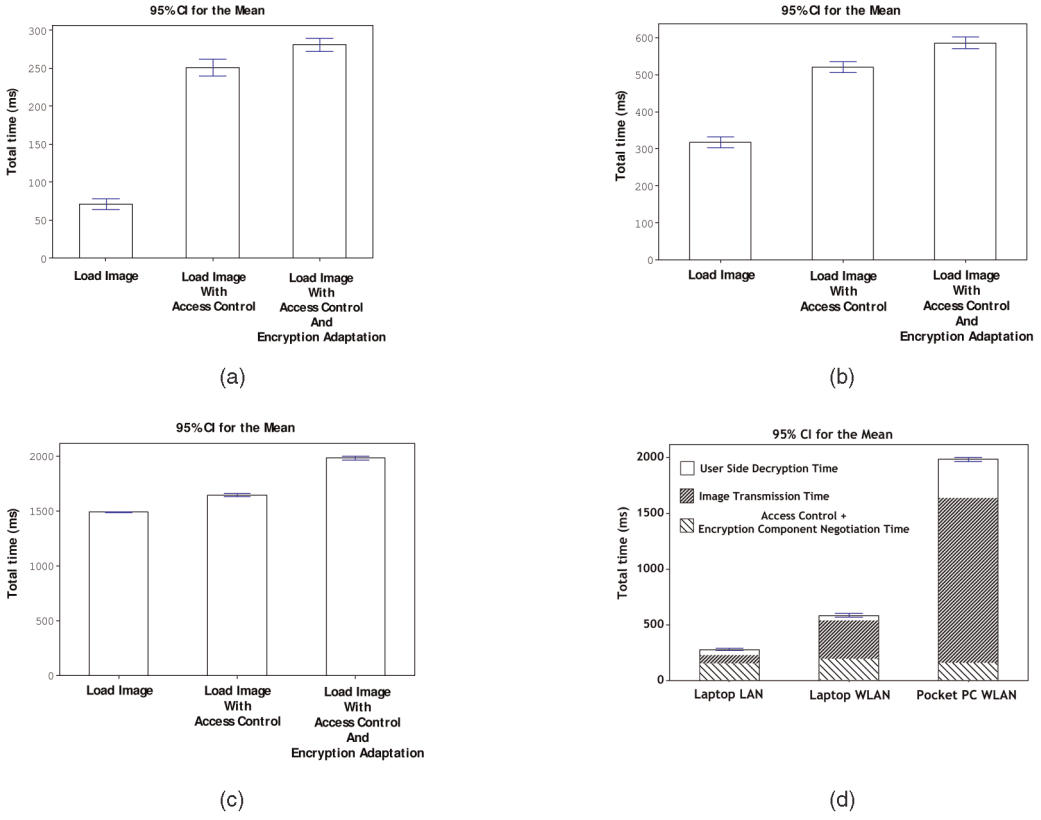


Fig. 13. Performance of adaptive secure access to load function. (a) Load function for laptop in LAN. (b) Load function for laptop in 802.11b WLAN. (c) Load function for PocketPC in 802.11b WLAN. (d) Breakdown of the image load function execution.

6.2 Adaptive Function Invocation Implementation

As mentioned in Section 4, diverse encryption algorithms have different performances on various platforms. It inspires us to adapt the encryption algorithms in the implementation of the function invocation module according to the adaptation policy shown in Fig. 9. It is reasonable to make the assumption that Windows XP is running on desktops or laptops with sufficient resources for the AES encryption algorithm. Windows CE represents the PocketPC- and Smartphone-like devices that have less power to run AES. Hence, a lightweight encryption algorithm, RC4, should be chosen instead.

6.3 Performance Evaluation

Now, we are in a position to evaluate the performance of secure access to two functions: image load and segmentation. System capacity and response times are investigated as well.

6.3.1 Image Load Function

In the secure access to the image load function, a user request will go through the two modules, adaptive access control, and function invocation, before accessing the function. For the purpose of demonstration, we reduce the number of DICOM images to four in one patient case. Each function invocation will download the four images to the user side and load these images to the GUI interface running on the client side. Fig. 13a shows three different function access scenarios for laptop users in LAN. In the first case, the image load function is called without access control and function invocation modules. In the second case, the access control module is added. Finally, both the access control and adaptive function invocation modules are called before the image load function is called. In each case, the same experiment is repeated 150 times.

In Fig. 13a, where a laptop user accesses the function in a LAN, we can see that the invocation with access control yields more overhead, an increase of about 200 ms, compared with the invocation without the access control. Since each invocation only triggers the access control module once, we think that the overhead is acceptable. The total time increment due to encryption adaptation is not significant, as shown in the rightmost bar. In Fig. 13b, where a laptop user access the function in a wireless 802.11b LAN, we can see that with the decreasing of the network bandwidth, the difference between the basic invocation and the invocation with access control is diminishing. The difference is closer in Fig. 13c, where a PocketPC user accesses the same function in 802.11b WLAN. To understand the reason for the diminishing difference, we illustrate the total time breakdown in Fig. 13d. We know that the image load function is a communication-intensive function. In low-bandwidth networks, the transmission time dominates the total time, as shown by the second part of the bars (Fig. 13d). The access control times are approximately the same in the three scenarios. Server-side encryption time is too small to be shown in the figure. The decryption time of the PocketPC user is larger than that of the laptop user due to the limited computing resource of PocketPCs.

Overall, secure access incurs 300 percent, 84 percent, and 33 percent more overhead compared with the direct image

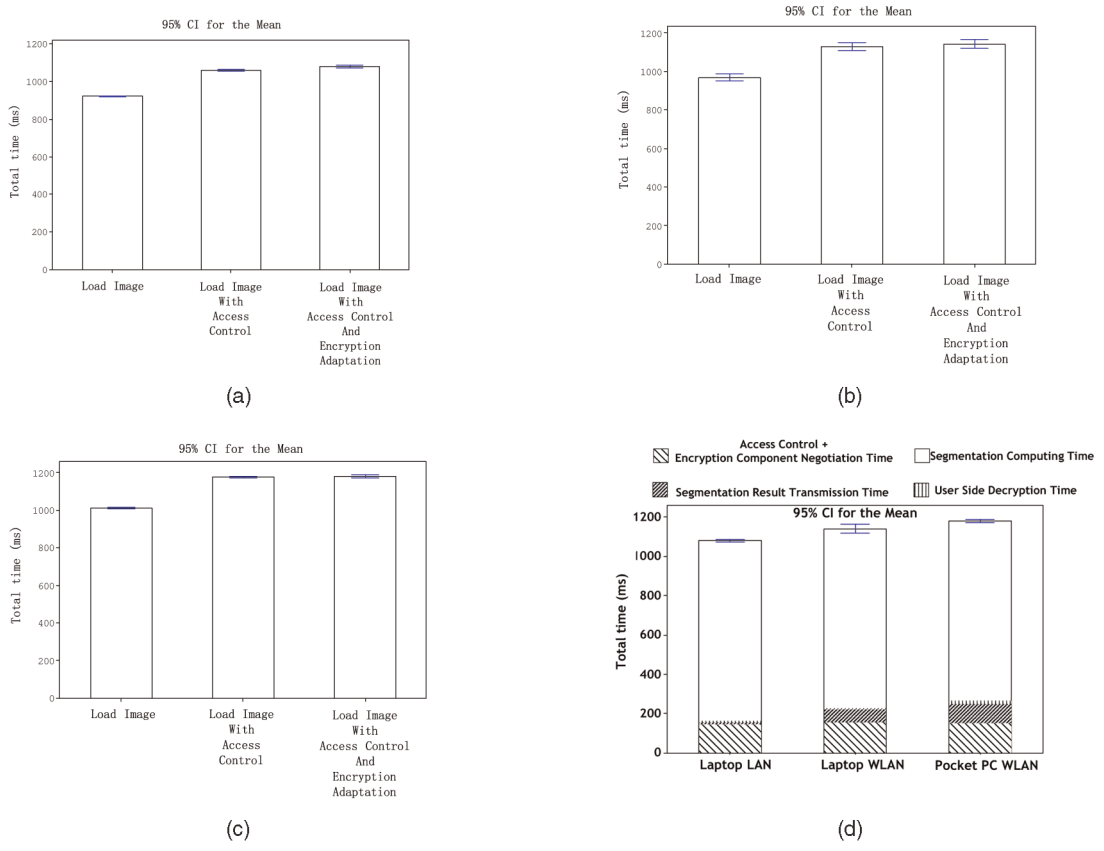


Fig. 14. Performance of adaptive secure access to the segmentation function. (a) Segmentation function for a laptop in LAN. (b) Segmentation function for a laptop in 802.11b WLAN. (c) Segmentation function for a PocketPC in 802.11b WLAN. (d) Breakdown of the segmentation function execution.

loading in three different scenarios, a laptop in LAN, a laptop in WLAN, and a PocketPC in WLAN, in terms of the total time. The overhead diminishes as the network bandwidth reduces. In summary, the two proposed modules do not jeopardize the performance of secure access to the image load function.

6.3.2 Image Segmentation Function

The image segmentation function deals with one image segmentation operation. When a user selects one pixel point of a loaded image on the device screen, the user-side program will send the two-dimensional coordinates of the point and the index of the image to the UbiCAS sever. The server will generate one result image with the same size as the original image, i.e., 134 Kbytes in UbiCAS, involving several dedicated image processing algorithms. The result image then will be sent back to the user.

Fig. 14 shows the performance of secure access to the segmentation function. Each experiment is repeated 150 times. Similar to the evaluation of the load image function, we choose three access scenarios. For three user scenarios, the three access methods are close to each other in terms of the total time, as shown in Figs. 14a, 14b and 14c, respectively. The breakdown of the total time is shown in Fig. 14d. Since there is only one image, the segmentation result image is transmitted over the network. Since segmentation is a computationally intensive function, the major part of the total time is contributed by the server-side computing time. The transmission time increases slightly as the network bandwidth decreases. The access control time remains approximately the same. Specifically, the secure access incurs 17 percent, 17 percent, and 16 percent more overhead compared with the direct function invocation in three different scenarios, which shows that for the segmentation function, the access control and adaptive function invocation introduces limited total time overhead.

6.3.3 System Capacity and Response Time

The last experiment is about the system capacity and response time evaluation, which indicates the number of users the two modules can handle simultaneously and their corresponding response time. We set up an experiment environment as shown in Fig. 15, which consists of a UbiCAS server and five request launchers running on five separate machines. Five user thread launchers issue 150 user threads to access the UbiCAS server. Each launcher starts 30 user threads sequentially following the curve of user numbers in Fig. 16. The interval between threads' start-up times is 4 seconds, so after 600 seconds, all of the 150 user threads start. For each user thread, after initialization, it keeps sending the request to the UbiCAS server every 2 seconds. For testing purposes, all the requests are identical. After receiving a request, the two modules will do the access control enforcement and encryption algorithm adaptation. The actual function invocation is not executed

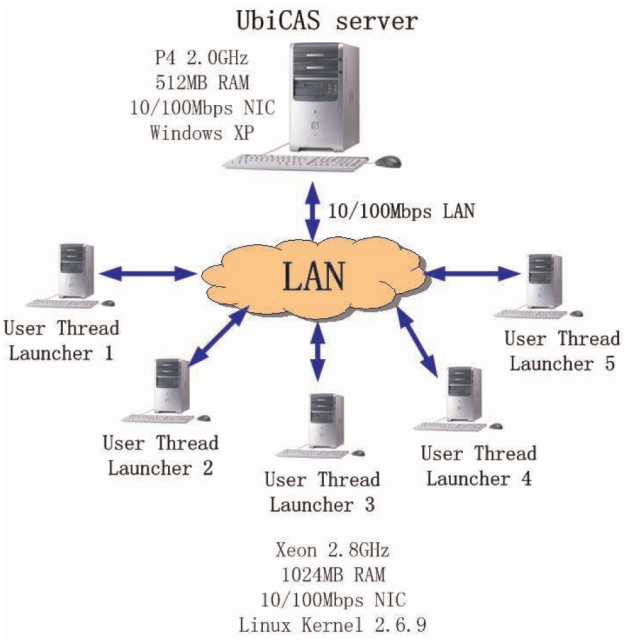


Fig. 15. System capacity experiment configuration.

since testing the capacity and response time of the two modules is the major purpose of this experiment.

From the perspective of the server side, Fig. 16 also illustrates the number of processed user requests. In the beginning, the number of processed requests goes up quickly because the server resource is underutilized. Then, it increases in a steady rate after 600 seconds due to the maximum number of user threads reached. From the curve, we observe that the system capacity can promptly catch up with the increasing number of user requests. Below, the response time from the user side confirms our observation.

The response time collected from the user side is shown in Fig. 17, which presents the response time distribution of four representative user threads during the whole testing interval, including user 1 (the first user thread), user 50 (the 50th user thread), user 100 (the 100th user thread), and user 150 (the 150th user thread). Every 2 seconds, the user thread records the response time of the request at that instant. All four figures demonstrate consistent results. It is worth noting that the first big dot (we intentionally mark it bigger than other dots) in the response time of user 1 on the y-axis is the response time of the first request. On the server side, our system needs to build up the ACPG data structure into the memory to do the access control enforcement, which requires extra time. All the successive requests will reuse the ACPG structure in memory. Therefore, all the successive response time is much lower than the first one. From the distribution of these dots (relative flat), there is no sign that the individual response time is substantially prolonged with the increasing number of user requests.

In summary, the experiment results demonstrate that two adaptive modules incur noticeable overhead for the transmission-intensive function like image load function but incur negligible overhead for computationally intensive functions like segmentation. We also observed that this extra overhead will diminish as the network speed decreases. This matches perfectly with the fact that more

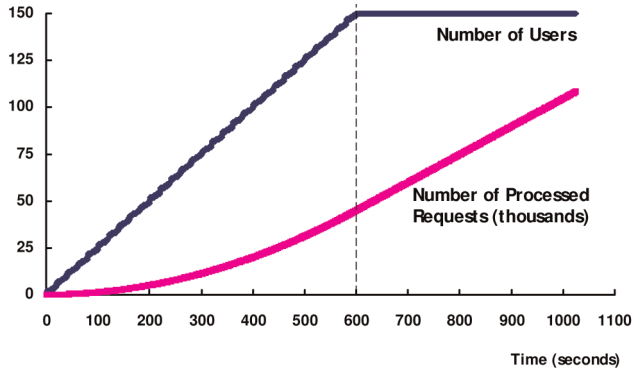


Fig. 16. Evaluation of system capacity.

and more users want to access remote services using resource-constrained devices ubiquitously.

7 Related Work

Our work shares its goal with several recent efforts that attempt to enforce access control for various objects in distributed environments and to inject adaptive functionality into the application. We categorize related research work into two groups: *access control* and *adaptation*.

Access control. Access control decides whether to grant the access right of the object to the principal. In [1], Abadi et al. propose the concepts, protocols, and algorithms for access control in distributed systems from a logical perspective. It also provides a logical language for access control lists and theories that decide whether requests should be granted. Sandhu et al. [40] introduce the role-based access control (RBAC) model, which efficiently associates permissions with roles rather than users to greatly simplify security management for administrators. Distributed RBAC (dRBAC) [14] is a scalable decentralized trust management and access control mechanism for systems that span multiple administrative domains. Temporal RBAC (TRBAC) [7] is an extension of the RBAC model. TRBAC supports periodic role enabling and disabling and temporal dependencies among such actions, expressed by means of role triggers, which are related to a different delay time. Generalized RBAC (GRBAC) [31] leverages and extends the power of traditional RBAC by incorporating subject roles, object roles, and environment roles into access control decisions. Two extensions of RBAC, GEO-RBAC [8] and P-RBAC [33], were proposed. Securing access to data in location-based services and mobile applications requires the definition of spatially aware access control systems. In GEO-RBAC, spatial entities are used to model objects, user positions, and geographically bounded roles. Roles are activated based on the position of the user. Based on GEO-RBAC, the same authors design an administration model [9] to meet the challenging requirements over policy administration for context-aware RBAC. The model is based on the notion of the hierarchy of spatial domains, which is an entity grouping objects based on organizational and spatial proximity criteria. Privacy has been acknowledged to be a critical requirement for a mobile environment. P-RBAC extends the well-known RBAC model and provides full

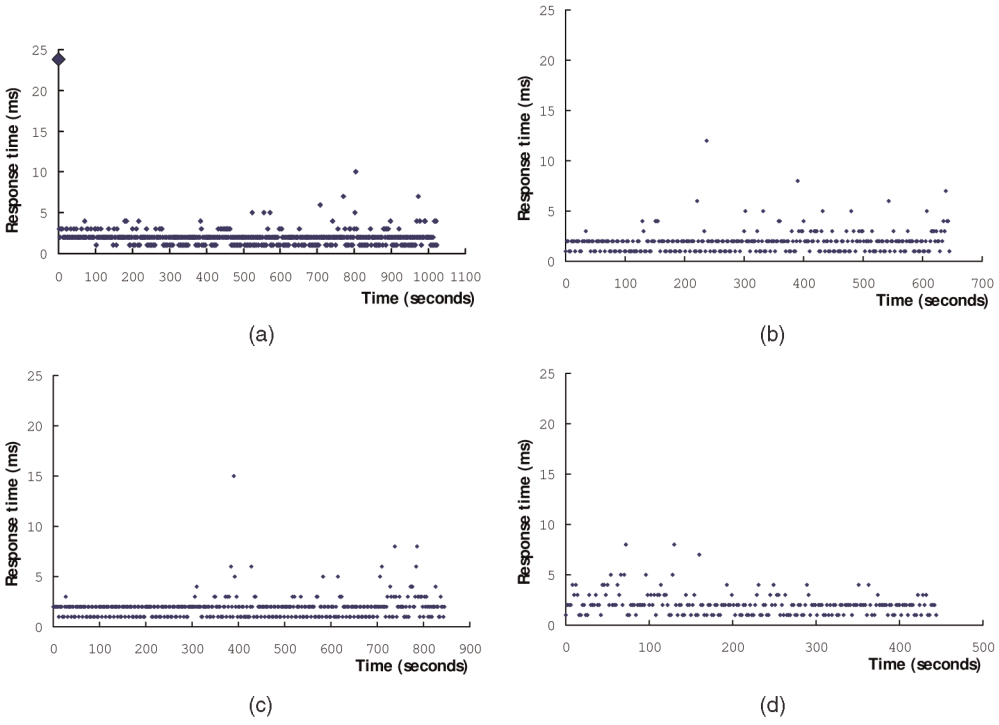


Fig. 17. The distribution of response time of four representative users. (a) Response time for the first user. (b) Response time for the 50th user. (c) Response time for the 100th user. (d) Response time for the 150th user.

support for expressing highly complex privacy-related policies by taking into account features like purposes and obligations. Those efforts focus on a specific access control model, while our work emphasizes adaptation of access policies based on contexts, complementing well with previous efforts.

Team-based access control (TMAC) [43] was first proposed by Thomas to provide a natural way to model access control for collaborative activities best accomplished by teams of users. C-TMAC [16] extends TMAC by using general contextual information. Such contextual information can include the time of access, the location from which access is requested, the location where the object to be accessed resides, transaction-specific values that dictate special access policies, and so on. Kumar et al. [25] extend the RBAC by introducing the notions of role context and context filters to make RBAC sensitive to the context of an attempted operation. Edjlali et al. propose the history-based access control for mobile code [11]. The key idea is there is to maintain a selective history of the access requests made by individual programs and to use the history to improve the security differentiation. This approach provides a nice means for adaptation and complements the proposed adaptive secure access very well.

In [22], a dynamic context-aware security infrastructure is proposed to provide flexible on-demand authentication extensible context-aware access control to healthcare applications. Zhang et al. [47] present a delegation framework that can be used within the security framework of healthcare applications. Wilikens et al. discuss how to apply CBAC to healthcare applications in [45].

Several research efforts about access control in Web services and Service-Oriented Architecture (SOA) have also been done, such as [32] and [42]. The Web Services Business Process Execution Language (WSBPEL) is now a core part of the orchestration process, which is the technology used to build applications based on Web services. However, WSBPEL does not provide a means for specifying human interactions, even less their access-control requirements. In [42], a language is proposed to specify the extension to WSBPEL, allowing the reuse of existing BPEL engines and specifying these extensions within the main BPEL script, hence preserving a global view of the process. In [32], a security punctuation framework is proposed. The main idea is that for streaming applications, security restrictions are not persistently stored on the DSMS server. Instead, they are streamed together with the data because the contexts, and with them the access control policies on the real-time data, may rapidly change. The access control policies are expressed via security constraints (called security punctuation) and are embedded into data streams. In this way, the flexibility, dynamics, and speed of enforcement are greatly improved.

Our work is different from above-mentioned previous efforts in the following ways. First, we do not introduce one specific context, like the location of time into the access control model. Actually, we systematically propose a general secure access mechanism to integrate any context that the application possibly needs. It gives the application great flexibility to scale the access control policy to include extended context items. To our knowledge, none of the existing access control models propose the enhanced enforcement algorithm to improve the access control procedure in terms of the computing time complexity. However, several previous efforts complement perfectly for our work. For example, P-RBAC provides full support for expressing highly complex privacy-related policies, which exactly match our needs of expression of the policy. Finally,

combining access control and adaptive function invocation is novel, and it is one of the early efforts enabling secure access of remote services in the coming service-oriented computing era.

Adaptation. In distributed heterogeneous environment, adaptation is pervasive. From the Internet topology's point of view, adaptation functionality can be introduced either at the end points or distributed on intermediate nodes. Odyssey [34], Rover [24], and InfoPyramid [30] are examples of systems that support end-point adaptation. Conductor [46] and CANS [15] provide an application-transparent adaptation framework that permits the introduction of arbitrary adaptors in the data path between applications and end services. Our work makes the adaptation happen at the end point to avoid the deployment hassle of a significant infrastructure change.

From the network structure's perspective, there are two issues: whether adaptation functionality is introduced at the network layer with application transparency or at the application level with application awareness. Systems such as transformer tunnels [41] and protocol boosters [29] are examples of application-transparent adaptation efforts that work at the network level. Such systems can cope with localized changes in network conditions but cannot adapt to behaviors that differ widely from the norm. Moreover, their transparency hinders composability of multiple adaptations. Similar efforts also work at the application level. The cluster-based proxies in BARWAN/Daedalus [12], TACC [13], and MultiSpace [17] are examples of systems where application-transparent adaptation happens in intermediate nodes (typically, a small number) in the network. Active Services [3] extends these systems to a distributed setting by permitting a client application to explicitly start one or more services on its behalf that can transform the data it receives from an end service.

8 Conclusions

In this paper, we propose an adaptive secure access mechanism for accessing remote services. Compared with previous efforts, which handle predefined static contexts for access control, our approach is able to integrate application-oriented access control contexts into the system and to dynamically evolve the access control policy to handle future system requirements. Besides access control, we also raise another important issue of adaptive function invocation, which has not been addressed in any previous work. We have successfully implemented our models in a distributed CAS system called UbiCAS. The performance evaluation on different configurations shows that our approach provides efficient secure access to remote services with an acceptable overhead. To our knowledge, this is the first effort on system support for adaptive secure access to remote services. We hope that this paper will raise the issue to the services computing community and bring more investigation in this direction. In the future, we plan to design an access control policy description language to work with our proposed enforcement algorithm. Enriching the function invocation module with more intelligently reactive adaptation is also an interesting research direction.

Acknowledgments

This research was supported in part by the Michigan Life Science Corridor Grant and US National Science Foundation CAREER Award CCF-0643521.

References

[1] M. Abadi, M. Burrows, B. Lampson, and G. Plotkin, "A Calculus for Access Control in Distributed Systems," *ACM Trans. Programming Languages and Systems,* pp. 706-734, Sept. 1993.

[2] *Advanced Encryption Standard,* http://csrc.nist.gov/CryptoToolkit/aes/, 2008.

[3] E. Amir, S. McCanne, and R. Katz, "An Active Service Framework and Its Application to Real-Time Multimedia Transcoding," *Proc. ACM SIGCOMM '98,* Aug. 1998.

[4] P. Bahl and V. Padmanabhan, "RADAR: An In-Building RF-Based User Location and Tracking System," *Proc. IEEE INFOCOM '00,* Apr. 2000.

[5] B. Benatallah, F. Casati, and F. Toumani, "Web Service Conversation Modeling: A Cornerstone for E-Business Automation," *IEEE Internet Computing,* pp. 46-54, 2004.

[6] D. Berardi, D. Calvanese, G. Giacomo, M. Lenzerini, and M. Mecella, "Automatic Service Composition Based on Behavioral Descriptions," *Int'l J. Cooperative Information Systems,* pp. 333-376, 2005.

[7] E. Bertino and P. Bonatti, "TRBAC: A Temporal Role-Based Access Control Model," *ACM Trans. Information and System Security,* pp. 191-223, Aug. 2001.

[8] M. Damiani, E. Bertino, B. Catania, and P. Perlasca, "Geo-RBAC: A Spatially Aware RBAC," *ACM Trans. Information System Security,* Feb. 2007.

[9] M. Damiani, C. Silvestri, and E. Bertino, "Hierarchical Domains for Decentralized Administration of Spatially-Aware RBAC Systems," *Proc. Third Int'l Conf. Availability, Reliability and Security (ARES '08),* Mar. 2008.

[10] W. Diffie and M. Hellman, "Multiuser Cryptographic Techniques," *IEEE Trans. Information Theory,* vol. 22, no. 1, pp. 644-654, Nov. 1976.

[11] G. Edjlali, A. Acharya, and V. Chaudhary, "History-Based Access Control for Mobile Code," *Proc. Fifth ACM Conf. Computer and Comm. Security (CCS '98),* pp. 38-48, Nov. 1998.

[12] A. Fox, S. Gribble, Y. Chawathe, and E.A. Brewer, "Adapting to Network and Client Variation Using Infrastructural Proxies: Lessons and Perspectives," *IEEE Personal Comm.,* vol. 5, no. 4, pp. 10-19, http://www.cs.washington.edu/homes/gribble/papers/adapt.ps.zip, Aug. 1998.

[13] A. Fox, S. Gribble, Y. Chawathe, E.A. Brewer, and P. Gauthier, "Cluster-Based Scalable Network Services," *Proc. 16th ACM Symp. Operating Systems Principles (SOSP '97),* Oct. 1997.

[14] E. Freudenthal, T. Pesin, L. Port, E. Keenan, and V. Karamcheti, "DRBAC: Distributed Role-Based Access Control for Dynamic Coalition Environments," *Proc. 22nd Int'l Conf. Distributed Computing Systems (ICDCS '02),* July 2002.

[15] X. Fu, W. Shi, A. Akkerman, and V. Karamcheti, "CANS: Composable, Adaptive Network Services Infrastructure," *Proc. Third Usenix Symp. Internet Technologies and Systems (USITS '01),* pp. 135-146, Mar. 2001.

[16] C. Georgiadis, I. Mavridis, G. Pangalos, and R. Thomas, "Flexible Team-Based Access Control Using Contexts," *Proc. Sixth ACM Symp. Access Control Models and Technologies (SACMAT '01),* May 2001.

[17] S.D. Gribble, M. Welsh, E.A. Brewer, and D. Culler, "The MultiSpace: An Evolutionary Platform for Infrastructual Services," *Proc. Usenix Ann. Technical Conf.,* June 1999.

[18] R. Grimm, J. Davis, E. Lemar, A. Macbeth, S. Swanson, T. Anderson, B. Bershad, G. Borriello, S. Gribble, and D. Wetherall, "System Support for Pervasive Applications," *ACM Trans. Computer Systems,* pp. 421-486, Nov. 2004.

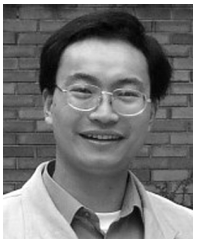
[19] A. Haeberlen, E. Flannery, A. Ladd, A. Rudys, D. Wallach, and L. Kavraki, "Practical Robust Localization over Large-Scale 802.11 Wireless Networks," *Proc. ACM MobiCom,* 2004.

[20] D. Halls, "Applying Mobile Code to Distributed Systems," PhD dissertation, Computer Laboratory, Univ. of Cambridge, 1997.

[21] *The American Health Insurance Portability and Accountability Act,* http://www.hipaa.org/, 2008.

[22] J. Hu and A. Weaver, "Context-Aware Security Infrastructure for Distributed Healthcare Applications," *Proc. First Workshop Pervasive Security, Privacy and Trust (PSPT '04),* Aug. 2004.
[23] *Java Implementation of RC4,* http://www.insanityflows.net/archive/index.php?title=RC4.java, 2008.
[24] A.D. Joseph, J.A. Tauber, and M.F. Kasshoek, "Mobile Computing with the Rover Toolkit," *IEEE Trans. Computers,* special issue on mobile computing, vol. 46, no. 3, pp. 337-352, Mar. 1997.
[25] A. Kumar, N. Karnik, and G. Chafle, "Context Sensitivity in Role-Based Access Control," *ACM SIGOPS Operating Systems Rev.,* July 2002.
[26] H. Liu, H. Lufei, W. Shi, and V. Chaudhary, "Towards Ubiquitous Access of Computer-Assisted Surgery Systems," *Proc. 28th Ann. Int'l Conf. IEEE Eng. in Medicine and Biology Soc. (EMBS '06),* Aug. 2006.
[27] H. Lufei and W. Shi, "An Adaptive Encryption Protocol in Mobile Computing," *Wireless Network Security,* Springer, 2006.
[28] H. Lufei and W. Shi, "Fractal: A Mobile Code Based Framework for Dynamic Application Protocol Adaptation," *J. Parallel and Distributed Computing,* pp. 887-906, July 2006.
[29] A. Mallet, J. Chung, and J. Smith, "Operating System Support for Protocol Boosters," *Proc. Fourth Int'l Workshop High Performance Protocol Architectures (HIPPARCH '97),* June 1997.
[30] R. Mohan, J.R. Simth, and C. Li, "Adapting Multimedia Internet Content for Universal Access," *IEEE Trans. Multimedia,* vol. 1, no. 1, pp. 104-114, Mar. 1999.
[31] M. Moyer and M. Ahamad, "Generalized Role-Based Access Control," *Proc. 21st Int'l Conf. Distributed Computing Systems (ICDCS),* 2001.
[32] R. Nehme, E. Rundensteiner, and E. Bertino, "A Security Punctuation Framework for Enforcing Access Control on Streaming Data," *Proc. 24th Int'l Conf. Data Eng. (ICDE '08),* Apr. 2008.
[33] Q. Ni, A. Trombetta, E. Bertino, and J. Lobo, "Privacy-Aware Role Based Access Control," *Proc. 12th ACM Symp. Access Control Models and Technologies (SACMAT 07),* June 2007.
[34] B.D. Noble, "Mobile Data Access," PhD dissertation, School of Computer Science, Carnegie Mellon Univ., http://mobility.eecs.umich.edu/papers/diss.pdf, May 1998.
[35] S. Paurobally and N. Jennings, "Protocol Engineering for Web Services Conversations," *Int'l J. Eng. Applications of Artificial Intelligence,* vol. 18, 2005.
[36] N. Priyantha, A. Chakraborty, and H. Balakrishnan, "The Cricket Location-Support System," *Proc. ACM MobiCom,* 2000.
[37] RC4 RFC 3268, http://www.faqs.org/rfcs/rfc3268.html/.
[38] *Remote Method Invocation,* http://java.sun.com/javase/technologies/core/basic/rmi/whitepaper/index.jsp, 2008.
[39] *Remote Procedure Call,* http://tools.ietf.org/html/rfc707, 2008.
[40] R. Sandhu, E. Coyne, H. Feinstein, and C. Youman, "Role-Based Access Control Models," *Computer,* pp. 38-47, Feb. 1996.
[41] P. Sudame and B. Badrinath, "Transformer Tunnels: A Framework for Providing Route-Specific Adaptations," *Proc. Usenix Technical Conf.,* June 1998.
[42] J. Thomas, F. Paci, E. Bertino, and P. Eugster, "User Tasks and Access Control over Web Services," *Proc. IEEE Int'l Conf. Web Services (ICWS '07),* July 2007.
[43] R. Thomas, "Team-Based Access Control (TMAC): A Primitive for Applying Role-Based Access Controls in Collaborative Environments," *Proc. Second ACM Workshop Role-Based Access Control (RBAC '97),* Oct. 1997.
[44] W3C Consortium, *Simple Object Access Protocol (SOAP) 1.1,* http://www.w3.org/TR/SOAP/, 2000.
[45] M. Wilikens, S. Feriti, A. Sanna, and M. Masera, "A Context-Related Authorization and Access Control Method Based on RBAC: A Case Study from the Health Care Domain," *Proc. Seventh ACM Symp. Access Control Models and Technologies (SACMAT '02),* June 2002.
[46] M. Yarvis, A. Wang, A. Rudenko, P. Reiher, and G.J. Popek, "Conductor: Distributed Adaptation for Complex Networks," *Proc. Seventh Workshop Hot Topics in Operating Systems (HotOS '99),* http://lasr.cs.ucla.edu/reiher/papers/yarvis.ps, Mar. 1999.
[47] L. Zhang, G. Ahn, and B. Chu, "A Role-Based Delegation Framework for Healthcare Information Systems," *Proc. Seventh ACM Symp. Access Control Models and Technologies (SACMAT '02),* June 2002.

Hanping Lufei received the bachelor's degree in electrical engineering from the Huazhong University of Science and Technology (HUST), China, in 1998, the master's degree in electrical engineering from the University of Toledo in 2001, and the PhD degree in computer science from Wayne State University, Detroit. His current research focuses on QoS, systems security, access control, and trust management in a mobile computing environment. He is also interested in computing enhancement for handheld devices and resource management in distributed systems.



Weisong Shi received the BS degree in computer engineering from Xidian University in 1995 and the PhD degree in computer engineering from the Chinese Academy of Sciences in 2000. He is an associate professor of computer science at Wayne State University, Detroit. His current research focuses on mobile computing, distributed systems, and high-performance computing. He has published more than 80 peer-reviewed journal and conference papers in these areas. He is the author of the book *Performance Optimization of Software Distributed Shared Memory Systems* (High Education Press, 2004). He has also served on the technical program committees of several international conferences, including WWW, ICPP, and MASS. He received the Microsoft Fellowship in 1999, the President Outstanding Award of the Chinese Academy of Sciences in 2000, one of 100 outstanding PhD dissertations (China) in 2002, the Faculty Research Award of Wayne State University in 2004 and 2005, and the Best Paper Award of ICWE 2004 and IPDPS 2005. He is a recipient of the US National Science Foundation CAREER award. He is member of the IEEE and the IEEE Computer Society.

Vipin Chaudhary received the BTech (Hons) degree in computer science and engineering from the Indian Institute of Technology (IIT), Kharagpur, in 1986 and the MS degree in computer science and the PhD degree in electrical and computer engineering from the University of Texas at Austin in 1989 and 1992, respectively. He is an associate professor of computer science and engineering at the New York State Center of Excellence in Bioinformatics and Life Sciences, University at Buffalo, State University of New York (SUNY). Earlier, he was the senior director of advanced development at Cradle Technologies, Inc. Prior to that, he was the chief architect with Corio, Inc. In addition, he is on the advisory boards of several start-up companies. His current research interests are in the areas of computer-assisted diagnosis and interventions, medical image processing, grid and high-performance computing and its applications in computational biology and medicine, embedded systems architecture and applications, and sensor networks and pervasive computing. He was awarded the prestigious President of India Gold Medal in 1986 for securing the first rank among graduating students at IIT. He is a member of the IEEE and the IEEE Computer Society.