Journal of Network and Computer Applications ∎ (∎∎∎∎) ∎∎∎–∎∎∎

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

# A trusted decentralized access control framework for the client/server architecture

Weili Han [a,*], Min Xu [b], Weidong Zhao [a], Guofu Li [a]

[a] Software School, Fudan University, Shanghai, PR China
[b] Department of Computer Science, George Mason University, VA, USA

ARTICLE INFO

*Article history:*
Received 25 March 2009
Received in revised form
8 November 2009
Accepted 15 December 2009

*Keywords:*
Trusted computing
Trusted decentralized access control
Context-aware access control

ABSTRACT

This paper proposes a trusted decentralized access control (TDAC) framework for the client/server architecture. As the fundamental principle, TDAC enforces access control policies at the client side and protects sensitive objects at the server side by leveraging trusted computing technologies. Compared with the previous work of Sandhu and Zhang (2005), TDAC uses fewer requirements for trusted components. To implement TDAC, we design a private trusted reference monitor that runs at the client side, evaluates an access control request, and signs a temporary access control credential for a client application trustworthily; we also design a master reference monitor that runs at the server side, evaluates the request from the client application only according to the temporary access control credential. As a typical application, TDAC can protect client's private context data in subject-context aware access control.

## 1. Introduction

Access control is a key mechanism to finer-granularity protection in an information system (Kane and Browne, 2006; Sandhu et al., 1996; Tolone et al., 2005). In a traditional client/server application, protected sensitive objects are stored at the server side, and a reference monitor, which evaluates the access control requests, runs at the server side, too.

However, it is imperative to enforce access control policies at the client side. Enforcing access control policies at the client side can protect sensitive data stored which could be either the data moved from another host (Sandhu and Zhang, 2005; Sandhu et al., 2006), or the subject's private data, such as entire visiting log or location of a subject (normally, a user) at the client side.

Sandhu and Zhang (2005) proposed an access control mechanism for the P2P (peer-to-peer) environment supported by trusted computing technologies. The work proposed by Sandhu et al. assumed the systems of peers are protected by trusted computing technologies, and enforce access control policies at one peer side, once the sensitive data moved from another peer to the peer, which consumes sensitive data. But their architecture requires too many trusted components. Their architecture requires *sealed storage*, which stores the sensitive objects, and *trusted applications*, such as VoIP applications, which are ensured by protected runtime environment. Generally, too many trusted components will dramatically reduce the applicability, especially the current researchers of trusted computing enforcement are aiming to run only a few trusted processes (McCune et al., 2007; Chen et al., 2008; Yang and Shin, 2008).

As the main contribution, this paper proposes an access control framework, namely trusted decentralized access control (TDAC), for the client/server architecture, with fewer requirements for trusted components compared with the previous work of Sandhu and Zhang (2005). TDAC evaluates access control requests according to access control policies at the client side, but not requires two trusted components: *sealed storage* and *trusted applications*. In TDAC, the server's sensitive data are stored at the server side, while the subject's private data are stored at the client side. There are two reference monitors in TDAC. One is the private trusted reference monitor (PTRM) that runs at the client side, evaluates an access control request, and signs a temporary access control credential (TACC) trustworthily. The other is the master reference monitor (MRM) that runs at the server side, evaluates the request according to the TACC. TDAC can be applied to protect both server's sensitive data and subject's private data in context-aware access control. trusted computing technologies (TCG, 2009, 2005; Garfinkel et al., 2003; McCune et al., 2007; Chen et al., 2008; Yang and Shin, 2008; Intel, 2008) provide the feasibility to ensure that the PTRM, as one trusted process, trustworthily runs at the client side, because these technologies, such as Trusted eXecution Technology (TXT) (Intel, 2008), ensure the trust of hardware, support protected execution environments.

* Corresponding author. Tel.: +86 21 51355388.
*E-mail addresses:* wlhan@fudan.edu.cn (W. Han), mxu@gmu.edu (M. Xu). wdzhao@fudan.edu.cn (W. Zhao), 0361024@fudan.edu.cn (G. Li).

1084-8045/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jnca.2009.12.012

The rest of the paper is organized as follows: Section 2 introduces the related work, including trusted computing, context-aware access control and the motivation of TDAC; Section 3 presents a framework for TDAC; Section 3 also analyzes the security issues in the framework; Section 4 introduces a prototype and evaluates the performance of the implementation; Section 5 describes typical applicable scenarios to apply TDAC; Section 6 discusses some issues in TDAC; finally, Section 7 summarizes the paper and presents our future work.

## 2. Related work and motivation

### 2.1. Trusted computing technologies and access control

Trusted computing technologies (TCG, 2009, 2005) proposed by trusted computing group (TCG) leverage a trusted platform module (TPM) as the root of *trust* in open computing environments. In trusted computing, the TPM, which is a tamper-protection hardware module, is usually a chip embedded in a motherboard. With regard to the tamper-resistant property of TPM, TCG considers a platform owner as a potential attacker (Mao et al., 2006), and tries to prevent the attacker from bypassing or breaching the protection running in an open platform. In contrast to the conventional security mechanisms which function against external or less privileged attackers, the owner of the platform is usually in a privileged position, i.e., a stronger attacker and thereby it is harder to prevent the owner from unmeant mistakes or vicious actions. In trusted computing environment, a service can verify the remote platform configuration through attestation, then trusts the remote platform.

In addition to a TCG compliant TPM, the TXT: Trusted eXecution Technology (or LaGrande Technology (LT)) of Intel (2008) includes an extended CPU which enables software domain separation and protection, and an extended chipset which enables protected graphics and basic I/O devices (i.e., keyboard/mouse). AMD (2008) also provides a similar trusted execution technology: SVM (secure virtual machine).

Based on these trusted hardware technologies, researchers have proposed the implementation of trusted execution environment which can run a few trusted processes at the client side. McCune et al. (2007) proposed a minimized TCB (trusted computing base) code solution to implement one trusted program based on AMD technologies. Garfinkel et al. (2003) proposed a TVMM-based (trusted virtual machine monitor) trusted execution solution. Chen et al. (2008) and Yang and Shin (2008) introduced how to use the TVMM to implement a trusted process in a commodity operating system.

Trusted computing environment requires novel mechanisms to implement access control in computing scenarios, such as P2P environment. Sandhu and Zhang (2005) proposed an access control mechanism to enforce access control policies in the peer-to-peer (P2P) environment by leveraging trusted computing technologies through assumptions that systems of peers are protected by trusted computing technologies.

In this paper, we present TDAC, which is motivated from Sandhu's work, but is different from Sandhu's work. Our work focuses on protection of subject-context data in the traditional client/server architecture, with fewer requirements for trusted components. The further work of Sandhu et al. (2006) to enforce access control policies at the client side still required *sealed storage* and *trusted applications*, which are not required in TDAC.

### 2.2. Context aware access control

Context data can be divided into three categories (Kapsalis et al., 2006; Gu et al., 2005; Han et al., 2008): subject context data, object context data and other context data.

- The subject context data include client's location or visiting log. The visiting log could be created by one trusted service and stored at the client side. For example, a service creates and signs user's visiting records, and sends the records to a client platform that will store them. When the user wants to visit another server, he or she could use the visiting records to get a more personalized context aware service.
- The object context data include the created time, visited log.
- The other context data include system time, system load.

The concept of context aware enables the expression of more flexible access control policies than traditional access control, such as discretionary access control, as well as role-based access control. Because the context data, besides subject reference and object reference, are important factors in making access control evaluation, context-aware access control (Han et al., 2005; Hulsebosch et al., 2005; Kapsalis et al., 2006; McDaniel, 2003; OASIS, 2005; Strembeck and Neumann, 2004; Wedde and Lischka, 2004; Zhang et al., 2006) is proposed. OASIS (2005) considered context data in its standards. Strembeck and Neumann (2004) introduced an integrated approach to implement context in role-based access control. McDaniel (2003) proposed a method to define context-sensitive authorization policies. Their research works have resolved some very important issues in context-aware access control. These issues include the overview of context-aware access control policy description and context-aware access control policy implementation.

Privacy is an important issue in context-aware access control. P3P provides some important work to protect privacy (W3C, 2009). In the access control field, Blauton and Atallah (2005) introduced a portable and flexible privacy-preserving access right for large online data repositories. Li et al. (2005) proposed a privacy protection mechanism and its related policy language in Automatic Trusted Negotiation which uses cryptographic credential. Squicciarini et al. (2007) and Cautis (2007) also considered the privacy issues in access control.

In a general design of context aware system, context data are dealt with at the server side, to ensure the trustworthiness of processing of context data, but the privacy protection of these context data only depends on the ethic of the server's master (i.e., a server could leak even sell these context data). We, therefore, design TDAC to protect these subject-context data by leveraging current trusted computing technologies.

### 2.3. Motivation

Sandhu and Zhang (2005) opened a way to leverage trusted computing technologies in the access control field. However, the solution proposed in Sandhu and Zhang (2005) requires a full-protected runtime environment to ensure the trustworthiness of applications, (or *trusted applications*).

Compared with the previous work of Sandhu and Zhang (2005), TDAC proposed in this paper uses fewer requirements for trusted components to enforce access control policies at the client side. TDAC do not require both *sealed storage* and *trusted applications*.

As the most important contribution in this paper, it is very important of TDAC to use the fewer requirements for trusted components. First, the fewer requirements means lower cost to deploy TDAC. To realize *sealed storage* is still developing and require more peripheral hardware besides a TPM (trusted platform module), because the TPM cannot deal with encryption for huge data. To deploy the peripheral hardware is obviously high cost if we deploy the hardware as one component of infrastructure. Second, the fewer requirements means more applicabilities . Especially, TDAC do not require *trusted applications*, which usually are very hard to realized due to the complexity of these applications. A common application, therefore, can leverage TDAC to enforce access control policies at the client side. Thus, it is very important to promote trusted computing technologies, if we can use fewer requirements for trusted components in TDAC.

#### 2.3.1. *To support subject-context aware access control*

To implement subject-context aware access control policies, how to fetch these subject context data and how to protect the privacy of these context data are two important issues. TDAC can provide a potential solution to resolve the above issues by fetching and consuming the context data at the client side, not at the server side.

As a comparison, a feasible but poor-privacy-aware solution to context aware access control is that the server stores these subject context data and evaluates access control requests at the server side. The disadvantages of this solution are:

- The server can only store part of subject context data and must fetch necessary subject context data. It could be difficult for the server to fetch the context data, such as all visiting log, stored at the client side, or at another server side.
- The server must convince a user that the server will protect the privacy of the subject context data. However, even though the server does not intentionally leak or sell these context data, a hacker can bypass the safeguards of the server's to steal these private data. Therefore, it is difficult to convince the user.

We propose a privacy-aware feasible solution, TDAC, where the client stores and fetches these context data and a reference monitor (PTRM) evaluates access control at the client side. We leverage trusted computing technologies to resolve the key issue of *Trust* during the process of the PTRM running. Based on these trusted computing technologies, the PTRM fetches subject context data, evaluates an access control request, and signs the temporary access control credential (TACC) for the request trustworthily. Another reference monitor, named master reference monitor (MRM), at the server side only evaluates the request according to the TACC. Therefore, TDAC provides a potential private data protection solution that a user can preserve his or her private context data at the client platform, and not fully leak them to the server.

## 3. The framework of TDAC

### 3.1. *The framework*

Fig. 1 shows the framework of TDAC. There are two main components in the framework: one is the PTRM at the client side, the other is the MRM at the server side. At the client side, there are general applications and trusted computing technologies besides the PTRM. The PTRM needs trusted computing technologies to ensure its implementation can be trusted. At the server side, there are sensitive objects besides the MRM.

As a critical credential between the PTRM and the MRM, we define the TACC as follows:

**Definition 1.** TACC (temporary access control credential): temporary access control is signed by the PTRM at the client side and held by an application. Then the credential is used when the application wants to access a sensitive object at the server side. The MRM will evaluate a request of the application according to the validation of the credential. We formally define the credential as follows: $TACC = \{SubRef, ObjRef, Action, TimeAfter, TimeBefore, H_{Capabilities}\}_{SK_{PTRM}}$

In Definition 1, *SubRef* refers to a reference of subject who logs in an application. *ObjRef* refers to a reference of object, which may be a service at the server side. *Action* refers to an action identity, which shows what kind of operations that the subject with the *SubRef* can perform on the object with the *ObjRef*. *TimeAfter*, *TimeBefore* refer to the life time of the credential. $H_{Capabilities}$ refers to the hash value (e.g. SHA-1 value) of current valid capabilities of the subject. The capabilities are an important implementation of access control mechanism. They are similar to ACLs (access control lists), a popular variation of access control mechanism to express access control policies. The difference lies in that capabilities are generated according to the subject while ACLs are generated according to the object.

We also define access control request as follows:

**Definition 2.** ACRequest (access control request): Access control request is used for an application with a TACC to request a sensitive object at the server side. We formally define it as follows: $ACRequest = \{SubRef, ObjRef, Action, TACC\}$

The descriptions of *SubRef*, *ObjRef*, *Action* are the same as those in Definition 1, and *TACC* is already defined in Definition 1.

As is shown in Fig. 2, the PTRM in TDAC is isolated from commodity operating system space and trustworthily runs at the client side through the assurance of trusted execution technologies.

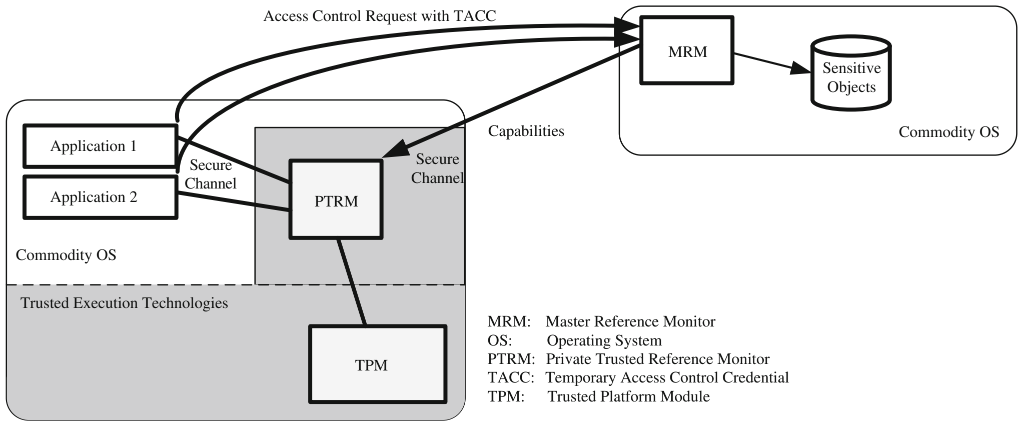
Access Control Request with TACC
MRM
Sensitive Objects
Commodity OS
Capabilities
Application 1
Application 2
Secure Channel
Secure Channel
PTRM
Commodity OS
Trusted Execution Technologies
TPM
MRM: Master Reference Monitor
OS: Operating System
PTRM: Private Trusted Reference Monitor
TACC: Temporary Access Control Credential
TPM: Trusted Platform Module

**Fig. 1.** The framework of TDAC implementation. Note: the components in the gray area are trusted components, and the dotted line separates the existed trusted components and the designed component (PTRM) in TDAC.

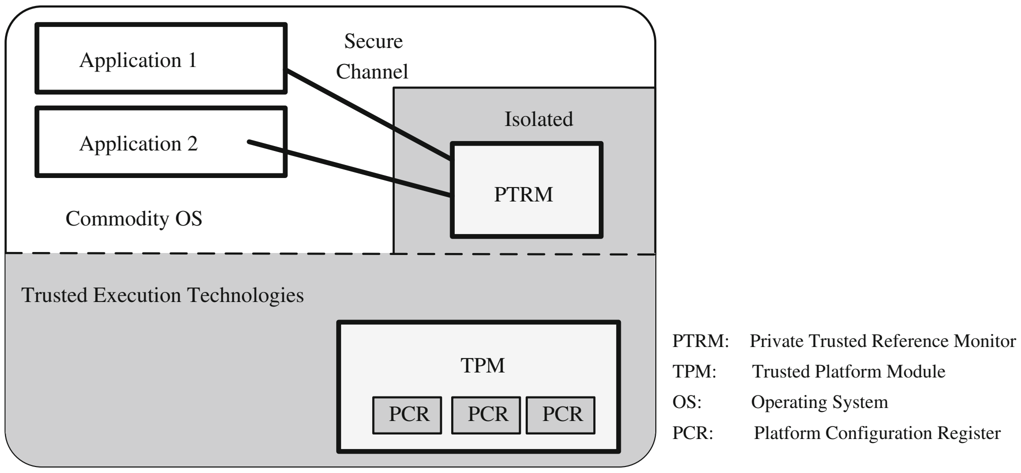
Application 1
Application 2
Secure Channel
Isolated
PTRM
Commodity OS
Trusted Execution Technologies
TPM
PCR
PCR
PCR
PTRM: Private Trusted Reference Monitor
TPM: Trusted Platform Module
OS: Operating System
PCR: Platform Configuration Register

**Fig. 2.** Proposed client implementation of TDAC.

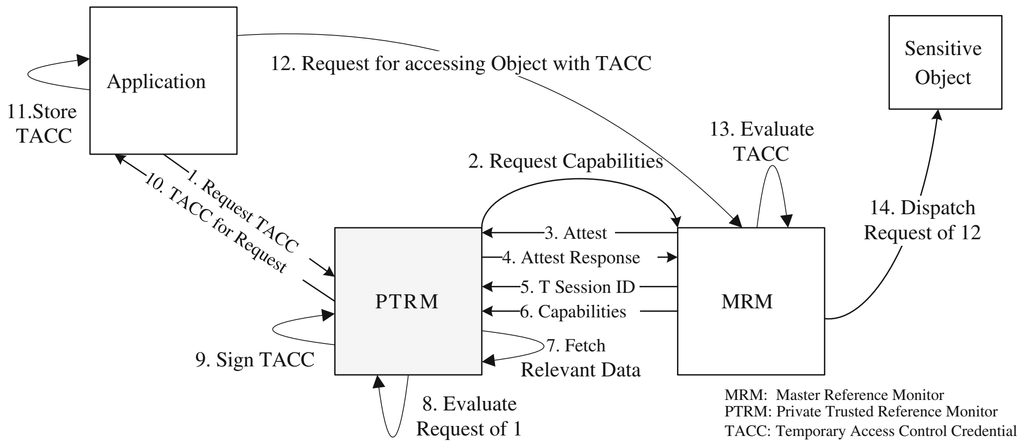

**Fig. 3.** Main workflow in TDAC.

## 3.2. *The main workflow*

Fig. 3 shows the main workflow in the framework of TDAC. The workflow can be divided into two phases: *TACC-sign phase* where an application requests a TACC from the PTRM; *Access phase* where the application requests a sensitive object from a server.

*TACC-sign phase*: Before an application at the client side wants to access a sensitive object, it must get a valid TACC. The PTRM takes charge of signing the TACC at the client side. After the PTRM receives the TACC-sign request which includes the information of who (subject) wants to access the resource (object) in what manner (action), the PTRM first requests capabilities of the subject in the TACC-sign request from the MRM at the server side (step 2). Then, the MRM attests the PTRM at the client side (steps 3–5). The aim of these steps is to build a trust session between the PTRM and the MRM. By attestation, the MRM may trust the PTRM. After attestation, the PTRM and the MRM will negotiate a *TSessionID* to identify the trusted session between them.

*TSessionID* is a random and temporary number. And it is used to reduce the frequency of attestation in trusted computing. If the PTRM has a valid *TSessionID*, the PTRM need not take the attestation process (steps 3–5) repeatedly. Because access control is a fine-granularity mechanism to protect sensitive objects, the application at the client side will frequently request a TACC, and frequently trigger attestation if there is no *TSessionID*. However, attestation is a time-consuming step in trusted computing, because the TPM is usually involved in the attestation, and its computing ability is very limited. According to our testing, it needs over 1000 ms for each attestation. Therefore, it is very important for the framework to use *TSessionID* to reduce the frequency of attestation.

After the PTRM and the MRM negotiate *TSessionID*, the MRM generates and signs capabilities according to the subject who launches the TACC-sign request in step 1, and sends them back to the PTRM (step 6).

From step 7 to step 8, the PTRM evaluates the TACC-sign request of step 1. If these policies need other relevant data in the process of evaluation, the PTRM will invoke some Broker modules to gather these data. These relevant data may come from a trusted remote service such as a general server supported by PKI (public key infrastructure), or a remote trusted physical sensor such as a temperature sensor.

After the PTRM evaluates the request of step 1, it will sign the TACC for the application if the result of evaluation is true (passed).

**Access phase**: After the application gets the TACC, it can access the sensitive object at the server side (steps 11–14).

The application sends a ACRequest (refer to Definition 2) to the server. After the MRM receives the ACRequest, the MRM will first evaluate the TACC. If the TACC meets the following requirements, it is valid:

- The TACC is integral. That is, The TACC is not changed during transmission. It can be evaluated according to digital signature signed by the private key of the PTRM.

- The lifetime in the TACC is valid. It can be evaluated by TimeAfter and TimeBefore in the TACC.
- The TACC is fresh. It means that the hash value of capabilities in the TACC is in the list of ValidCapabilitiesHashValueList, which is stored in the MRM. If a system security officer modifies security policies of a subject at the server side, ValidCapabilitieshashValueList will remove the hash value of invalid (old) capabilities of the subject, and insert the hash value of valid (new) capabilities of the subject.

If the TACC is valid, the MRM will dispatch the request to access the sensitive object (step 14). However, if the TACC is invalid, the MRM will return an error code, and the application may request a new TACC from the PTRM or abort the access.

In the whole workflow, the steps in **TACC-sign phase** will be executed when an application needs a new TACC. And the **Access phase** will be executed when the application needs to access the sensitive object. If ACRequest is duplicated in a short time, the application can reuse the TACC stored at the client side in step 11. As a result, the TACC can reduce the frequency to execute the steps of **TACC-sign phase**.

### 3.3. *Security analysis*

To implement TDAC needs to be resolved the following important security issues:

- The server must trust the PTRM. That is, no attacker can modify the process of evaluation and TACC-sign in the PTRM.
- The server must trust the TACC signed by the PTRM at the client side.

The first issue can be resolved by trusted computing technologies, such as TPM, TXT or SVM, trusted virtual machine monitor as is described in Section 2.1.

The second issue is the core issue in TDAC. First the TPM at the client platform has a private signed key. The PTRM will consign the TPM to signs the TACC with the private sign key at the client side. If the MRM trusts the public key of sign key, packed by an independent CA (certificate authority), the MRM will believe that the TACC is trusted against TACC modification.

Second the MRM also need verify whether $H_{Capabilities}$ is in the valid hash value list of capabilities. If $H_{Capabilities}$ is in the list, the MRM will believe that the TACC is fresh and trusted against capabilities modification at the client side. The verification implies that the capabilities stored at the client side need no protection from trusted computing technologies.

We can resolve the second issue by formally proving the following proposition using BAN logic (Burrows et al., 1989) and get the trust assumptions. The proof is attached in Appendix B. Based on formal analysis, the secure TDAC implementation needs the following assumptions:

- We assume that the hardware layer of trusted computing is tamper resistant.
- We assume the client environment in this paper is homogeneous. That is, each client platform is equipped uniformly with necessary trusted computing hardware. It is not necessary for the server to be equipped with trusted computing hardware in TDAC.
- We assume the PTRM is tamper resistant with the support of trusted computing technologies.

Compared with the assumption of the framework proposed by Sandhu and Zhang (2005), the above assumptions of our framework are fewer, because Sandhu et al. assumes that trusted reference monitor in their architecture at any peer requires *sealed storage* which stores sensitive objects, and *trusted applications*, such as VoIP applications, are ensured by *protected runtime environment* in *their Fig.* 1, but TDAC does not require the above two trusted components.

## 4. Performance evaluation

### 4.1. *Prototype description*

As the operations of the PTRM in TDAC depend on a series of TPM operations, which include many asymmetric encryption/decryption operations and some of these operations are implemented in the TPM which is a chip with low computing ability, the performance of the framework should be considered.

Performance study with our implemented prototype system is conducted in a 100 M switch-based LAN.

- The configuration of the server is HOST: DELL Precision 380 (P4 3.0 GHz/1 GB/160 GB); OS: RedHat Enterprise 4 with kernel 2.6.9; PolicyDB: MySQL 4.1.2; Cryptography Library: OpenSSL 0.9.7.
- The configuration of the client is: HOST: IBM R60 with Atmel TPM (TCG Specification 1.2); OS: Ubuntu with Linux kernel 2.6.19; Cryptography Library: OpenSSL 0.9.7; TSS (TCG, 2005): TrouSerS 2.8.0. The sign key size and attestation key size of TPM are 1024 bit and 2048 bit respectively.

To study the impact of sign key size, we also test the cases with sign key size of 2048 bit. We design four scenarios with different users and access control policies as is presented in Table 1. In the access control policy set, every user is randomly assigned 10% roles in the role set. That is, the capabilities of every user will consist of 10% access control policies.

### 4.2. *Evaluation results*

The test results are shown in Figs. 4 and 5.

From the test results, we can conclude that:

- TPM operations (attestation and sign TACC) cost a lot of computing time. Therefore, to reduce TPM operations will improve the performance of TDAC implementation, such as using *TSessionID* to reduce the frequency of attestation. Furthermore, because of low computing ability of TPM, it is unsuitable to deploy a similar module of the PTRM at the server side.
- The server evaluation operation (MRM evaluation) is very fast, and is not related to capabilities size, therefore the server side is very scalable to implement TDAC. But the get capabilities operation (get capabilities) will consume much of server's CPU time and is related to the size of system's policies. Therefore reducing the frequency of get capabilities is also a correct

**Table 1**
Description of four scenarios.

| # | User num. | Role num. | Object num. |
|---|---|---|---|
| 1 | 20 | 10 | 10 |
| 2 | 40 | 20 | 100 |
| 3 | 60 | 30 | 1000 |
| 4 | 80 | 40 | 10 000 |

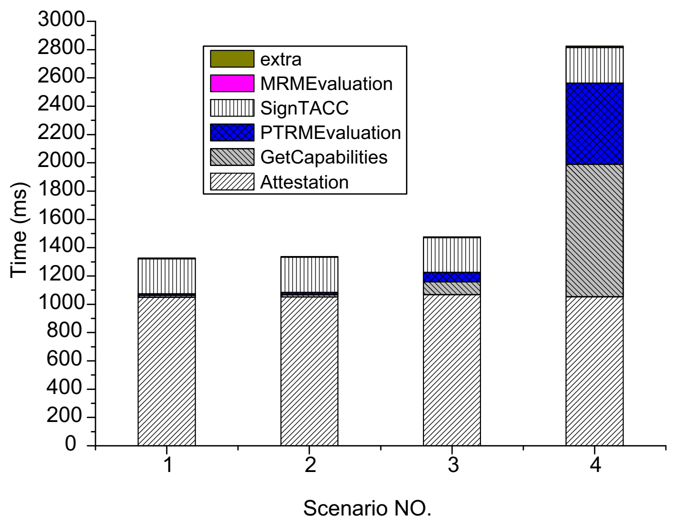

**Fig. 4.** Test result for performance evaluation with sign key size of 1024 unit (ms). Notes: Attestation is the performance consumption of step 3 to step 4 in Fig. 3; get capabilities is the performance consumption of step 6 in Fig. 3; PTRM evaluation is the performance consumption of step 8 in Fig. 3; sign TACC is the performance consumption of step 9 in Fig. 3; MRM evaluation is the performance consumption of step 13 in Fig. 3.

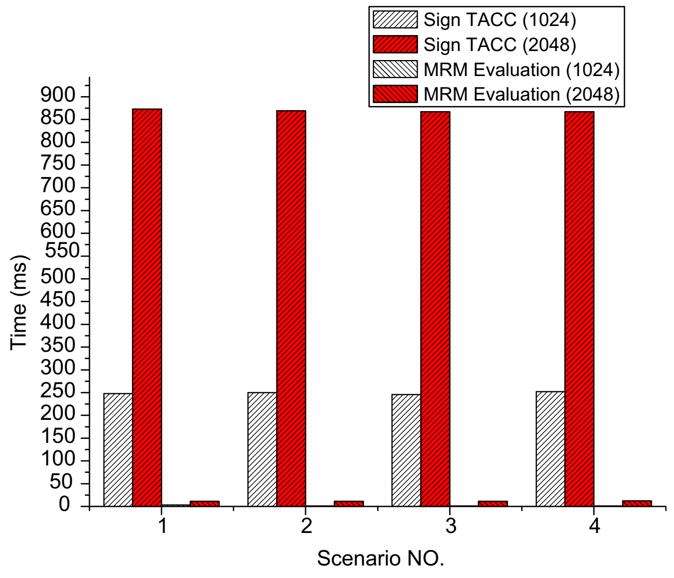

**Fig. 5.** Test result for performance evaluation with different sign key size units (ms).

direction to improve performance to efficiently implement TDAC.

- As is shown in Fig. 5, the sign key size seriously affects the performance of the operations of sign TACC and MRM evaluation. The time consumption of sign TACC will increase by 249% on average from the cases with 1024 bit to the cases with 2048 bit. Moreover, the time consumption of MRM evaluation will increase by 650% on average from the cases with 1024 bit to the cases with 2048 bit. Because the TACC is a temporary credential, the sign key size of 1024 bit is strong enough against the brute force attack. Thereby, we suggest a reasonable sign key size is 1024 bit.

## 5. Applicable scenarios analysis

TDAC can protect privacy of subject-context aware access control, because TDAC processes subject-context data at the client side. Subject-context aware access control is one of context aware access control mechanisms. In Subject-context aware access control, context data are created or stored at the client side. These context data are usually private data of subject or subject's platform. As a protection result, TDAC prevents the server from holding the subject-context data which are required in the access control evaluation.

We assume that there is a typical function-applicable scenario of subject-context aware access control, where there are two web sites: siteA and siteB. SiteA and siteB will provide signed visiting log for visitors. SiteA enforces an access control policy that *any user can visit pagea1 if he/she has visited siteB.*

A user, Jerry, who has visited siteB's pages, taken a signed log data from siteB as follows:

```
<?xml version=1.0?>
<histories>
  <entry>
    <date>2009-3-24</date>
    <location>w\rm\color\{none}{{\clap{w}}}w\rm\color\{none}{{\clap{w}}}w\rm\color\{none}{{\clap{w}}}.siteb.com</location>
    <page>/pageb1</page>
    <parameters>
      <param name=`` a'' >1</param>
      <param name=`` b'' >3</param>
    </parameters>
  </entry>
  ...
  <signuture>
    <signer>siteA</signer>
    <signdata>12323434abcdef....<signdata>
  </signuture>
</histories>
```

If Jerry wants to visit pagea1 at siteA, the PTRM at Jerry's computer can get the capacities of Jerry, and sign a TACC for Jerry if he satisfies the access control policy. As a result, Jerry can visit siteA and access pagea1 but not leak his private visiting log, which shows he used to visit pageb1 at siteB.

Another typical function-applicable scenario of subject-context aware access control is location privacy protection (Beresford and Stajano, 2003). Location information is very important to a roaming user. That is, the roaming user does not let an application server get as much location information as possible, when the user requests a location aware service at the application server. For example, when the location aware service only enforces a rough-granularity location aware policy, e.g. *a user can access a certain service if he or she is at Shanghai*, the user obviously avoids providing an accurate location, e.g. *Room 111, Software Building, Zhengheng Road, Shanghai.* TDAC, as a solution to the above problem, can protect the privacy of the accurate location information, because the PTRM evaluates the access request at the client side, and the accurate location information is processed at the client side, therefore will not be leaked to the application server side. As a result, the application server can provide a personalized location aware service, while the user can consume the service and protect his or her location privacy as much as possible.

## 6. Discussion

Protection of sensitive objects stored at the client platform is out of the scope of TDAC. What this paper discusses is the protection of the sensitive objects which may be services not data at the server side. Thus, our architecture may be applied in protecting sensitive services and data at the server side. Once the sensitive object is stored at the client platform, Sandhu's (Sandhu and Zhang, 2005) work could provide a feasible architecture to protect it. However, Sandhu's architecture needs more requirements for trusted components at the client platform.

In subject-context aware access control, the trust of context data should be assured by trusted services to implement TDAC for subject-context aware access control. For example, visiting log needs to be signed by a trusted web server and stored at the client side.

TDAC provides a feasible solution that TDAC will protect client's private context data from leaking them to a server during access control evaluation, but this paper does not discuss the defense of a *reason attack*. A *reason attack* can reason the subject's private data according to evaluation results. For example, if we use the case in Section 5, and siteA enforces an access control policy that *any user can visit pagea1 if he/she has visited pageb1 at siteB*, then, if Jerry successfully visited pagea1 at siteA, the master of siteA can know Jerry used to visit pageb1 at siteB. It could be a feasible solution to design a private data filter that analyzes the applicable access control policies which are provided by the PTRM after the sign of a TACC. The filter determines whether Jerry will visit pagea1 at siteA according to the policies analysis and Jerry's privacy protection preferences. The private data filter can co-run with the application in a commodity environment.

## 7. Conclusion and future work

TDAC is a novel access control mechanism with the support of trusted computing technologies but requires fewer trusted components compared with the previous work of Sandhu et al. This paper discusses the architecture of TDAC and the workflow the architecture; and evaluates performance of TDAC. Finally, we present applicable scenarios how to apply TDAC to protect subject's private data in subject-context aware access control.

In the future work, we will analyze and resolve the potential reason attacks which is mentioned in Section 6; and implement our framework in a cloud computing platform with support of TPM and TXT.

## Acknowledgments

This paper is partly supported by NSFC (Grant no. 90818015) and 211-Project Sponsorship Projects for Young Professors At Fudan. Thanks to Prof. Jianyin Zhou, Prof. Gang Pan for their comments. Thanks also to Ms. Min Li for English polish.

## Appendix A. Abbreviation list

| Abbreviation | Phrase |
|---|---|
| MRM | Master reference monitor |
| PTRM | Private trusted reference monitor |
| SVM | Secure virtual machine |
| TCG | Trusted computing group |
| TDAC | Trusted decentralized access control |
| TPM | Trusted platform module |
| TXT | Trusted eXecution Technology |

## Appendix B. The proof in Section 3.3

**Proposition.** *$MRM \models X$ where $\{X\}_{k^{-1}} = TACC$ where $X$ refers to the content of TACC, $k^{-1}$ refers to the private key of PTRM, and $MRM \models X$ refers that MRM believes X.*

**Proof. Step 1**: Apply Jurisdiction or control ($\Rightarrow$ −*elimination*) rule:

$$\frac{MRM \models PTRM \Rightarrow X, MRM \models PTRM \models X}{MRM \models X}$$

(The equation refers that if *MRM* believes that *PTRM* is an authority of *X* and *MRM* believes *PTRM* believes *X*, then *MRM* believes *X*.)

Because of the support by trusted computing technologies, $MRM \models PTRM \Rightarrow TACC$ (Which refers that *MRM* believes that *PTRM* is an authority of *TACC*) is true. This proposition needs an assumption to assure the credible process of *TACC* in *PTRM*.

**Step 2**: Apply $\mid\!\sim$ − elimination rule:

$$\frac{MRM \models \#(X), MRM \models PTRM \mid\!\sim X}{MRM \models PTRM \models X}$$

(The equation refers that if *MRM* believes *X* is fresh, and *MRM* believes that PTRM once said *X*, then *MRM* believes that *PTRM* believes *X*.)

Because *MRM* will check whether $H_{Capablities}$ is in the list of ValidCapabilitiesHashValueList, *MRM* can judge whether *X* is fresh. Therefore, $MRM \models \#(X)$ is true.

**Step 3**: Apply $\mid\!\sim$ − introduction rule:

$$\frac{MRM \models \overset{k}{\mapsto} PTRM, MRM \lhd \{X\}_{k^{-1}}}{MRM \models PTRM \mid\!\sim X}$$

(The equation refers that *MRM* believes that *PTRM* has *k* as a public key and MRM sees the *X* which signed by the private key of *k*, then MRM believes that PTRM once said *X*.)

Because *MRM* can verify a public key certificate of *PTRM*, $MRM \models \overset{k}{\mapsto} PTRM$ is true; $MRM \lhd \{X\}_{k^{-1}}$ is obviously true. Therefore, $MRM \models X$ is true. □

## References

AMD. AMD virtualization 〈http://www.amd.com/us-en/0,3715_15781,00.html?redir=SWOP08〉; 2008.

Beresford AR, Stajano F. Location privacy in pervasive computing. Pervasive Computing 2003;1(1):46–55.

Blauton M, Atallah MJ. Provable bounds for portable and flexible privacy-preserving access rights. In: Proceedings of the 10th ACM symposium on access control models and technologies (SACMAT '05), Stockholm, Sweden, June 1–3, 2005. New York, NY: ACM Press; 2005. p. 95–101.

Burrows M, Abadi M, Needham R. A logic of authentication. ACM Operation Systems Review 1989;23(5):1–13.

Cautis B. Distributed access control: a privacy-conscious approach. In: Proceedings of the 12th ACM symposium on access control models and technologies (SACMAT'07); 2007. p. 61–70.

Chen X, Garfinkel T, Lewis EC. Overshadow: a virtualization-based approach to retrofitting protection in commodity operating systems. In: Proceedings of the 13th international conference on architectural support for programming languages and operating systems (ASPLOS 2008), Seattle, WA, USA; 2008. p. 2–13.

Garfinkel T, Pfaff B, Chow J, et al. Terra: a virtual machine-based platform for trusted computing. In: Proceedings of the 19th ACM symposium on operating systems principles; 2003. p. 193–206.

Gu T, Pung HK, Zhang DQ. A service-oriented middleware for building context-aware services. Journal of Network and Computer Applications 2005;28(1): 1–18.

Han W, Zhang J, Yao X. Context-sensitive access control model and implementation. In: Proceeding of the 5th international conference on computer and information technology, Shanghai, China; 2005. p. 101–5.

Han W, Shi X, Chen R. Process-context aware matchmaking for web service composition. Journal of Network and Computer Applications 2008;31(4): 559–76.

Hulsebosch RJ, Salden AH, Bargh MS, et al. Context sensitive access control. In: Proceedings of the 10th ACM symposium on access control models and technologies (SACMAT '05), Stockholm, Sweden, June 01–03, 2005. New York, NY: ACM Press; 2005. p. 111–9.

Intel. Intel trusted execution technology 〈http://www.intel.com/technology/security〉; 2008.

Kane K, Browne JC. On classifying access control implementations for distributed system. In: Proceedings of the 9th ACM symposium on access control models and technologies (SACMAT '06), Lake Tahoe, California, USA; June 7–9, 2006. p. 29–38.

Kapsalis V, Hadellis L, Karelis D, et al. A dynamic context-aware access control architecture for e-service. Elsevier Transactions on Computers & Security 2006;25(7):507–21.

Li J, Li N, Winsborough WH. Automated trust negotiation using cryptographic credentials. In: Proceedings of the 12th ACM conference on computer and communications security (CCS '05), Alexandria, VA, USA, November 7–11, 2005. New York, NY: ACM Press; 2005. p. 46–57.

Mao W, Yan F, Chen C. Daonity—grid security with behaviour conformity from trusted computing. In: Proceedings of the first ACM workshop on scalable trusted computing (STC'06), Alexandria, Virginia, USA; November 3, 2006. p. 43–6.

McCune JM, Parno B, Perrig A, et al. Minimal TCB code execution. In: 2007 IEEE symposium on security and privacy (SP'07); 2007. p. 267–72.

McDaniel P. On context in authorization policy. In: Proceedings of the 8th ACM symposium on access control models and technologies (SACMAT '03), Como, Italy; June 2–3, 2003. p. 80–9.

OASIS. Authentication context for the OASIS security assertion markup language (SAML) V2.0; January 2005.

Sandhu R, Coyne E, Feinstein H, et al. Role-based access control models. IEEE Computer 1996;29(2):38–47.

Sandhu R, Zhang X. Peer-to-peer access control architecture using trusted computing technology. In: Proceedings of the 10th ACM symposium on access control models and technologies (SACMAT '05), Stockholm, Sweden, June 1–3. New York, NY: ACM Press; 2005. p. 147–58.

Sandhu R, Zhang X, Ranganathanand K, Covington MJ. Client-side access control enforcement using trusted computing and PEI models. Journal of High Speed Networks 2006;15:229–45.

Squicciarini AC, Hintoglu A, Bertino E, et al. A privacy preserving assertion based policy language for federation systems. In: Proceedings of the 12th ACM symposium on access control models and technologies (SACMAT'07); 2007. p. 51–60.

Strembeck M, Neumann G. An integrated approach to engineer and enforce context constraints in RBAC. ACM Transactions on Information and System Security 2004;7(3):392–427.

TCG. TCG specification architecture overview 〈https://www.trustedcomputinggroup.org/〉; 2009.

TCG. TCG software stack (TSS) specification 〈https://www.trustedcomputinggroup.org/groups/software〉; 2005.

Tolone W, Ahu GJ, Pai T, et al. Access control in collaborative systems. ACM Computing Surveys 2005;37(1):29–41.

Wedde HF, Lischka M. Role-based access control in ambient and remote space. In: Proceedings of the 9th ACM symposium on access control models and technologies (SACMAT '04), Yorktown Heights, New York, USA; June 2–4, 2004. p. 21–30.

W3C. Platform for privacy preferences (P3P) project 〈http://www.w3.org/P3P/〉, 2009.

Yang J, Shin KG. Using hypervisor to provide data secrecy for user applications on a per-page basis. In: Proceedings of the ACM/Usenix international conference on virtual execution environment, Seattle, WA, USA; 2008. p. 71–80.

Zhang X, Nakae M, Covington M, et al. A usage-based authorization framework for collaborative computing systems. In: Proceedings of the 9th ACM symposium on access control models and technologies (SACMAT '06), Lake Tahoe, California, USA; June 7–9, 2006. p. 180–9.