Trustworthy Mutual Attestation Protocol for Local True Single Sign-On System: Proof of Concept and Performance Evaluation

Zubair Ahmad Khattak[1, 2], Jamalul-lail Ab Manan[2] and Suziah Sulaiman[1]
[1] Department of Computer and Information Sciences, Universiti Teknologi PETRONAS
Bandar Seri Iskandar, Tronoh, Perak 31750 - MY
[2] Advanced Analysis and Modeling Cluster, MIMOS Berhad
Technology Park Malaysia, Kuala Lumpur 57000 - MY
[e-mail: zubair_g00953@utp.edu.my, suziah@petronas.com.my, jamalul.lail@mimos.my]
*Corresponding author: Zubair A. Khattak

Received December 3, 2011; revised July 6, 2012; accepted June 16, 2012; published September 26, 2012

Abstract

In a traditional Single Sign-On (SSO) scheme, the user and the Service Providers (SPs) have given their trust to the Identity Provider (IdP) or Authentication Service Provider (ASP) for the authentication and correct assertion. However, we still need a better solution for the local/ native true SSO to gain user confidence, whereby the trusted entity must play the role of the ASP between distinct SPs. This technical gap has been filled by Trusted Computing (TC), where the remote attestation approach introduced by the Trusted Computing Group (TCG) is to attest whether the remote platform integrity is indeed trusted or not. In this paper, we demonstrate a Trustworthy Mutual Attestation (TMutualA) protocol as a proof of concept implementation for a local true SSO using the Integrity Measurement Architecture (IMA) with the Trusted Platform Module (TPM). In our proposed protocol, firstly, the user and SP platform integrity are checked (i.e., hardware and software integrity state verification) before allowing access to a protected resource sited at the SP and releasing a user authentication token to the SP. We evaluated the performance of the proposed TMutualA protocol, in particular, the client and server attestation time and the round trip of the mutual attestation time.

Keywords: Remote attestation, mutual attestation, local true SSO, TC, TPM, IMA

Parts of this paper have been presented in the International Conference on Security and Management (SAM'11), Las Vegas, Nevada, USA. This version includes an updated abstract, experiment setup, trustworthy mutual attestation protocol implementation, performance analysis, Security, Trust and Privacy (STP) aspects comparison, advantages of proposed work and discussion. This research has been sponsored by the Universiti Teknologi PETRONAS Postgraduate Assistantship scheme and MIMOS Berhad, Malaysia.
http://dx.doi.org/10.3837/tiis.2012.09.025

1. Introduction

The Internet, on one hand, brings numerous advantages such as access to distributed services, the flexibility of using multiple services and the freedom to access these services from anywhere and at anytime. On the other hand, the problem of managing many credentials for accessing every resource or service for which the users are registered raises several security concerns [1]. In addition to the security concerns, trust is another important issue in an open environment and a lack of trust leads to security concerns. In online resource access scenarios such as confidential document access, a user's concerns over an enterprises' security and platforms' trustworthiness are increased throughout the world. The conventional computer platforms lack the much needed integrity checking of the platforms. In a typical real case scenario, if an enterprise's system, either a client or a server, is found running a malicious software or rootkit, then it is considered a compromised [2] system and reduces the users' confidence in the intended SPs.

To overcome the aforementioned issues, new mechanisms are needed to reduce the lack of confidence related to security and trust in an open environment. Some reasonable approaches to diminish such security implications have been introduced, such as SSO systems [3] and an in-depth analysis of SSO taxonomy in [4] identified four main SSO categories, namely:

- Local pseudo SSO
- Proxy-based pseudo SSO
- Local true SSO
- Proxy-based true SSO

The majority of existing SSO schemes consist of a third party called ASP [4][5] or IdP [7]. Microsoft [5], Liberty Alliance [6] and Shibboleth [7] are examples of third party based SSO schemes. In all third party schemes, the user and SP put their trust in the ASP or IdP for trustworthy user authentication assertion. In the local true SSO, a user will not trust an entity that is under external control. The trusted component TPM within the user's system takes the role of the authentication server [1]. In this paper, we focus only on the local true SSO and the other types of SSO schemes are out of scope.

In a number of areas such as psychology, economics, sociology, political science, and computer science, trust always plays an important role but many different definitions exist. Here in, the authors limit their scope to the machine platform trust established by utilizing the TPM [9] functionalities. The TCG [8] trust refers to "a device that behaves as expected for a specific purpose and always in a particular way" [10]. Furthermore, the terms "specific purpose" and "particular way" were elucidated by Alam et al. [27] as follows: "particular way" is linked to the enquiry as to how a chore is likely to be carried out; and "specific purpose" refers to a precise chore such as the object utilization, net resource (service) access, or some computational activity. The authors in this work adopted the TCG trust definition as stated above. In our approach, an interacting platform maintains a Database (DB) of good hashes. Good hashes are actually trusted states of all the executables, libraries and applications of the interacting platforms. Good hashes of a platform means that the platform is trusted and bad hashes (compromised by virus, Trojan horse etc.,) means otherwise.

The TCG [8], remote attestation mechanism, is designed to measure and report the integrity of the computer platforms. The TPM [9] works as a tamper-resistant microprocessor against software based attacks. Each TPM has an Endorsement Key (EK) which is a manufacturer built-in key and uniquely identifies a particular platform in a similar way as an Internet

Protocol (IP) address that specifies a particular client on the Internet. So the owner of a platform creates an Attestation Identity Key (AIK), which is a pseudonym key and is certified by a PrivacyCA.

The main contributions in this paper are as follows:

- We extended the traditional remote attestation protocol and proposed a mutual attestation protocol.
- We incorporated the proposed mutual attestation protocol in a novel trustworthy local true SSO architecture. Using this mutual attestation protocol, the user system (client) and SP (server) carry out each other's platform attestation. So, we configured a client and server with an IMA and used a TPM to protect the integrity of the in-kernel Measurement List (ML).
- For the purpose of this paper and proof of concept, we have used an eXtensible Markup Language (XML) [21] for sending and receiving the attestation request between the client and the server. Alternatively, we can make use of a Security Assertion Markup Language (SAML), an XML-based open standard for exchanging authentication and authorization data between security domains in a particular way, [11] to encode the attestation request and response.

In this paper, our scope is limited to the following:

- To construct a *TMutualA* protocol proof of concept for the local true SSO scheme and to figure out how our proposed *TMutualA* protocol will react if a threat is detected at both of the machines' platforms. To validate the trustworthiness of the communicating machines' platforms, a security test is to be carried out to check if the protocol can sense whether there is a threat or not. Threats to the interaction link are out of the scope of this paper.

The remainder of the paper is ordered as follows. Section 2 presents the background which includes the local true SSO, TC, and remote and mutual attestation. Section 3 discusses the problem description. Section 4 presents the related works. Section 5 describes the proposed *TMutualA* protocol architecture. Section 6 presents the result analysis, advantages of the proposed scheme and discussion. We conclude the paper with the future works in section 7.

2. Background

2.1 Local True Single Sign-On

Pashalidis et al. [1] discussed, in detail, how TC technology can be used in SSO schemes which are based on their two key observations [12]: (i) user authentication can be delegated to the Trusted Platform (TP) and (ii) an Identity (ID) credential (which is actually a X.509 public key certificate carrying a unique serial number) is assigned by the PrivacyCA corresponding to the TPM identity.

2.2 Trusted Computing

In the early 2000s, the Trusted Computing Platform Alliance (TCPA) [13], which has been changed to TCG [8], launched the notion of TP. The basic idea of the TCG was to first bring trust into the traditional computing platforms through a TPM chip and secondly, it was designed in response to escalating security breaches [8][14]. TPM, by definition, is a cryptographic co-processor device that offers a range of collateral services, e.g., Random Number Generator (RNG), defended memory settings called Platform Configuration Registers (PCRs) and asymmetric key generation. The PCRs can store platform configurations

of varied entities in the form of cryptographic hashes using the Secure Hash Algorithm-1 (SHA-1) [15]. The entities may be BIOS, a kernel, an application or a Boot Loader.

According to the TCG specification [10], the TPM can only be controlled by two operations: (i) when the system is rebooted, which resets all PCR values and (ii) through a PCR-extended operation, for instance whenever a value is stocked in the PCR, first, its mess (i.e., hash) value is added onto the current PCR and secondly, the SHA-1 value of the subsequent structure is stocked in an identical PCR. The coupling of this technique with the SHA-1 irreversibility ensures that an infinite number of configurations can be stocked in a single PCR and is protected through a tamper-resistant facility. Later, the assembled measurements are conveyed to the challenger (either a client or server) to validate the attesting machine platform's honesty and the TPM's legitimacy. The AIK, a pseudonym key, is only accessible to the TPM and is used to sign the accumulated values for vouching trust in it. The signing with the AIK actually provides assurance and guarantees that the accumulated value is really signed by a genuine TPM. However, is important to note that the AIK private part is never released outside the TPM because of platform privacy concerns.

2.3 Remote Attestation

Remote attestation is a technique through which the attesting machine validates its platforms' (i.e., hardware and software) integrity states to a remote machine known as a challenger. The main objective of this technique is to let the remote machine ascertain the degree of trust in a target machine on the basis of its platform integrity health status. In the remote attestation scheme, the challenger is an entity which challenges the target machine for its platform authenticity. The protocol which is used for this purpose is called the platform integrity challenge-response protocol. In a simple remote attestation process, the challenger (e.g., a server) machine challenges the target (e.g., a client) machine. The target machine, on receiving the attestation request, collects the requested component's integrity and returns it to the challenger as an attestation response. The challenger machine upon receiving it then performs the validation process to confirm whether the target machine's platform integrity is trustworthy or not.

The TCG initiates a remote attestation technique based on the notion of a load-time measurement. Although, the TCG's remote attestation protocol is deficient in gauging the software loaded after the BIOS, BOOT loader etc. Therefore, to fetch trust to the level of the OS and applications, Sailer et al. [2] provided an IMA approach. The IMA exploits the boot-time measurement technique to validate the integrity of a target machine's platform. The IMA was the initial technique to widen the TCG technique to the OS level by gauging all the executables and libraries loaded after the booting of the OS. The application hashes are recorded at boot-time and a Stored Measurement Log (SML) is upheld in the Linux file system (i.e., securityfs). In integrity dimension operations, the target machine's PCR and SML values are conveyed to the challenger. The challenger machine then analyzes the loaded application at the host and makes a decision as to whether the communicating platforms are trustworthy or not.

2.4 Mutual Attestation

The problem with the remote attestation scheme is that the target and challenger machines cannot establish mutual trust amongst themselves, i.e., in proving their mutual platform integrity. The mutual attestation or bi-directional remote attestation scheme by using the mutual integrity challenge-response protocol enables both communicating machines to verify

to each other that their platform stacks are trusted as anticipated and have not been meddled with.

3. Problem Description

The security concerns in the local true SSO are [1]: (1) Absence of mutual trust establishment and (2) Complexity. The absence of mutual trust leads to the security concerns as follows:

3.1 Dishonest Client Machine

This refers to the case where the web server machine does not have any knowledge about the client's machine integrity state, i.e., whether it is in a trusted state or not. If a user wishes to access a resource sited at the web server (SP) it could be a compromised client machine which may have been the consequence of a theft of the user's credentials.

3.2 Dishonest Server (Service Provider) Machine

This refers to the case when the web server's machine is compromised by a malicious means (i.e., rootkit). As a consequence, the server is a dishonest (i.e., under the control of an invader) web server machine which can access sensitive resources (services) or users' credentials.

4. Related Work

There are numerous works related to the TPM and IMA based attestations. However, we discuss only the most recent and related work to the work presented in this paper. The most related work to this paper is from Pashalidis and Mitchell [1] who described how the SSO among separated SPs can be achieved with a TP. However, the Pashalidis and Mitchell scheme has some limitations as discussed in section 3.

Remote attestation in computing, mobile platforms and running executables is an emerging and exciting research area. Garris et al. [16] presented the design and implementation of a trustworthy kiosk computing prototype. Their prototype was based on two protocols; the first protocol allows a personal handheld mobile device owner to establish trust on a public computer kiosk before revealing any personal information to the kiosk, and the second protocol allows a kiosk owner to verify that the kiosk is running approved software. Their implementation used a new instruction added recently to the x86 architecture. However, their approach did not provide a run-time measurement such as tracking a software's state while it is running.

Sadeghi and Stüble [17] proposed the idea of a Property Based Attestation (PBA) where platform or application configuration can be mapped into specific properties. Although their approach could be a bit arduous in a sense when it comes to varied configuration discovery and mapping them into specific properties, their approach can help to overcome platform configuration discrimination issues that exist in traditional TCG specification remote attestation mechanisms. These properties will not allow a remote verifier to find out what configuration of the running software needs to be verified.

Sailer et al. proposed the IMA [2] as an extension of the TCG's trust concept. Their proposed scheme allows the TPM to measure not only the system static states but also the dynamic states. In their approach, the IMA is implemented as a Linux security module; where upon loading, measures each executable, library or kernel module and stores it as evidence

(i.e., in the form of SHA-1 hash values, that will later be used for verification purposes) in the TPM and Linux log file.

Ali and Nauman [18] proposed a trust-aware web server architecture for enforcing access control policies. In their approach, access to protected resources is based on the client's integrity state which makes use of an IMA approach [2], Integrity Based Access Control (IBAC). The difference between Ali and Nauman's [18] work and the work presented in this paper is as follows. In their approach, a web server can assess the trustworthiness of the client (using remote attestation) but not vice versa, i.e., not a mutual attestation. Our approach is different in the sense that the client and server platforms mutually authenticate each other using the mutual attestation protocol. For instance, a server attests the AS including all executables residing on a client's side and vice versa. The mutual attestation is based on the integrity status of the platforms, before allowing any party (client or server) to access protected resources and releasing the user authentication token. In their implementation, attestation was performed on Linux fedora core systems, while our work is carried out on Linux Ubuntu core systems.

To remotely certify the behaviour of a policy model, Alam et al. [27] have proposed a model based approach. The Alam et al. work actually provides Usage Control (UCON) [29] as an example target policy model which combines the different remote attestation schemes' strengths.

Mutual attestation is a relatively new notion amongst researchers in SSO systems and a few works have utilized or discussed mutual attestation in the client server environments. Sailer et al. [31] described the mutual attestation for an open environment and highlighted how in the future, especially in areas such as Software as a Service (SaS), Cloud and Grid computing, the mutual trust establishment via machine platform integrity checking may play a vital role. Their work is based on the IMA mechanism which expands the trust chain notion into the machine run-time environment.

Shane et al. [32] described how to provide a shield against crimeware threats (e.g., rootkits, keystroke loggers, worms, Trojan horses and viruses) in open environments using TC technology. For example, in shielding credit card transactions before making an online transaction, the customer and merchant machine platforms must mutually attest their platforms' states to guarantee that both machines are in a trusted state.

Bringing TC based security and trust into a Grid environment, Zhan et al. [33] suggested the design of the trusted Grid Environment. The authors utilized a mutual attestation scheme to demonstrate how to construct a trustworthy sub-domain for a Grid environment using a TC based mutual attestation scheme. Cáceres et al. [34] presented the notion of trust overlays utilizing a TC based mutual attestation scheme between communicating mobile devices and its infrastructure.

Our main idea, in this paper, is to use mutual attestation for user access to multiple resources (services). Our proposed framework may be used either in a stand alone local domain or in a federated situation for access to foreign domains. In our framework, the *TMutualA* scheme, the server machine, before releasing any resource to its client, must ensure that the user authentication has as well as the mutual attestation process have succeeded.

In the local true SSO scheme, we are not particularly concerned about the scalability of the measurement, scripts and libraries because only a small number of platform configurations exist and there are very few software updates or patches needed in a particular organization's internal network which is normally performed centrally. We are more concerned about the

management of the good hash database in a geographically separated domain such as those in a federated environment.

The *TMutualA* protocol architecture and proof of concept details are given in the next section.

5. Trustworthy Mutual Attestation Protocol

In this section, we give details about the proposed *TMutualA* protocol such as proof of concept setup, proof of concept architecture and implementation for a local true SSO system.

In the proposed architecture, we assume a scenario whereby the SP has some protected resource while the client has an acquired authentication token and Authentication Service (AS) whose integrity must be verified before releasing the authentication assertion to the SP. As proof of concept, we show how the interacting platforms assure that they are in trustworthy states.

The work presented in this paper is new, because we are focusing more on the mutual attestation of platforms; this is because the traditional remote attestation scheme has limitations and cannot measure both of the interacting platforms' integrity. To accomplish a mutual attestation between a client and server, we created modules such as an integrity provider, attestation challenger-Corroboration Service (CS), validation, daemon-attestation presenter, and a DB which stores good hashes.

The integrity provider requests a *CS* at the server to send an attestation query to the client machine and the daemon-attestation presenter at the client which responds with a fetched SML and Quote over the 10th PCR and nonce. We selected the 10th PCR because the IMA uses the 10th PCR by default. However, the user can use *+config IMA_MEASURE_PCR_IDX* to determine the TPM PCR register index that the IMA uses to hold the integrity aggregate of the ML. For instance, +int "PCR for Aggregate (8<-Index<=14)" means using the TPM PCR register between index 8-14. The validation module, part of a *CS*, performs the validation process to validate the received nonce freshness, PCR composite structure and SML list. The received SML comparison is performed against the SML entries in the DB which contains good hashes (un-tampered SML values, hashed with SHA-1).

5.1 Proof of Concept Setup

Table 1, given below, shows the technological details used in our experiment. The proof of concept was implemented using java because the IMA supports java. Furthermore, we extended the Linux Ubuntu system kernel 2.6.35.11 with the IMA [2] to establish a trust chain up to the OS and beyond it to the running applications.

Table 1. Experiment setup

Client	Server
Ubuntu	
RAM 1GB	RAM2.5GB
2.00GHz Genuine Intel (R) CPU	Dual 1.83GHz Intel Core (TM) 2 CPU
Apache web server	
MySQL DB (contains good hashes of executables)	
jTSS (used for communication with the TPM) Used to carry-out the quote process over the PCR and the nonce, and extract this value for us to report to the challenging party.	
jTpmTools (we use jTSS to communicate with the TPM) Create the AIK using the *AIK_TPM* key *jTpmTools* and *jTSS libraries*.	
TPM 1.2 (Complies with the TCG TPM specification v1.2 [20])	
Kernel version 2.6.35.11 (Kernel configured to work with the IMA [2])	

5.2 Proof of Concept Architecture

In the proof of concept, for simplicity, we will use a client to represent a user system and a server to represent an SP, respectively. We construct an integrity provider daemon on both the client and server. This service issues an integrity request to the *CS* that generates and sends a nonce to the target platform, and the *Daemon Attestation Validation at the Client (VD-ClientA)* and the *Daemon Attestation Validation at the Server (VD-ServerA)* are responsible for validating the received SML, PCR and Nonce. In addition, on the client and server, an integrity reporting daemon was developed which consists of an attestation presenter that conveys the integrity information to the server or client as a response to the request they received from the integrity provider. The *Daemon Attestation Presenter at the Client (PD-ClientA)* and the *Daemon Attestation Presenter at the Server (PD-ServerA)* listen to the attestation requests to extract the PCR from the TPM and SML from the securityfs. The attestation request and response between the client and server are achieved via an XML. So, the SML, PCR and Nonce verification process is initiated in the XML format that can benefit not only communication between internal systems but also external systems such as vendors, customers and partners. However, it is important to note that only an SAML can be used in a Shibboleth environment or in a Web service [18] environment, which means that it cannot be used in a simple mutual attestation setup. We can use the SAML, **Fig. 1**, in Web service environment to encode attestation request and response messages from both the client and the server.

```
<samlp:Request
MajorVersion="1" MinorVersion="0"
RequestID="nonce" >
<samlp:AttributeQuery>
<saml:Subject />
<saml:AttributeDesignator AttributeName="urn:utp:edu:ClientORServerPlatformIntegrityStatus" />
</samlp:AttributeQuery>
</samlp:Request>
```

Fig. 1. Client or server integrity status attestation request in web services environment

For this paper, we have used the XML for sending and receiving attestation requests. For this, we created two XML files. The examples for the attestation request (samplerequest.xml) and the attestation response (responseSkel.xml) are given in **Fig. 2** and **Fig. 3**. The explanations of the attestation request and attestation response are given in **Fig. 4**.

```
<?xml version="1.0" encoding="UTF-8"?>
<wss:Security>
<SignChallenge>
<wst:Challenge>7EE52B76A564C1E5A4B7EBCDBC9D8C723651DD9C</wst:Challenge>
</SignChallenge>
</wss:Security>
```

Fig. 2. XML attestation request

```
org.xml.sax.InputSource@1c9a690
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<wst:RequestedSecurityTokenResponse>
<wst:RequestedSecurityToken>
<Challenge>7EE52B76A564C1E5A4B7EBCDBC9D8C723651DD9C</Challenge>
<pcrs>
<num>10</num>
<ValStructure10>5AEDD6085C9260648111AC18B4E12C633030DB0515A12CC5B4B91B9B8F176E204AAA4227421FF92C6D9ADAACDF16C036212F
F627F7E3D807BB217B7E179C4AADDB066F1F719AD949A9D5FF31B60206EEA08E38E5B3C95F14E77881C5907DD11AFF84028257BFCD688E2991
6CF2CE95F889632A0A322ECF9BDE9CCDDB3FED96B2A268B3EC4F5B022209820FBC896937F2C70565A707BFAF11038A5E8C8FB72F84AA169C7
DAFD6B01EBD6B43545093EFA9B530693920D5EAFAB41DF9ADFD301DDCC94033E1325503E178028149A0ACA5401AC25A9B61CDD8ACD5F4F8D
91168702A2FCEEC9B19362719A14E8A87820A1604C00106E5ABE8A532D5E9A86A78CCDDC7
</ValStructure10>
<Data10> 0101000051554F54F4816A84EC09E3E16116E889A5BD2633A3186DCA7EE52B76A564C1E5A4B7EBCDBC9D8C723651DD9C </Data10>
</pcrs>
<smlcontents>
10 9fdd422f8b402d3111e410de7f2c08a55ca7d3d8 ima c3a0a5d8e10b69abb3f42bdcc827d3b7822ea44f boot_aggregate
10 16020d495113eb400a4cb12bdc2a340b9f9c4461 ima 5fb8c8479f31574c435aa06d5bae6ae5a737ca81 sh
.
10 fa3788f31e6751e8c14050b601520631aade364b ima cce383d096e7ef6eed5dc4b377d0d735ff124f12 cksum
</smlcontents>
</wst:RequestedSecurityToken>
</wst:RequestedSecurityTokenResponse>
```


Fig. 3. XML attestation response

The overall mutual platform attestation protocol architecture depicted in **Fig. 4** is as follows: The client requests for a protected resource located at the server (step 1). After receiving the request and realizing that the target is restricted, the server then passes control to the integrity provider module. The integrity provider at the server side makes an attestation request to the client (step 2) via the *CS* which consists of a sign-challenge, for instance, a nonce (step 3). The *PD-ClientA* on the client continuously listens for the attestation request and after receiving the nonce, the attestation presenter daemon assembles the client platform integrity information.

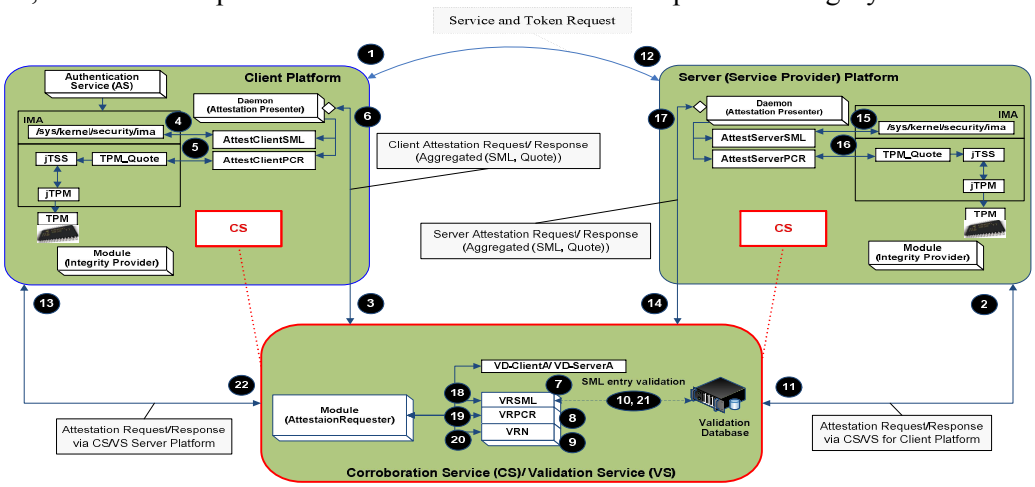

Fig. 4. Trustworthy mutual attestation protocol architecture for the True SSO system

In (step 4), the SML is collected from the */sys/kernel/security*. In (step 5), the acquired nonce is passed on along with a query for a *TPM Quote* operation over the 10th PCR value to the Trusted Software Stack (TSS) (jTSS in our case) [16]. The PCR's role is to store platform configurations (e.g., the BIOS, kernel and application executables). It is important to note that whenever a value is stored in the PCR, the hash of this value is appended with the current PCR value and the subsequent SHA-1 structure is stocked in the similar PCR. The SML and QUOTE are assembled in an attestation reply and are returned to the *CS* located at the server (step 6).

The role of the IMA in our architecture is to maintain a runtime integrity ML of all executable content, such as BIOS, bootloader, OS and applications, loaded into a Linux system since booting of the system (as depicted in **Fig. 5** located at the */sys/kernel/security/ima*). So for this, we created a mechanism in the Linux kernel and runtime

system to take integrity measurements as soon as executable content is loaded into the system. The order list of these measurements is maintained inside the Linux kernel. We use the TPM to protect the integrity of the in-kernel ML instead of holding the measurements directly.

PCR#	file-hash		template-hash	filename
10	2bbf0592c24b4ae980d0c03f21af7bf1b2723b66	ima	7c8982b2541320e3c5cfe58b5d6af1e3c9f3af7b	boot_aggregate
10	16020d495113eb400a4cb12bdc2a340b9f9c4461	ima	5fb8c8479f31574c435aa06d5bae6ae5a737ca81	sh
10	c982bd59464142c1ecdf62a7ac713971645653c5	ima	827c01503a92b1e202939ae5a0e3e4d5ff02f4ae	ld-linux.so.2
10	d782ea52513e8ecbef7e8fda0cabe69b9eec914b	ima	2e2046d5e8fb4ccc18f7fdf1a4dd076bf2333417	libc.so.6
10	7b9ac1cb86c3986ae37935dabf6b9018f43891e5	ima	5fb8c8479f31574c435aa06d5bae6ae5a737ca81	busybox
10	bf20139dd868dacc64018bf7f0406fa8a844129b	ima	a9b5059e18c01926a1fb057adad3f86bd7de4603	udevd
10	08bc991910ae50a23d133e09bfed71cac9efef3b	ima	b73213c29037d10d26b986f3d743b75d3de2c490	libselinux.so.1

Fig. 5. SML log example

The associated validation daemon located on both the server and the client performs basic functionalities such as (a) SML validation: This is to make sure the hashes in the SML received are corroborated against well-known trusted hashes in the DB. This is also to confirm that no rootkit activity is operating at the target (i.e., client) machine's platform (step 7), (b) PCR validation: The received quote PCR value is checked against the PCR value calculated from the hashes in the SML. This will make sure that the particular TPM signed the PCR quote which vouches for the authenticity of the SML (step 8), and (c) Nonce validation: This quote value includes the nonce to provide protection against a reply attack. This is to verify the TPM signature which ensures a genuine TPM hardware signed the acquired quote. This is carried out during the mutual attestation at which time the TPM must have signed the values of the PCR using the Private Key known only to the specific TPM (step 9). In the proposed architecture (**Fig. 4** on previous page) for the SML verification, a validation DB is created on both the client and server to verify the returned hashes (step 10). If all of these inspections are successful, the *CS* generates an XML response including information about whether the target platform attestation was successful or not. This XML response is released to the server (step 11) and on the basis of the attestation results, the server requests from the client, the user token (step 12).

Furthermore, we are assuming that the server requested the AS integrity. The AS in our case might be a daemon. The integrity of the AS has been assessed during the integrity challenge/response protocol. So before delivering the user token to the server, the client should, first, remotely attest the server platform trustworthiness. The attestation process will be performed in a similar fashion as per the client platform. Steps 13, 14, 15, 16, and 17 - 21 are similar to the steps 2, 3, 4, 5, and 6 - 10 given in **Fig. 4**.

5.3 Proof of Concept

We have successfully implemented a prototype of a *TMutualA* protocol. For details of the prototype setup (see **Table 1** in section 5.1). The run_challenger executes a java program on the challenger (server) to generate and send a nonce to the target platform (client). The *daemon- PD-ClientA* on the target platform, which keeps listening for this request, will perform the PCR quote (over the 10th PCR), extract the SML and PCR, and send back the trust tokens to the challenger platform. The basic requirement for mutual platform attestation is that both the challenger and target platforms are configured with the IMA, and the TPM must be enabled, activated and owned. In addition, the AIK must be generated to perform the quote operation. For the purpose of proof of concept implementation, we generated an AIK using the

trusted Java (jTSS libraries, jTPM tools) [19] and then passed it to the *bacaikExporter* class to generate an AIK_Pub key. The EK will attest the AIK when we execute the above process. However, the PrivacyCA provided by the IAIK [22] can also be used for signing the AIK or else a PrivacyCA can be designed as needed to carry out this process.

The *CS* on the server and client send a nonce that is encapsulated in the attestation request to the target platform that communicates with the TPM through the jTSS. The jTSS is used to perform the quote over the 10th PCR and the nonce value. The nonce actually guarantees freshness and provides protection against reply attacks. The attestation presenter also forwards the SML to the server in the attestation response. At the client and the server, attestation responses are handled by the Client and Server classes, respectively. Each class implements, internally, two private functions which generate and add-on a unique nonce for each time in the attestation request. For this, the public class function for validation purposes is exposed by our created interface. For the purpose of this implementation, we adopted three comprehensions of this interface and they are named as: *ValidationofReceivedPCR (VRPCR)*, *ValidationofReceivedSML (VRSML)*, and *ValidationofReceivedNonce (VRN)*. For details of the internal representation of the *PcrCompositestructure*, we refer interested readers to [23]. The *PD-ClientA* and *PD-ServerA* located on the client and the server, respectively, provide seamless handling of attestation requests that are emitted by a client or a server and provide a single public function to perform the attestation of the client or server. For the purpose of our work, we implemented this class as an abstract interface (as part of the attestation architecture) which consists of two Attesters named as: *AttestClientPCR, AttestClientSML* (at a client) and *AttestServerSML, AttestServerPCR* (at a server). The *AttestClientPCR* and *AttestServerPCR* provide a TPM-signed quote over the current PCR values and the nonce sent by the *CS*. Internally, we used AIK for signing the PCR values. The *AttestClientSML* and *AttestServerSML* return the SML retrieved from the *securityfs* of the client or server. The collected trust token which is returned via the integrity provider to the *CS* and *VD-ClientA* and *VD-ServerA* validates it respectively.

In this paper, we use the PrivacyCA notion in our mutual platform attestation implementation for a local true SSO scenario. The alternative way is to use Direct Anonymous Attestation (DAA) [28] which eliminates the need for a third party, PrivacyCA or authority to certify each AIK. In DAA the TPM gets a digital certificate from an entity called an "Issuer" for only one time, which is then used to sign random messages. However, to our knowledge, the DAA attestation has not been fully implemented yet for a local true SSO.

To handle potential errors, the prover must get some feedback (i.e., the particular hash corresponding to a particular executable). However, this could generate an issue if, say, one out of five hundred hashes could not be found in the DB of good hashes. We mitigate this issue in our proof of concept by always providing feedback to the prover, whether good or otherwise. When a bad hash is found, the process of integrity measurement is stopped and the status of the hash is reported to the system.

There are other parameters, which must be considered as well, for example, the security policy of the organization. Implementation of our proposed proof of concept will be most useful if the policy regarding access to sensitive information within the organization is critical.

Dealing with the complexity of different measurements within the trust chain, such as the measurement order, is also important. This is due to its effects on performance if the order of the measurements differs, i.e., it may cause performance degradation. The work presented in this paper has used an IMA and we have strictly followed the same order of measurements for both executables and configuration files. The measurements in the SML are in the same order, as are calculated and when this SML is sent to the server for verification, there is no possibility

that the order of the measurements will differ and hence we expect that there will be no effect on the performance.

We noted that the main drawback of the IMA is in dealing with the complexity of "good hashes" in an open environment, and that is the main reason why researchers proposed the DAA. In a single organization scenario with a heterogeneous OS, each client and SP needs to maintain its own DB for "good hashes". This DB must be able to do auto update if any new component is installed or found in the system. The validation process will be performed locally instead of sending it to the server, which makes it attractive to system owners and administrators in terms of maintenance.

6. Results, Advantages of the Proposed Scheme and Discussion

6.1 Proof of Concept

In this section, we will show how we evaluated the performance of our *TMutualA* protocol. First, we evaluated individually the client and server platform *attestation time, overall time* (*attestation time* + *network overhead*) and then the *complete round trip* (*attestation time* + *network overhead*) against the *number of measurements in the SML*.

Client and server machines platform attestation time taken

Fig. 6 and **Fig. 7** shows the performance results for client and server platforms: attestation time and overall time vs. the number of measurements in the SML. In **Fig. 6** and **Fig. 7**, we can clearly observe when the number of measurements in the SML increases, the attestation time (in ms) and the overall time also increases, accordingly. This means that when the number of executables and applications on each system increases it will distress the performance of the system.

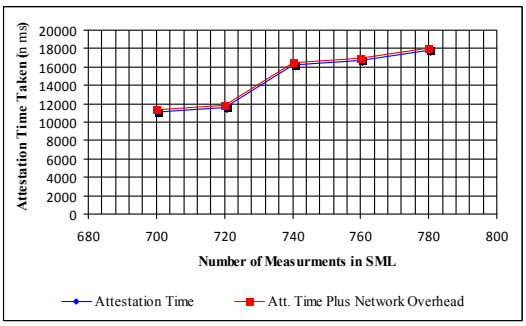

Fig. 6. Client machine platform attestation time, overall time Vs SML entries

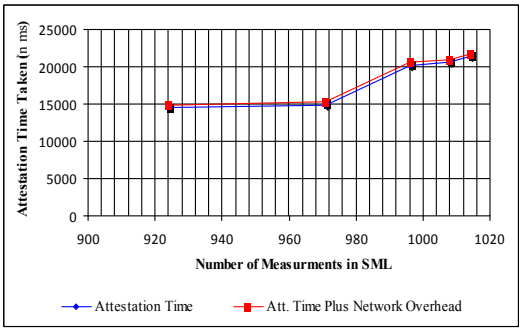

Fig. 7. Server machine platform attestation time, overall time Vs SML entries

Round trip time

Fig. 8 shows *the round trip time* of the mutual attestation process among the client and the server which includes sending and receiving the attestation request and response, validation of the Nonce, SML, and Certificate. In **Fig. 8**, we can observe this relationship when the *number of measurements in the SML* increments it affects the *attestation time* and *network latency* (i.e., *overall time* (in ms) minus *attestation time* (in ms)).

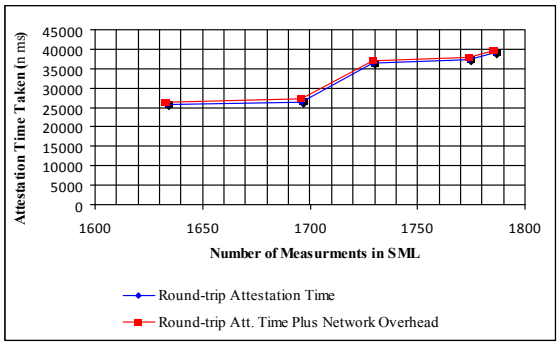

Fig. 8. Round-trip: Attestation time taken plus network overhead vs. SML entries

IMA based attestation scheme's performance comparison

Table 2 illustrates the performance comparison for the machine platform attestation in some related works [1][16][18][31] which make use of IMA in their studies. We, therefore, adopted IMA in which SML plays a major role in the platform mutual integrity validation. It has also been established that SML strongly affects the attestation time. This can be observed in **Fig 6**, **Fig. 7** and **Fig. 8**; when the number of SML entries increases, the respective SML attestation time also increases. The figures also show the attestation time plus the network overhead against the SML entries, which demonstrate similar performance. There is little effect of the overhead on the overall network performance.

Table 2. IMA based attestation scheme's performance comparison

Related work	Attestation Type	Attestation Mechanism	Number of SML	Attestation Time
Our Scheme	MA	IMA	SML ↑ (e.g. 709)	↑ (e.g. 11146ms)
Pashalidis et al. [1]	RA	IMA	SML ↑ (e.g. 500)	↑ (e.g. 10022ms)
Garriss et al. [16]	RA	IMA	SML ↑ (e.g. 580)	↑ (e.g. 11020ms)
Ali et al. [18]	RA	IMA	SML ↑ (e.g. 630)	↑ (e.g. 11100ms)
Sailer et al. [31]	MA	IMA	SML ↑ (e.g. 20000)	↑ (e.g. 23200ms)

Machine platform security and trustworthiness testing

We decided to do some tests on the suitability of our protocol. They consisted of two important aspects, i.e., (1) Goal of the protocol and (2) Act in response of a threat; these are described as follows:

- **Goal of the protocol** - To protect both the client and server machines' platform integrity against malevolent threats such as a rootkit.
- **Act in response to a threat** - A threat refers to a potential harm to the client or server machine platform integrity (which also means their respective health status), such as alteration of a legitimate machine platform measurement to become an infected or compromised state. Therefore, in this work, if either a client or server machine's health status is changed from a healthy (trustworthy) to an unhealthy (un-trustworthy) state, it

means that the machine's platform is not trustworthy any more.

We need to accomplish the *TMutualA* protocol specified goal for local true SSO systems and to check if its defense against a particular threat has also been achieved. We performed, exactly, the necessary experiments to achieve this. The results of our experiments show the secure and trustworthy system log and its corresponding response to a particular threat, respectively.

The experiment results obtained are given in **Fig. 9** and **Fig. 10** and show how our precise goal has been achieved against a particular threat. In the experiment, we demonstrated how the *TMutualA* protocol may react if a threat such as a rootkit lrk5 attack is detected. The log of the trustworthy and secure machine is given in **Fig. 9** below. After installing and executing the rootkit on both machines, we performed the attestation process for both machines to validate the trustworthiness of the platforms, and how the *TMutualA* protocol responded to it. **Fig. 10** below shows the log of a trusted machine after being compromised by the rootkit. The highlighted text (in pink) shows that the signature of the apt-get is altered by the rootkit.

To protect the measurement list's confidentiality, the authors assume that the mutual attestation mechanism is carried out on top of a safe and sound Secure Socket Layer (SSL) [36] link.

<smlcontents>
10 9fdd422f8b402d3111e410de7f2c08a55ca7d3d8 ima c3a0a5d8e10b69abb3f42bdcc827d3b7822ea44f boot_aggregate
...
10 40938443f71393ec89f3629f814b1ad752420b79 ima 5055599ce55ee09fd8cbe40933020673971cc596 apt-get
10 fa3788f31e6751e8c14050b601520631aade364b ima cce383d096e7ef6eed5dc4b377d0d735ff124f12 cksum
10 d509381332ae827598cb9652a2ff3111fee2ba83 ima 5188431849b4613152fd7bdba6a3ff0a4fd6424b 2998 </smlcontents>
Good hash:boot_aggregate
boot_aggregate -- 9fdd422f8b402d3111e410de7f2c08a55ca7d3d8
...
Good hash:apt-get
apt-get -- 40938443f71393ec89f3629f814b1ad752420b79
Good hash:cksum
cksum -- fa3788f31e6751e8c14050b601520631aade364b
2998 -- d509381332ae827598cb9652a2ff3111fee2ba83
Expected PCR value: E3A02AD1F9E1EFF4826AE7902C8B2B9E84D9803A
PcrCompositeHashExpected : F4816A84EC09E3E16116E889A5BD2633A3186DCA
PcrCompositeHashReceived : F4816A84EC09E3E16116E889A5BD2633A3186DCA
* ---------- SML Verification successful.
Verification Process Time Taken (ms):12612

Fig. 9. Secure and trustworthy system log

<smlcontents>
……
10 40938443f71393ec89f3629f814b1ad752420b79 ima 5055599ce55ee09fd8cbe40933020673971cc596 apt-get
……
</smlcontents>
……
Unknown hash:apt-get
apt-get – A4C32355m6826gh22x1354c021v3dx002741p009
.
.
.
* ---------- Unknown hash found. So SML Validation Faild.

Fig. 10. Rootkit infected or compromised system log

Security, trust and privacy (STP) integration comparison

As far as our knowledge goes, the *TMutualA* protocol proof of concept presented in this paper for local true SSO systems is the first practical demonstration. The proposed approach [25][30] was able to fulfill the STP integration requirements for a local domain as shown in **Table 3**. However, as shown in the **Table 3**, in an Inter-domain scenario, the *TMutualA* protocol may not fulfill the platform privacy concerns because of a lack of trust association between the local domain and the foreign-domain SPs.

Table 3. Comparison of the security, trust and privacy features

	Features of the proposed scheme	[1]	[31]	[16]	[18]	[25][30]
Security	Bi-directional threats detection and prevention	x	√	x	x	√
	Binding resource access & Authentication token release with MA successful run	x	x	x	x	√
Trust	Mutual trust establishment	x	√	x	x	√
	Trust association	1-way	2-way	1-way	1-way	2-way
Privacy	Platform privacy in local domain	√	√	√	√	√
	User ID privacy (Anonymity & Unlinkability)	x	x	x	x	√
	Platform privacy in external domain	x	x	x	x	x

6.2 Advantages of the Proposed Scheme

The advantages of the proposed scheme are as follows.

- Firstly, our proposed scheme may be used both in the local domain (network) as well as in a federated environment between the local and foreign domains. The main strength of the proposed approach in the internal domain include: (1) mutual platform privacy protection and (2) the updating of the machine measurements is an easy task as compared to other proposed schemes.
- Secondly, in the proposed mutual attestation based scheme, the web server gives to a client its access right to a resource or a service if and only if their machine integrity is mutually validated. In this way, we have eliminated any kind of machine hijacking by rootkits in the online transactions.
- Thirdly, the proposed scheme binds resource access permission by the server with the user authentication token release and validation, which depends on the above mutual attestation's successful execution which is based on the mutual machine integrity validation.

6.3 Discussion

In addition to the performance evaluation, we also look at the security and trustworthiness, usability, AIK privacy, trust relationship and the identity federation of the proposed scheme as follows:

- **Security and trustworthiness:** We tested our protocol with and without the existence of the rootkit. In the absence of the rootkit, both platforms successfully performed mutual attestation and established trustworthiness. This means all hashes in the SML were received and its comparison with good known hashes (in DB) showed no rootkit or any other malicious software running on it.
- **Trust relationship:** The trust relationship between the user, PrivacyCA and SP in the *TMutualA* protocol for a local true SSO was achieved as follows. The user and SP

should both trust the PrivacyCA for certification of the AIKs. This means that the user needs to trust the PrivacyCA chosen by the SP to certify the AIKs and similarly the SP needs to trust the PrivacyCA selected by the user to certify the AIKs. The SP must also trust all executable libraries that are executed before the AS.

- **Identity federation:** To achieve the identity federation in an open environment, the user must first obtain a blind token using the blind signature [24] method described in [25][26][30]. The acquired token is then stored in the user's browser or on the system that would be accessible only by the targeted SP. The AS, which resides on the user's system, is responsible to capture the token, verify publicly or privately with the Blind Token Generating Service (BTGS) [25] and send it to the target SP. The trust relationship is then built between the users of the federated systems and the target SP, which also depends on the agreed policy between the participating parties, and will be executed if the attestation is successful. For the identity federation in the local true SSO, the user and SP must agree on common policies.

7. Conclusion

The authors constructed a practicable STP framework [35] which integrates an open source, a standard Federated Identity and an Access Management (FIAM) system using TC. In this paper, we have demonstrated a *TMutualA* protocol proof of concept implementation for a local true SSO. The proof of concept implementation, which is based on the *TMutualA* protocol, has shown that platform trust must be successfully established before any transaction can take place. Otherwise, if trust is not established, as in the case of a system with rootkits running, transactions will be stopped. In evaluating its performance, our preliminary experimental results showed that when the number of SMLs increases, the attestation time taken will also increase, hence the performance is affected.

The architecture presented in this paper can be adapted to different attestation methods such as PBA [17] etc. We can safely predict that the latter attestation mechanisms will affect the client as well as a server's platform performance and privacy. For the purpose of this paper, we have implemented an IMA based mutual attestation mechanism for local/native SSO systems. According to our current knowledge, there is no other mutual attestation mechanism that has been implemented practically, mainly due to the uncertainties surrounding the determination of the exact properties to be used in the PBA. Hence, its implementation using our proposed architecture would need further research.

In addition, important issues such as the development of the AS and BTGS, interaction between these two, and the policy specification for token and resource release if platforms are mutually attested need further research.

References

[1] Andreas Pashalidis and Chris Mitchell, "Single sign-on using trusted platform," in *Proc. of 6th Int. Conf. on Information Security*, pp.54-68, Oct.2003. Article (CrossRef Link).

[2] Reiner Sailer, Xiaolan Zhang, Trent Jaeger and Leendert van Doorn, "Design and implementation of a TCG-based Integrity Measurement Architecture," in *Proc. of 13th USENIX Security Symposium*, pp.223-238, Aug.2004.

[3] T. A. Parker, "Single sign-on systems-the technologies and the products," in *Proc. of European Convention on Security and Detection*, pp.151-155, May.1995. Article (CrossRef Link).

[4] Andreas Pashalidis and Chris Mitchell, "A taxonomy of single sign-on systems," in *Proc. of 8th Australian Conf. on Information Security and Privacy*, pp.249-264, Jul.2003. Article (CrossRef

Link).
[5] R. Oppliger, "Microsoft .net passport: A security analysis," *IEEE Computer,* vol.36, no.7, pp-29-35, Jul.2003. Article (CrossRef Link).
[6] Liberty Alliance Project. http://www.projectliberty.org/
[7] Mark Needleman, "The shibboleth authentication/ authorization system," *Journal Serials Review,* vol.30, no.3, pp.252-253, Aug.2004. Article (CrossRef Link).
[8] Trusted Computing Group (TCG). http://www.trustedcomputinggroup.org
[9] Sandeep Bajikar, "Trusted Platform Module (TPM) based Security on Notebooks PCs-White Paper," Technical Report, Intel Corporation, Jun.2002.
[10] TCG, "Trusted Computing Group (TCG) specification architecture overview," revision 1.4, pp.11-12, Technical Report, Aug.2007.
[11] John Hughes and Eve Maler, "Security Assertion Markup Language (SAML) 2.0 technical overview," Technical Report, Jul.2004.
[12] S. Pearson, *Trusted Computing Platforms: TCPA Technology in Context,* 1st Edition, Prentice Hall, New Jersey, 2002.
[13] Trusted Computing Platform Alliance (TCPA): Trusted Platform Module Protection Profile, v 1.9.7, Jul.2002.
[14] Eimear Gallery and Chris Mitchell, "Trusted computing: security and application," *J. Cryptologia*, vol. 33, no. 3, pp.217-245, Jul.2009. Article (CrossRef Link).
[15] Donald Eastlake and Paul Jones, "RFC 3174 - US Secure Hash Algorithm 1 (SHA1)," September 2001. http://www.openrfc.org/rfc/3174.pdf
[16] S. Garriss, R. C´aceres, S. Berger, R. Sailer, L. Doorn and X. Zhang, "Towards trustworthy kiosk computing," in *Proc. of 8th IEEE Workshop on Mobile Computing Systems and Applications*, pp.41-45, Mar.2007. Article (CrossRef Link).
[17] Ahmad-Reza Sadeghi and Christian Stüble, "Property-based attestation for computing platforms: caring about properties, not mechanisms," in *Proc. of the 2004 Workshop on New Security Paradigms*, pp.67-77, Sep. 2004. Article (CrossRef Link).
[18] Tamleek Ali and Mohammad Numan, "Incorporating remote attestation for end-to-end protection in web communication paradigm," in *Proc. of the 3rd Int. Conf. on Internet Technologies and Applications*, Sep.2009.
[19] Trusted Computing for the java (tm) Platform. http://trustedjava.sourceforge.net/
[20] TCG, "Trusted Platform Module (TPM) specification v1.2 enhances security," Jun.2004.
[21] Tim Bray et al., "Extensible markup language (xml)," Technical Report, Aug.1997. http://www.w3pdf.com/W3cSpec/XML/2/REC-xml11-20060816.pdf
[22] Institute of Applied Information Processing and Communication, Graz University of Technology, Austria. http://www.iaik.tugraz.at/content/about_iaik/
[23] Massom Alam, Xinwen Zhang, Mohammad Nauman and Tamleek Ali, "Behavioral attestation for web services (BA4WS)," in *Proc. of ACM Workshop on Secure Web Services*, pp.21-28, Oct.2008. Article (CrossRef Link).
[24] David Chaum, "Blind signatures for untraceable payments," in *Proc. of Crypto '82*, pp.199-203, 1982.
[25] Zubair Ahmad Khattak, Jamalul-lail Ab Manan and Suziah Sulaiman, "Analysis of open environment sign-in schemes- privacy enhanced & trustworthy approach," *Journal Advances in Information Technology*, vol.2, no.2, pp.109-121, 2011. Article (CrossRef Link).
[26] Arkajit Dey and Stephen Weis, "PseudoID: Enhancing privacy in federated login," in *Proc. of the 10th Hot Topics in Privacy Enhancing Technologies*, pp.95-107, Jul.2010.
[27] Masoom Alam, Xinwen Zhang, Mohammad Nauman, Tamleek Ali and Jean-Pierre Seifert, "Model-based behavioral attestation," in *Proc. of the 13th ACM Symposium on Access Control Models and Technologies*, pp.175-184, Jun.2012. Article (CrossRef Link).
[28] Erine Brickell, Jan Camenisch and Liqun Chen, "Direct anonymous attestation," in *Proc. of the 11th ACM conference on Computer and communications security*, pp. 132-145, Oct.2004. Article (CrossRef Link).
[29] Jaehong Park and Ravi Sandhu, "Towards usage control models: beyond traditional access

control," in *Proc. of the 7th ACM Symposium on Access Control Models and Technologies*, pp. 57-64, Jun.2002. Article (CrossRef Link).
[30] Zubair Ahmad Khattak, Jamalul-lail Ab Manan and Suziah Sulaiman, "Proof of concept implementation of trustworthy mutual attestation architecture for local true SSO," in *Proc. of the 10th Int. Conf. on Security and Management,* pp. 721-724, Jul. 2011.
[31] Reiner Sailer, Leendert van Doorn and James Ward, "The Role of TPM in Enterprise Security," Technical Report, IBM Research, Oct.2004.
[32] S. Balfe, E. Gallery, C. Mitchell and K. Paterson, "Crimeware and Trusted Computing," in *Crimeware Understanding New Attacks and Defenses*, pp. 457-472, Apr.2008.
[33] Jing Zhan, Huanguo Zhang and Fei Yan, "Building Trusted Sub-domain for the Grid with trusted computing," in *3rd SKLOIS Conf. on Information Security and Cryptology*, pp. 463-471, Sep.2007. Article (CrossRef Link).
[34] Ramon Cáceres and Reiner Sailer, "Trusted mobile computing," in *Proc. of IFIP Networking Workshop on Security and Privacy in Mobile and Wireless Networking*, May 2006.
[35] Z. A. Khattak, J-L. Manan and S. Sulaiman, "Security, trust and privacy framework for federated single sign-on environment," in *Proc. of the 5th Int. Conf. on Information Technology and Multimedia,* pp. 1-6, Nov.2011. Article (CrossRef Link).
[36] Mittal Bhiogade, "Secure Socket Layer," in *Proc. of the Informing Science and Information Technology Education Conference*, pp. 85-90, Jun.2002. Article (CrossRef Link).

Zubair Ahmad Khattak received his Bachelor of Science (BSc) and Master of Science (MSc) in Computer Science from the University of Peshawar, Khyber-Pakhtunkhwa, Pakistan, in 1998 and 2004, respectively. He completed his Master in Information Technology (MIT) in Computer Science and Information Technology (a composite program) from the International Islamic University, Malaysia in 2008. Since February 2009, he has been pursuing a PhD degree at the Department of Computer and Information Sciences, Universiti Teknologi PETRONAS, Perak, Malaysia. In April 2009, he started his Research Fellowship at the Information Security Cluster, MIMOS Berhad, KL, Malaysia. He is currently attached to the Advanced Analysis and Modeling (ADAM) Cluster, MIMOS Berhad, KL, Malaysia. His research interests include Trusted Computing, Identity and Access Management particularly in the areas of Federated Identity and Access Management (FIAM) Systems, and Unified Security, Trust and Privacy (STP) Solutions for FIAM System.

Jamalul-lail Ab Manan graduated from the University of Sheffield, UK with a Bachelor in Electrical Engineering (BEng). He pursued his Master of Science (MSc) in Microprocessor Engineering in the University of Bradford, UK and PhD in Communications Engineering in the University of Strathclyde, Glasgow, UK. He is currently the Head of the Cryptography Lab, Advanced Analysis and Modeling (ADAM) Cluster, MIMOS Berhad. He has many years of experience in teaching Electrical and Electronics, Microprocessor Engineering and Network Security. He also has many years of industrial experience as a Network Engineer, Senior Manager and Senior Vice President in ICT based government linked companies in Malaysia. In MIMOS Berhad, his current research focus is Information Security, particularly in Trusted Computing, Cryptography and Unified Solutions for Security, Trust and Privacy(STP).

Suziah Sulaiman obtained her Bachelor of Science in Mathematics from the Dalhousie University, Canada. She completed her Master of Science in Business Information Systems, University of East London, UK and Master of Philosophy (Human Factors and Computing) from the South Bank University, UK. She received her PhD in Computer Science from University College London, UK. She is currently a Senior Lecturer, attached to the Department of Computer and Information Sciences, Universiti Teknologi PETRONAS, Malaysia. Her current interests include Human Computer Interaction, User Haptics Experience, and Usability Evaluation.