Single Sign-On using Trusted Platforms

Andreas Pashalidis** and Chris J. Mitchell

Royal Holloway, University of London,
Egham, Surrey, TW20 0EX, United Kingdom,
{A.Pashalidis, C.Mitchell}@rhul.ac.uk,
WWW home page: http://www.isg.rhul.ac.uk

Abstract. At present, network users have to remember a username and a corresponding password for every service with which they are registered. One solution to the security and usability implications of this situation is Single Sign-On, whereby the user authenticates only once to an 'Authentication Service Provider' (ASP) and subsequently uses disparate Service Providers (SPs) without necessarily re-authenticating. The information about the user's authentication status is handled between the ASP and the desired SP transparently to the user. This paper describes a method by which the end-user's computing platform itself plays the role of the ASP. The platform has to be a Trusted Platform conforming to the Trusted Computing Platform Alliance (TCPA) specifications. The relevant TCPA architectural components and security services are described and associated threats are analysed.

Keywords: single sign-on, TCPA, authentication

1 Introduction

Network users have to remember usernames and passwords for every Service Provider[1] (SP) they are registered with. If they could remember different, ideally 'secure' [5] passwords (and corresponding user names) for every such SP, they might have done everything they could from their perspective with respect to security. Unfortunately, this has proven to be very difficult, and thus users either write their passwords down (on paper or in a file in their computers), or — more commonly — use the same password with every SP they visit. This behaviour has a number of security implications; most obviously any SP can impersonate the user to all SPs with whom the same password is used. Thus, it might be possible for, say, the administrator of a news portal website to gain access to the portal visitors' bank accounts, credit card details or emails.

Single sign-on (SSO) is a technique whereby the user authenticates only once to an entity called an *Authentication Service Provider* (ASP) and is subsequently

** The author is sponsored by the State Scholarship Foundation of Greece.

[1] In the context of this paper a service provider is any entity that provides a service or content to network users, e.g. messenger/web services, FTP/web sites, or streaming media providers.

logged into a number of SPs without necessarily re-authenticating. This seamless experience increases the usability of the network as a whole and eliminates the security implications mentioned above (but introduces its own). It is obvious that, under SSO, SPs require some kind of notification from the ASP about the user's authentication status. These notifications are termed *authentication assertions*. The SP assesses the authentication assertions provided by the ASP and determines whether or not to grant access to a protected resource to the specified user.

The Liberty Alliance, a consortium of over 160 companies, has developed a set of open specifications that, among other things, provide web-based SSO [8–13] (the specifications are publicly available at `www.projectliberty.org`). Authentication assertions under Liberty are based on the Security Assertions Markup Language (SAML), a platform-independent framework for exchanging authentication and authorisation data [16]. Liberty employs the notion of *trust circles*, formed by trusted ASPs and the SPs that rely on the ASPs for the purposes of user authentication (the 'relying SPs'). Users are uniquely identified by pseudonymous identifiers that do not contain any personally identifying information. Relationships between an ASP and relying SPs are based on contractual agreements outside the scope of the specifications.

Liberty specifies generic requirements for the protocols for conveying assertion requests and responses between parties. Concrete protocol bindings are only specified in the context of a Liberty *profile*. In the 'Web Browser/POST profile', for example, a scheme for web SSO is specified where SSL or TLS are the underlying mechanisms that provide for authentication, integrity and confidentiality of authentication information [11].

The actual authentication method used by the ASP is not specified by Liberty. Authentication assertions may, however, include descriptions of the initial user identification procedure at the ASP, the physical, operational and technical protection procedures, and the authentication method used (such as password, smartcard or public key certificate) [9].

The Trusted Computing Platform Alliance[2] (TCPA) is an industry working group with the following main goal: "Through the collaboration of hardware, software, communications and technology vendors, [to] drive and implement TCPA specifications for an enhanced hardware and operating system based trusted computing platform that implements trust into client, server, networking, and communications platforms" [19]. A computing platform that conforms to the TCPA specifications is termed a 'Trusted Platform' (TP)[3]. SSO has recently been identified as one of the applications that can benefit from TPs [1].

This paper describes a method where a user's TP plays the role of the ASP in order to achieve SSO among disparate SPs. The next section is a review

[2] http://www.trustedcomputing.org

[3] The TCPA specifications have very recently been superseded by the specifications of the Trusted Computing Group (www.trustedcomputinggroup.org) which has replaced the TCPA. The TCG specifications appear to support a similar SSO process to that described in this paper.

of relevant TCPA architectural components and security services. Section 3 describes the SSO protocol, while section 4 analyses the associated security threats. Section 5 discusses advantages and disadvantages and sections 6 and 7 give an overview of related work and conclude the paper.

2 Review of TCPA security services

This section introduces those components of the TCPA specification that are relevant to this paper. For a full description see, for example, [1, 4].

Every TP has a Trusted Platform Module (TPM), essentially a crypto co-processor with special functionality. It is a chip closely bound to a computing platform's main hardware (e.g. soldered to a PC's motherboard). Information stored in the TPM (in so-called 'shielded locations') is resistant to any direct software attack, as the information can only be accessed through well-defined commands known as *TPM capabilities*. Three TCPA security services are relevant to this paper, namely TPM Identities, Integrity Metrics and Key Certification.

2.1 TPM identities

Every TPM has a unique RSA key pair imprinted in it, termed the 'Endorsement Key'. The private part of this key pair (PRVEK) *never* leaves the TPM and is used only to decrypt certain data structures that are sent to the TPM for very specific purposes. The public part (PUBEK) can be retrieved from the TPM, but only under certain conditions.

Exposing a TPM's PUBEK outside the TP would enable third parties to uniquely identify that particular platform, which is a potentially serious privacy threat in today's interconnected world. TCPA therefore introduces the notion of *TPM Identities*, which allow a user to signify to third parties that she is using a genuine (in the sense of TCPA-conformant) TP without revealing its particular identity. A TP can have an arbitrary number of TPM Identities. Each TPM Identity has an associated RSA key pair, termed the TPM *Identity Key* (IDK) which can only be used for signature generation and verification. The private part of an IDK is *never* exposed outside the TPM in the clear.

IDKs have to be certified by a so-called Privacy Certification Authority (PRV-CA) before use. A simplified description of the procedure by which a public key certificate for an IDK can be obtained from a PRV-CA is given below.

1. The TPM owner issues a command to the TPM (TPM_MakeIdentity) that generates a new IDK for the TPM, and outputs the public part of it.
2. The TPM owner contacts a PRV-CA of his/her choice and submits (among other things) the new IDK's public part and evidence that proves the authenticity of the TP. This latter evidence includes the following two types of credential.

   - An Endorsement Credential that contains the TPM's PUBEK. This is essentially a public key certificate issued by the TPM manufacturer. The PRV-CA can use the PUBEK to encrypt data that can only be decrypted by the specific TPM. Unfortunately, it also allows the PRV-CA to uniquely identify the TP in question.
   - A Platform Credential — a digital certificate that is typically issued by the platform manufacturer — that describes the general characteristics of the computing platform and binds the Endorsement Credential to a Conformance Credential. The latter is a document produced by a test laboratory that has independently verified that a particular TPM/Platform design conforms to the TCPA specification.
3. The PRV-CA verifies the supplied certificates against locally-stored trusted root public keys of their respective issuing authorities. If convinced that the TP in question is genuine, the PRV-CA generates a public key certificate for the new IDK, known as an *Identity Credential*. The Identity Credential is signed by the PRV-CA and binds the public part of the new IDK to a user-chosen text label and general information about the platform type. At this point the PRV-CA has no assurance that the IDK was *indeed* generated by the TPM in question. In order to provide this assurance, the PRV-CA encrypts the Identity Credential using a symmetric session key which is itself encrypted using the TPM's PUBEK, such that only the intended TPM will be able to decrypt it. The resulting encrypted data and session key are sent back to the TP.
4. The TPM owner must then issue a command (TPM_ActivateTPMIdentity) to activate the new IDK. This command needs to be submitted with the encrypted session key, and the TPM will only activate the new IDK if the certified key inside the Identity Credential corresponds to an IDK of the TPM that has yet to be activated.

After having successfully activated an IDK, it can only be used to digitally sign data structures *within* the TPM (since the private part of an IDK never leaves a TPM in cleartext). The TP user can then send the IDK-signed data along with the corresponding Identity Credential to a third party. The third party can verify the Identity Credential using the trusted root public key of the issuing PRV-CA. If the signature of the IDK-signed data verifies using the public key from the Identity Credential, then the third party can be confident that the data was signed by a genuine TPM. At the same time there is no way for the third party to uniquely identify the TP in question.

Every IDK (in fact, every TPM-protected object) has certain authorisation data associated with it, which is specified at generation time. Knowledge of an IDK's authorisation data must be demonstrated to the TPM prior to use of TPM functions that need to access that IDK. Since IDKs can only be generated by TPM owners, there is an issue if the ultimate IDK user is not the TPM owner (for example, the TPM owner could be a company's sales department and the employees its users). The TCPA specification [4] defines a protocol, termed the Asymmetric Authorisation Change Protocol (AACP), that allows the

authorisation data of an IDK to be changed, such that the previous authorisation data owner does not get to know the new authorisation data (the protocol, however, requires participation of the previous authorisation data owner). In this way the TPM owner can generate and assign different sets of TPM Identities to different users of the same TP.

2.2 Integrity metrics

TCPA-enabled platforms are able to reliably measure, store and report their configuration and software state. This is achieved by applying a one-way hash function to software that is about to be executed on the TP; the resulting hash value (called an 'integrity metric') is stored inside one of the TPM's Platform Configuration Registers (PCRs). The initial value of each PCR is zero, but their values are updated during a TP's boot cycle. In particular, whenever the TP is about to execute a critical piece of software or firmware (such as the BIOS, the OS and applications), the following actions are performed:

1. The software about to be executed is cryptographically hashed (using SHA-1 [15]).
2. The resulting digest is concatenated with the current value of a specific PCR and this concatenation is hashed again.
3. The digest resulting from step 2 becomes the new value of the affected PCR.
4. An entry is appended to a log file called the 'history information'. The entry contains information about the measured event (such as the software name and version), which PCR was affected, and a Validation Certificate (or a reference to it). The latter is a certificate issued and signed by the measured component's manufacturer or vendor that binds the component to the expected hash value of step 1.
5. The measured software is executed.

In this way, accurate 'integrity metrics' of the platform's software state are kept inside the PCRs, where they are protected against interference from software. In order for a communicating third party to assess the software state of a TP, it issues an 'Integrity Challenge' to the TP. The challenge includes a nonce that protects against replay attacks. The TP responds with an 'Integrity Response' that includes: the current PCR values, a digital signature over the PCR values and the nonce generated using one of the TPM's IDKs (using a well-defined TPM capability (TPM_Quote)), the Identity Credential for the IDK used to produce the signature, and the history information.

The communicating third party can now assess the trustworthiness of the received Integrity Response. It does so by

- Verifying the Identity Credential with the trusted public key of the issuing PRV-CA.
- Verifying the signature over the PCR values and the nonce using the public key from the Identity Credential.

– Evaluating the history information and verifying that the given sequence of measured components yields the current values of the PCRs. This includes verifying the Validation Certificates issued by the manufacturers or vendors of the respective components.

If the above steps succeed, the third party can be confident that the TP's software state is as represented by the quoted integrity metrics (and has not been modified). The third party then decides whether or not to trust this software state for the intended purpose.

2.3 Key certification

A TPM can cryptographically protect different types of data in a facility known as the TPM's *Protected Storage*. One of these data types is a *non-migratable Signature Key* (SK), i.e. an RSA key generated by the TPM using a TCPA capability (TPM_CreateWrapKey). These SKs may only be used to sign data and — as opposed to migratable keys — the private (signing) part of these keys is *never* exported from the TPM in unencrypted form. This means that signing with non-migratable SKs can only be performed by the TPM itself.

The non-migratable SK is a data structure that can be signed by an IDK, using another well-defined TPM capability (TPM_CertifyKey). The result is a public key certificate for the public part of the SK. A third party can inspect this public key certificate in conjunction with the Identity Credential corresponding to the IDK used to sign it. If the Identity Credential verifies against the PRV-CA's trusted root public key, and the public key certificate of the SK verifies against the Identity Credential, then the third party can be confident that any data signed with the non-migratable SK in question has been signed by a genuine TPM.

3 Using Trusted Platforms for SSO

Before describing the TCPA-based SSO scheme, we make two key observations on which the design of the scheme is based.

Firstly, as pointed out in [1], user authentication can be delegated to the user's TP and carried out by an Authentication Service (AS) within that TP. The AS may be purely software, or a combination of software and hardware, depending on the authentication method. If, for example, authentication is based on a username/password pair, a software AS will be adequate. Alternatively, if a smartcard is involved, the AS will consist of the card reader and the supporting software. In any case, the AS's integrity will be measured in the TPM's PCRs.

The second observation is that Identity Credentials corresponding to TPM Identities are unique. This is because, since they are X.509 public key certificates [7], they carry a unique serial number assigned by the issuing PRV-CA. Thus, the (PRV-CA Identifier, Serial Number) tuple uniquely identifies any given Identity Credential. Furthermore, Identity Credentials are pseudonymous as they

do not contain any personal data about the user. Therefore, they can serve as opaque user identifiers in an SSO context. If Identity Credentials are used by SPs to uniquely identify users, the corresponding TPM Identities will act as SSO Identities for users. In the remainder of this paper the term 'SSO Identity' therefore refers to a designated TPM Identity that is used for user identification at one or more SPs.

3.1 System entities

The main entities involved in the SSO scheme described in this paper are the user system and the relying SPs.

User system The user system is made up of the user and his/her TCPA-conformant computing platform which plays the role of both the network access device (and therefore the SPs' client) and the ASP for relying SPs.

A set of TPM Identities that will act as SSO Identities needs to be generated and activated for each user of a given TP, as explained in section 2.1. As SSO Identities can only be created by TPM owners, there is an issue if the TPM owner is not the ultimate user, or the TP has multiple users. In such a case, the AS (or some other component in the TP) has to manage the authorisation data of IDKs corresponding to SSO Identities. In particular, it should allow TPM owners to create a set of distinct SSO Identities for each user of the platform (see also section 2.1).

For the purposes of SSO, relying SPs communicate with the AS that resides within the user's TP. The AS's task is to locally authenticate the user and subsequently provide authentication assertions to the relying SPs in order to facilitate SSO. In such a case the AS will be tightly integrated into the TP's operating system, probably as a dæmon (service) or part of the operating system login mechanism. As the integrity of the AS is reliably measured in the TPM's PCRs, SPs can assess its trustworthiness using an Integrity Challenge/Response session as described in section 2.2. The key point is that the Integrity Response will be generated using one of the user's SSO Identities.

Service providers SPs require user authentication before granting access to protected resources, and acquire the necessary authentication assertions from the AS inside the users' TP.

Before trusting authentication assertions, SPs need to verify the AS's integrity and assess its trustworthiness using an Integrity Challenge/Response session, as mentioned above. Only if the AS in place is judged trustworthy by an SP can authentication assertions subsequently conveyed from the AS to the SP be trusted. It is important, however, to note that the Identity Credential corresponding to the user's SSO Identity is conveyed from the AS to the SP during the Integrity Challenge/Response session. The SP can use this unique Identity Credential in order to differentiate between SSO Identities (and therefore user

accounts). In this way the TCPA Integrity Challenge/Response mechanism simultaneously provides integrity assurance and user identification. Of course this relies on an initial registration procedure between a user and an SP, during which the user registers using a particular SSO Identity. This is referred to below as SSO Identity association. The protocol described in section 3.3 supports both this initial registration process and subsequent SSO authentication.

In order to guard against man-in-the-middle attacks (section 4.3), users have to somehow authenticate the desired SP before releasing an Integrity Response or authentication assertions. The SSO protocol described in section 3.3 therefore requires that every SP has a well-known, human-readable unique identifier (SPID) such as a Uniform Resource Identifier [2]. This SPID must be easily identifiable by the end user.

3.2 Trust relationships

When using the SSO scheme described herein, the trust relationships between the different entities are as follows. End users need to trust the PRV-CA(s) chosen to certify the IDKs that correspond to their SSO Identities, to protect their privacy (see section 4.4). SPs, on the other hand, need to trust

- The PRV-CA(s) chosen by the user to certify the IDKs of their SSO Identities.
- The AS installed and run on the user TP, as well as any software executed before the AS (such as the BIOS and the Operating System).

Generally speaking, trusting a PRV-CA also means trusting all entities vouched for by that PRV-CA, namely the TPM manufacturer, the TP manufacturer, the conformance testing lab, and the TCPA specification itself.

In Liberty terms [8], no explicit circles of trust need to be formed by SPs (e.g. no explicit contractual agreements are made between SPs and the AS in the TP). SSO can be achieved at those SPs that trust the PRV-CA(s) chosen by the users and the system software (AS, OS, BIOS) that runs on their TPs. Therefore, 'trust circles' are defined by the intersection of the sets of PRV-CAs and system software configurations that are trusted both by the user and the SPs. (This may, of course, involve contractual and/or liability transfer arrangements between SPs and AS system providers and/or TP vendors).

3.3 The SSO protocol

The SSO protocol consists of a single message exchange between the SP and the AS and starts when the user requests a protected resource from the SP. It can also be used for initial user registration at a SP. A detailed description follows.

1. The SP sends a message to the AS, which consists of its SPID, an Integrity Challenge (section 2.2) and, in Liberty terms [13], an *authentication request*.
2. The AS asks the user to positively verify and acknowledge the SPID of the SP before continuing. This is to counter the attack described in section 4.3.

3. If the user's current authentication status does not satisfy the requirements of the authentication assertion request from step 1, the AS now authenticates the user by some (acceptable) means.
4. If more than one SSO Identity association exists for this SP, the AS asks the user which SSO Identity to use for this session. If no SSO Identity association exists for this SP, this protocol instance is used for initial user registration at the SP. In this case the AS asks the user which of his/her SSO Identities to *register* with this SP.
5. The AS constructs the following data objects:
   - The Integrity Response using the IDK corresponding to the chosen SSO Identity, as explained in section 2.2.
   - A public key certificate for a non-migratable SK, generated using the aforementioned IDK as explained in section 2.3. For privacy reasons, there must exist a strict one-to-one relationship between this SK and the IDK. (The SK may be created dynamically or be permanently stored in the TPM's Protected Storage.)
   - An authentication assertion that contains the SPID of the SP and information about the user's authentication status. The AS signs the authentication assertion using the non-migratable SK (see Figure 1).

   These data objects are concatenated and sent back to the SP as a reply. In Liberty terms [13], this message corresponds to an *authentication response*.
6. The SP evaluates the Integrity Response as explained in section 2.2. If this assessment completes satisfactorily, and if the (TPM measured) AS in place is trusted, the process continues normally. Otherwise, SSO/Registration fails.
7. The SP verifies the SK's public key certificate against the Identity Credential contained in the Integrity Response, as explained in section 2.3. If verification fails, SSO/Registration also fails.
8. The SP verifies the signature of the authentication assertion against the (verified) public key certificate of the SK. If verification fails, SSO/Registration also fails.
9. The SP assesses the authentication assertion provided by the AS. This includes checking that the SPID inside the authentication assertion actually represents the SP. If the SP's authentication requirements are met, SSO/Registration succeeds and access to the protected resource is granted. Otherwise, SSO/Registration fails.

The relationships between the different types of involved keys and the PCR values of the TP are shown in Figure 1.

If a protocol run is used for initial user registration at the SP (as determined in step 4), and if registration actually succeeds (as determined in step 9), the AS permanently stores the new SSO Identity/SP association and the SP creates a new user account for the newly encountered SSO Identity Credential.

The AS achieves SSO by maintaining (caching) the user authentication status within the AS and by storing SSO Identity/SP associations permanently. Every time the user requests a protected resource from a SP with whom a SSO Identity

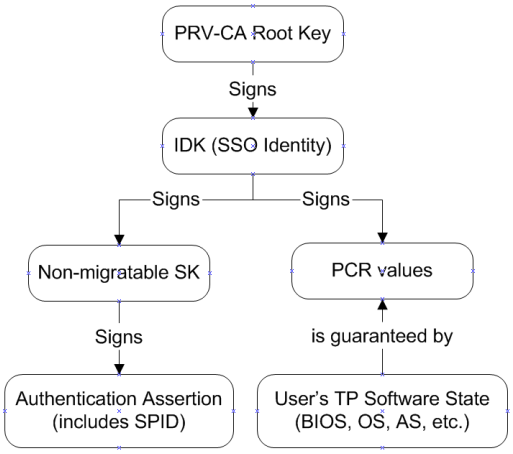

Fig. 1. Data structure relations

association exists, the above protocol will run, without necessarily requiring an authenticated user to re-authenticate.

The AS can achieve single logout by 'remembering' every open SP session. If the user requests a single logout, the AS can contact each SP in turn in order to log out the user.

3.4 SSO identity federation

The TCPA specifications do not allow the migration of TPM Identities from one TP to another. This has the immediate consequence that the SSO scheme supports user mobility only if the TP itself provides for this (such as a laptop or PDA).

Although it is advisable from a privacy perspective to use different SSO Identities with every SP (see section 4.4), it may be desirable to federate two (or more) user SSO Identities[4] at a SP, such that the user will be able to use any of the federated Identities for logging into his/her account. If the federated SSO Identities are generated on different TPs, the user will be able to enjoy SSO from any of these TPs.

Such SSO Identity federation is outside the scope of the scheme described herein, but could be provided as a service by individual SPs, for example using a one-time password scheme; see, for example, [14]. The user logs into the SP using one Identity and obtains a one-time password. He/she then logs in with another Identity and uses the one-time password to federate the two Identities.

[4] Note that the term 'identity federation' is used in a slightly different manner to that employed in the Liberty specifications. While in Liberty terms 'Identity Federation' means linking user identities at separate SPs, in this paper the term means linking user identities at the same SP.

Similar techniques could be used to support the transition from legacy user identification/authentication schemes to the one described in this paper.

4 Threat analysis

In this section the threats to the scheme and corresponding countermeasures are considered. The threats can be divided into four categories: threats in the user system, threats to the SPs, threats to the communication links, and threats from collusion of entities. Note that we do not consider threats to the underlying cryptographic framework provided by TCPA.

4.1 Threats to the user system

The AS within the user's TP is trusted by both users and SPs; it can easily impersonate users to SPs. The proper functioning of the AS itself must therefore be rigorously checked and its integrity protected. TCPA services that might prove useful in helping meet these goals are 'protected storage' and 'secure boot'.

As the AS also has access to potentially sensitive information (such as user/SP relationships and usage patterns), measures should be taken in order to avoid accidental or intentional leakage of this information to third parties. An open-source AS could help meet this goal, and would potentially attract greater trust from the security community than a closed one.

4.2 Threats to service providers

As explained in section 3.2, SPs trust the PRV-CA (and thereby the TPM/TP manufacturers, the TCPA conformance testing lab and the TCPA specification itself), the AS, and all system software executed before the AS. The TCPA architecture allows SPs to choose which PRV-CAs and software configurations to trust, thereby delegating most of the trust management to SPs themselves. Even so, vulnerabilities in system software, inadequate security procedures at PRV-CAs, and errors in the TCPA specification itself (and implementations thereof) pose threats to SPs, as they might allow impersonation of users. However, these threats do not just apply to the scheme described here; indeed they apply to any TCPA environment.

4.3 Threats to communication links

An (external) adversary that is in control of the communication link between the user system and an SP could launch the following attacks against the system.

Man-in-the-middle attack The adversary could forward the Integrity Challenge and authentication request message (step 1 in section 3.3) received from an SP as part of the SSO process to a victim user, while masquerading as the SP to that user (by spoofing the SPID). Forwarding the user's valid response (step 5 in section 3.3) to the SP might result in successful impersonation.

It is therefore of great importance to authenticate the origin of the Integrity Challenge and authentication request message (step 1 in section 3.3). This can be achieved, for example, using an SSL/TLS channel with server-side certificates[5] [18] in conjunction with the security extensions for DNS [6] or a suitable challenge/response protocol involving message signing [20].

The attack is prevented as long as the user inspects the SPID (step 2 in section 3.3), and makes sure that it indeed represents the desired SP.

As the authentication assertion contains the SPID and is digitally signed by the AS, intermediaries (such as an adversary) cannot change the SPID without being detected. At the same time the SP is provided with assurance that the assertion is indeed meant for this particular SP (and not any other).

Traffic analysis The adversary could eavesdrop on the messages exchanged between the user's TP and an SP. This could compromise the user's privacy as it reveals which SP the user is communicating with. The attack cannot be prevented by encrypting traffic (using SSL/TLS, for example), as traffic analysis can still be carried out by simple network monitoring. However, this threat exists regardless of the protocol being used between the SP and the user and is not addressed in the scheme described here.

4.4 Collusion threats

SP collusion If a number of SPs collude, they can compromise user privacy by correlating SSO Identities. Users can counter the attack by using different SSO Identities with different SPs. Ideally a new, dedicated SSO Identity should be used with every SP during initial registration.

However, a 'SP collusion' attack cannot be completely prevented as SPs may also be able to correlate users based on other profile information they may maintain (such as names or telephone numbers). "The only protection is for Principals [users] to be cautious when they choose service providers and understand their privacy policies" [11, p.71].

SP/Privacy CA collusion The PRV-CA can correlate Identity Credentials it has issued. Thus, if SPs collude with the PRV-CA, user privacy is compromised, even if a different SSO Identity is used with every SP. This is, of course, a property inherent in the TCPA architecture. Users can counter the attack by using

[5] Since the user requests a protected resource, it is likely that a SSL/TLS channel will be required anyway. Without protection of the communications channel between user and SP, initial user authentication is in any case of limited value to the SP.

different PRV-CAs to certify different IDKs. This may be a tradeoff between privacy protection and SSO, as it is likely that not all PRV-CAs that are trusted by the user are also trusted by all SPs (and vice versa).

5 Advantages and Disadvantages

Advantages of the SSO scheme described in this paper include the following.

- It is a local SSO scheme. This means that no third party can impersonate users, since the private keys of SSO identities are protected by the TPM of the local system.
- SSO identities are pseudonymous as they do not include any personal identifying information. This not only protects user privacy, but also allows for separation of identity roles. Further, no risks of personal information exposure arise at the SP.
- It does not necessarily require an online third party. SPs may, however, wish to periodically consult online Certificate Revocation Lists (CRLs) to check the status of the certificates used.
- The SSO protocol can be repeated whenever appropriate, without necessarily requiring user intervention. An online banking SP, for example, may wish to ensure that the user's authentication status and her TP's software state (including the AS) are still acceptable whenever access to a sensitive resource is requested. Rerunning the SSO protocol during an TP/SP session increases the achieved level of security without usability implications.
- The AS resides within the user's TP and its integrity is guaranteed by the trusted TPM. Therefore, and in contrast to other SSO schemes, it is hard to spoof the user interface without being detected. In other words, a 'bogus ASP' attack is likely to fail.
- The scheme does not require changes to the TCPA architecture.
- The scheme can potentially be adapted as a new Liberty [11] profile.

The scheme inherits certain potentially undesirable properties of the TCPA architecture. In particular the following points should be noted. How important these issues are will clearly depend on the implementation environment.

- The complexity of the system is quite high. The SPs must be able to verify Validation Certificates for every possible client software configuration. This aspect requires particular attention by implementers, as it potentially enables service denial attacks.
- PRV-CAs are able to compromise user privacy as they are able to correlate SSO Identities (see also section 4.4).
- Every TPM Identity is bound to its TP. This means that a particular SSO Identity can only be used on the particular TP it was created on. User mobility is only supported if the TP itself provides for this (such as TCPA-enabled laptops or PDAs), or if an SSO Identity federation service is provided by individual SPs.

6 Related Work

Single sign-on architectures within enterprise environments are examined in [3]. Open specifications for web-based SSO amongst disparate security domains include those of the Liberty Alliance [10] that use the SAML schema [16] to transport authentication assertions.

A comprehensive overview of the TCPA specification and potential usage scenarios is given in [1]. In particular, [1, p.255] proposes the use of TPs for sign-on in a corporate environment by delegating user authentication to the user TP and checking its authenticity and integrity. This paper extends these ideas to an open environment such as the Internet.

In the taxonomy of SSO systems described in [17], the scheme presented in this paper falls into the category of local true SSO schemes. An alternative 'local SSO' approach would be a special SSO application running on the users' PCs which stores their SP-specific authentication credentials and executes the SP-specific authentication mechanisms automatically whenever needed. Of course users would have to authenticate themselves to the SSO application at the beginning of a session. This approach is potentially transparent to the SPs, and, if implemented properly, may also offer cross-platform user mobility. Using a TCPA-conformant computing platform in conjunction with the SSO application offers certain advantages. For example, passwords may be retained in the TPM's protected storage, and only released if the platform has booted into a trusted state. Such a scheme falls into the category of pseudo-SSO schemes [17], since user authentication to SPs still relies on 'legacy methods'. This means that no guarantees can be made with respect to the pseudonymity, and therefore unlinkability, of credentials. Furthermore, if vulnerabilities exist in applications, such as web browsers, that may need to inter-operate with the SSO application, long-term credentials may become compromised without the TP being aware.

7 Conclusion

This paper demonstrates how SSO among disparate SPs can be achieved using TCPA-conformant computing platforms. The system design makes use of the TCPA security services of TPM Identities, Integrity Metrics and Key Certification. User authentication is delegated to the local TP. SPs need to check the authenticity of the user's TP and the integrity of its software state before trusting any authentication assertions.

The system protects user privacy through the use of pseudonymous SSO identities, and is independent of the local user authentication method. However, TCPA's dependence on PRV-CAs and its inherent complexity are inherited.

Acknowledgements

We would like to thank Liqun Chen for her helpful remarks and suggestions on an early version of this paper.

References

1. Boris Balacheff, Liqun Chen, Siani Pearson, David Plaquin, and Graeme Proudler. *Trusted Computing Platforms: TCPA Technology in Context*. Prentice-Hall, 2003.
2. T. Berners-Lee, R. Fielding, and L. Masinter. *Request For Comments 2396: Uniform Resource Identifiers (URI): Generic Syntax*. Internet Engineering Task Force, August 1998.
3. Jan De Clercq. Single sign-on architectures. In George I. Davida, Yair Frankel, and Owen Rees, editors, *Infrastructure Security, International Conference, InfraSec 2002, Bristol, UK, October 1-3, 2002, Proceedings*, volume 2437 of *Lecture Notes in Computer Science*, pages 40–58. Springer-Verlag, 2002.
4. Compaq Computer Corporation, Hewlett-Packard Company, IBM Corporation, Intel Corporation, Microsoft Corporation. *TCPA Main Specification v. 1.1b*, 2000-2002.
5. Computer Security Center of the Department of Defense, Fort George G. Meade, Maryland 20755. *Department of Defense Password Management Guideline*, April 1985. CSC-STD-002-85.
6. Donald Eastlake. *Request For Comments 2535: Domain Name System Security Extensions*. Internet Engineering Task Force, March 1999.
7. International Telecommunication Union. *ITU-T Recommendation X.509 (03/2000), Information technology — Open systems interconnection — The Directory — Public-key and attribute certificate frameworks*, 2000.
8. Liberty Alliance. *Liberty Architecture Glossary v.1.2-04*, April 2003.
9. Liberty Alliance. *Liberty Authentication Context Specification v.1.2-05*, April 2003.
10. Liberty Alliance. *Liberty ID-FF Architecture Overview v.1.2-03*, April 2003.
11. Liberty Alliance. *Liberty ID-FF Bindings and Profiles Specification v.1.2-08*, April 2003.
12. Liberty Alliance. *Liberty ID-FF Implementation Guidelines v.1.2-02*, April 2003.
13. Liberty Alliance. *Liberty ID-FF Protocols and Schema Specification v.1.2-08*, April 2003.
14. A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Boca Raton, Florida, 1997.
15. National Institute of Standards and Technology. *Federal Information Processing Standards Publication 180-1: Secure Hash Standard*, April 1995.
16. OASIS, http://www.oasis-open.org/committees/security/. *Security Services Technical Commitee Homepage*.
17. Andreas Pashalidis and Chris J. Mitchell. A taxonomy of single sign-on systems. In R. Safavi-Naini and J. Seberry, editors, *Information Security and Privacy, 8th Australasian Conference, ACISP 2003, Wollongong, Australia, July 9-11, 2003, Proceedings*, volume 2727 of *Lecture Notes in Computer Science*, pages 249–264. Springer-Verlag, Berlin, July 2003.
18. Eric Rescorla. *SSL and TLS*. Addison-Wesley, Reading, Massachusetts, 2001.
19. TCPA. *TCPA Frequently Asked Questions, Rev 5.0*, November 2002.
20. World Wide Web Consortium. *XML-Signature Syntax and Processing*, w3c recommendation edition, Feb 2002.