A Taxonomy of Single Sign-On Systems

Andreas Pashalidis** and Chris J. Mitchell

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

Abstract. At present, network users have to manage one set of authentication credentials (usually a username/password pair) for every service with which they are registered. Single Sign-On (SSO) has been proposed as a solution to the usability, security and management implications of this situation. Under SSO, users authenticate themselves only once and are logged into the services they subsequently use without further manual interaction. Several architectures for SSO have been developed, each with different properties and underlying infrastructures. This paper presents a taxonomy of these approaches and puts some of the SSO schemes, services and products into that context. This enables decisions about the design and selection of future approaches to SSO to be made within a more structured context; it also reveals some important differences in the security properties that can be provided by various approaches.

1 Introduction

Network users typically maintain a set of authentication credentials (usually a username/password pair) with every Service Provider[1] (SP) they are registered with. The number of such SPs with which a typical user routinely interacts has grown beyond the point at which most users can memorize the required credentials. The most common solution is for users to use the same password with every SP with which they register[2] — a tradeoff between security and usability in favour of the latter.

A potential solution for this security issue is Single Sign-On (SSO), a technique whereby the user authenticates him/herself only once and is automatically logged into SPs as necessary, without necessarily requiring further manual interaction. SSO thereby increases the usability of the network as a whole and at the same time centralises the management of relevant system parameters.

** 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 some kind of service or content to a user. Examples of SPs include web services, messenger services, FTP/web sites, and streaming media providers.

[2] There are even tools that 'synchronize' passwords, such as those at www.psynch.com and www.proginet.com/products/securpass/spsync.asp.

Several different SSO architectures have emerged, each with different properties and underlying infrastructures. In this paper we identify four generic architectures for SSO systems and discuss their strengths and weaknesses. The remainder of the paper is organised as follows. Section 2 presents the taxonomy, and section 3 discusses the main properties of the architectures. Section 4 then presents a variety of SSO schemes in the context of the taxonomy, and section 5 concludes the paper.

2 A taxonomy of SSO systems

As mentioned above, SSO systems support automatic user authentication to SPs. As authentication implies identification, SSO systems have to incorporate the life cycle management of the identifiers by which a user is known to the SPs he/she is registered with. These identifiers can take various forms. We refer to them collectively as 'SSO identities'.

We distinguish between two main types of SSO systems. The first type, which we call 'pseudo-SSO', involves use of a component which achieves SSO simply by automatically executing whatever authentication mechanisms are in place at the different SPs. At the beginning of a session the user has to authenticate him/herself to this pseudo-SSO component: we call this step *primary authentication*. One key distinguishing feature of this SSO type is that, during an 'SSO session', a separate authentication occurs every time the user is logged into an SP. The pseudo-SSO component manages the SP-specific authentication credentials, which constitute the SSO identities in this case. Since these SSO identities are thus SP-specific, the SSO Identity/SP relationship can be said to be $n : 1$; that is, any given SSO identity corresponds to exactly one SP, and a user may, in principle, have multiple SSO identities for a single SP. This type of SSO scheme is illustrated in Figure 1.

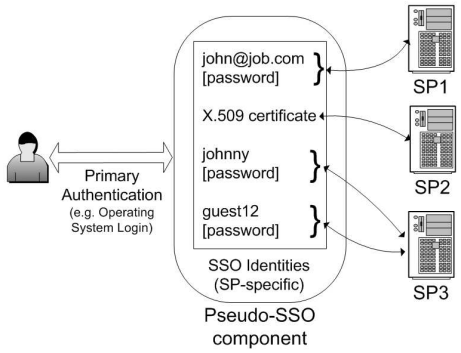

Fig. 1. Pseudo-SSO

Pseudo-SSO is fundamentally different from the second type of SSO system, which we call 'true SSO'. In a true SSO scheme the user authenticates to an entity called the *Authentication Service Provider* (ASP). The ASP is required to have an established relationship with all SPs to which SSO is to be achieved. This relationship requires a level of trust, and might typically be supported by a contractual arrangement. There would also typically need to be a supporting infrastructure to enable secure communications between ASPs and SPs. The key distinguishing feature of true SSO is that the only authentication process that involves the user occurs between the user and the ASP — SPs are notified of the authentication status of the user via so-called *authentication assertions*. These are statements that contain the user's SSO identity and his/her authentication status at the ASP. Note that the transport of these authentication assertions will itself need to be secured by a means specific to the SSO scheme in use. The exact form of SSO identities depends on the system design, but would typically be uniform throughout the whole system.

In contrast to pseudo-SSO, under true SSO the SSO identity/SP relationship can be $n : m$. That is, if supported by the scheme, not only can the user can potentially choose from a 'pool of identities' for any given SP, but the same SSO identity could, if the user wishes, be used with multiple SPs. This enables the assignment of specific *roles* to SSO identities (which then act as 'role pseudonyms' as defined in [19]). A true SSO scheme is depicted in Figure 2.

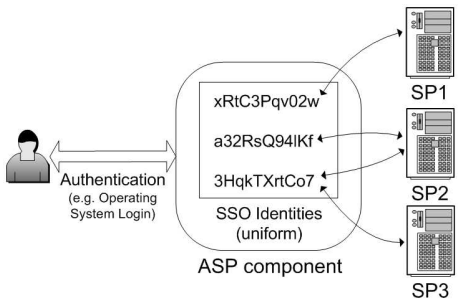

Fig. 2. True SSO

More generally, the information flows in an SSO system are shown in Figure 3. The user first authenticates to the ASP or pseudo-SSO component in step 1. Whenever a service is requested (step 2), the ASP or pseudo-SSO component automatically logs the user into the SP in step 3. Under true SSO this identity establishment phase involves the ASP (securely) conveying assertions about the user's authentication status to the SP, whilst in a pseudo-SSO scheme this step simply involves the pseudo-SSO component automatically performing the SP-specific (legacy) authentication mechanism on the user's behalf. Finally, assuming that step 3 is successfully completed, service provision proceeds in step 4.

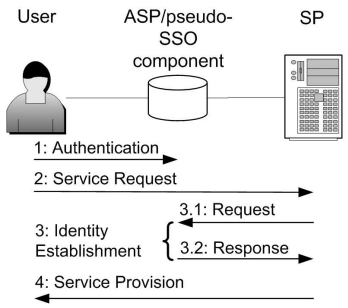

Fig. 3. Information flows in a generic SSO system

SSO architectures can be further categorised based on the location of the ASP/pseudo-SSO component; specifically, this component can either be local to the user platform, or offered as a service by an external entity, which we refer to as the 'SSO proxy'. We thus arrive at the following four main categories of SSO systems:

- Local pseudo-SSO systems,
- Proxy-based pseudo-SSO systems,
- Local true SSO systems, and
- Proxy-based true SSO systems.

We next consider each of these four types of scheme in a little more detail.

2.1 Local pseudo-SSO systems

In a local pseudo-SSO system, the pseudo-SSO component is resident within the user machine. This component maintains a (typically encrypted) database of the various authentication credentials for the different SPs. The user authenticates him/herself to the pseudo-SSO component at the beginning of a session. From that point on the component automatically executes SP-specific authentication protocols whenever needed, by supplying the (if necessary decrypted) authentication credentials. A key property of this SSO architecture is that the user machine needs access to cleartext versions of the long-term authentication credentials, and the user must therefore trust the machine not to compromise these credentials.

2.2 Proxy-based pseudo-SSO systems

In a proxy-based pseudo-SSO architecture, the pseudo-SSO component resides on an external proxy server; as in the local case, this external server has access to the user's credentials, and hence must be trusted for this purpose by the user. Primary authentication occurs between the user and the proxy at the beginning of a session (and possibly thereafter if the proxy wishes to perform

a re-authentication). Subsequent user authentication at SPs is redirected to, or intercepted by, the proxy, which automatically executes the SP-specific authentication protocol including supplying the requested credentials. A key property of this architecture is that the local machine never has access to the user's SP-specific credentials; authentication to SPs occurs directly between the proxy and the SPs.

2.3 Local true SSO systems

As explained above, under true SSO the ASP authenticates the user, and subsequently conveys authentication assertions to relying SPs whenever necessary. When a trusted component within the user system takes the role of the ASP, the resulting architecture falls into the category of local true SSO systems. Of course a trust relationship between the local ASP, the relying SPs, and a supporting security infrastructure must exist. Since, in this setting, the ASP is under the physical control of the user, mechanisms must be in place that guarantee the integrity and trustworthiness of the ASP component itself.

2.4 Proxy-based true SSO systems

In a proxy-based true SSO architecture, an external server takes the role of the ASP. This external server acts as a broker between users and SPs; registered users can benefit from SSO at SPs that maintain a trust relationship with the ASP. It is worth noting that the ASP can trivially impersonate any registered user at every relying SP, simply by conveying an assertion. Therefore, both users and SPs have to trust the ASP for the purposes of SSO. Note also that this observation holds for the ASP in all (local or proxy-based) true SSO systems, as well as for the pseudo-SSO component in all pseudo-SSO systems.

3 Properties of SSO schemes

This section presents some properties that distinguish the various types of SSO system. The four SSO categories are examined with respect to these properties. Examples of SSO implementations are also mentioned wherever appropriate.

3.1 Privacy protection

Privacy protection is of particular importance in open environments [23]. With respect to SSO systems, privacy concerns arise if SSO identities contain personally identifying information, or if it is possible for an attacker (which could be one or more colluding SPs) to correlate distinct identities of the same user without his/her consent. Thus SSO identity pseudonymity, in the sense that they do not include any personally identifying information, and unlinkability are potentially desirable properties, where the latter implies the former [19].

Pseudo-SSO systems cannot guarantee the pseudonymity and therefore cannot guarantee the unlinkability of SSO identities, because the identities are SP-specific; some SPs may require users to log in using their e-mail addresses (as is the case with 'anonymous' FTP sites and some web sites), public key certificates, or their social security numbers. True SSO systems, on the other hand, can be designed such that SSO identities do not contain any personally identifying information and remain unlinkable even if SPs collude.

Of course, the success of privacy protection assumes that unlinkable means of payment exist (where users need to pay for services). Whilst many such schemes have been proposed, they have yet to make a major impact in the commercial environment.

3.2 Anonymous network access

Whether or not otherwise unlinkable SSO identities remain secure when an attacker has access to network address information depends on the wider context within which the SSO scheme is deployed. If, for example, the lower layer protocols do not provide anonymous network access, an attacker can easily compromise unlinkability by correlating SSO identities using information found in packet headers or traffic analysis [1]. Unfortunately, the user's network address (which *is* typically included in packet headers) also reveals the identity and geographical location of his/her network access provider. The provider can then help link the network address (and thereby SSO identities) back to the user's real identity.

These privacy issues are addressed by schemes that provide anonymous network access. Such schemes are typically implemented through an externally operated 'anonymising proxy'. All traffic between the user and the SPs is physically routed through that proxy, which replaces the user's real network address (and perhaps other identifying information) with its own. The achieved level of anonymity depends, of course, on the number of users in the system [19]. Examples of such anonymity services are the Anonymizer (`www.anonymizer.com`) and Freedom WebSecure[3] from zerøknowledge.

Such single-proxy schemes do not protect against traffic analysis, and all trust resides within the proxy operator. There exist more powerful schemes, such as those described in [4, 7], that do protect against traffic analysis and ideally distribute the trust amongst disparate security domains. An example of such a system is JAP (`anon.inf.tu-dresden.de`).

Anonymising services can be employed in conjunction with SSO systems in order to increase the level of SSO identity unlinkability. In fact, SSO-proxies could be augmented with the functionality of an anonymising proxy. Of course all traffic would have to be physically routed through the proxy, but the user would not have to trust an additional entity. In the case of proxy-based pseudo-SSO, the architecture might be completely transparent to SPs: they would not even need to be aware of the proxy's existence.

[3] www.freedom.net/products/websecure

3.3 User mobility

Proxy-based SSO architectures inherently support user mobility; users can authenticate themselves to the ASP/pseudo-SSO component from anywhere in the network (except, of course, if the authentication method itself imposes restrictions on this). We thus concentrate on how user mobility can be supported in local SSO schemes.

In local pseudo-SSO systems, user mobility can be supported if the credential database is held on an external server: initial authentication between user and server occurs once at the beginning of a session and credentials are then downloaded to the local machine as needed. The degree of trust in the server varies according to whether or not credentials are stored in an encrypted form. Novell's SecureLogin[4], Passlogix' V-GO (`www.passlogix.com/sso`) and Protocom's SecureLogin (`www.protocom.cc`) products are examples of local pseudo-SSO systems with support for user mobility. One can also regard automatic form-fillers as products with pseudo-SSO functionality. Examples include the automatic form completion functions of popular web browsers and Novell's DigitalMe (`www.digitalme.com`) service.

However, the local true SSO scheme described in [18] does not support cross-platform user mobility per se, as SSO identities are bound to the user platform itself (see section 4.5).

3.4 Use in an untrusted environment

Some scenarios require users to access SPs from untrusted or hostile environments, such as Internet cafés or public terminals. In this situation it is undesirable if the untrusted machine ever has access to authentication credentials that will allow it to later launch successful impersonation attacks.

In such scenarios proxy-based SSO schemes might prove useful. Of course, the initial authentication between user and proxy must have the property that observation of one authentication exchange does not enable subsequent impersonation of the user, as in, for example, a one-time password scheme or a suitable challenge/response protocol (see, for example, [16]). Building on this, and assuming that all network traffic between the user and the SPs is physically routed through it, the proxy may also provide an additional privacy protection service by 'stripping off' all personal data before it reaches the untrusted machine.

3.5 Deployment and maintenance costs

Generally speaking, it is far less costly to deploy pseudo-SSO schemes, because they do not require any common security infrastructure; moreover, existing SPs may not need to change at all. On the other hand, if any SP *does* change its user authentication mechanisms after deployment, this has to be reflected in the pseudo-SSO component. This increases the maintenance costs, especially in dynamically changing environments.

[4] www.novell.com/products/securelogin

The situation is reversed in true SSO-schemes. Deployment of true SSO systems requires a costly, system-wide security framework (such as a Public Key Infrastructure) to support SP-ASP secure communications, which might involve service level and liability transfer agreements. Once the infrastructure is in place, however, maintenance costs are likely to be small, since changes in the authentication interface occur only between the user and his/her chosen ASP(s).

3.6 Running costs

The running costs of local SSO schemes are likely to be lower than those of proxy-based systems. This is because local SSO schemes do not require the continuous online presence of a server.

Proxy-based systems depend, of course, on the security and availability of the external proxy server. The server constitutes a single point of failure in the system, and must be protected against service denial attacks. Moreover additional communications costs might be incurred if all traffic is physically routed through the proxy. Further, in proxy-based true SSO systems, SPs may be charged by the ASP for the authentication service.

3.7 Trust relationships

As explained in section 2.4, under both pseudo and true SSO, users and SPs need to trust the Pseudo-SSO component/ASP for the purposes of SSO. A local SSO system, such as the one described in [18], has the advantage that the user does not have to trust an externally operated entity, although the user will still need to trust the integrity of the software providing the SSO functionality.

It is important to note that, despite the universal need for user trust in the SSO component, there remain differences in the nature of the trust relationships involved in true and pseudo-SSO schemes. In the case of a true SSO scheme, the common security infrastructure allows for the trust relationships between the SPs and ASPs to be precisely described and regulated by policies, service level and liability transfer agreements, and other means outside the scope of SSO in the technical sense. Furthermore, if the user's authentication credentials are compromised, SSO can be disabled centrally.

Under pseudo-SSO the trust relationships are more diffuse. SPs may not even be aware that a SSO scheme is in place. Credential databases may be encrypted and replicated at several servers. The trust relationship between the user, any servers and the pseudo-SSO component depends on the implementation details of the scheme. The trust relationship between pseudo-SSO component and SPs may be different for each SP, and may change whenever individual SPs modify their authentication interfaces.

3.8 Conflict resolution and lawful access

In the event of a dispute or lawful investigation, the operator of the proxy in a proxy-based (true or pseudo) SSO scheme may provide evidence of events, as a

trusted third party, by keeping logs of authentications. The situation is likely to be better defined in a true SSO system, as in such systems there is necessarily a well-defined relationship between ASPs and SPs.

Local SSO schemes are much less likely to be useful in this sense, since evidence can either be deleted, e.g. if it might be embarrassing to the user, or modified. However, if the ASP in a local true SSO system incorporates physical protection measures, which enables it to be trusted by third parties (see, for example, [18]), then locally stored event logs may still possess evidential value.

3.9 Open versus closed environments

The issue of privacy protection (which includes anonymous network access and use in untrusted environments) is usually deemed to be of less importance in closed environments. Thus, the main focus for SSO for closed systems is likely to be the deployment, running and maintenance costs. Since these are less for pseudo-SSO systems, especially in relatively stable environments, 'enterprise' pseudo-SSO solutions promise a rapid and concrete return on investment. Architectures of such systems are examined in [6], and examples of real-world implementations include Computer Associates' eTrust Single Sign-On[5] and Evidian's AccessMaster (`www.evidian.com/accessmaster`).

However, the need for privacy protection in open environments may well outweigh the deployment cost of true SSO schemes. Thus it seems likely that true SSO systems will eventually be required in open environments such as the Internet. Some well-known proxy-based true SSO systems designed for open environments are discussed in the next section.

4 Some examples of SSO schemes

This section provides more detailed descriptions of certain SSO schemes. First note that the preponderance of existing discussed and deployed SSO schemes fall into two of the four categories discussed above, namely local pseudo-SSO and proxy-based true SSO schemes (examples of the former class have already been mentioned in section 3.3). Because of this, in this section we first discuss three important examples of the latter class. However, to show that other possibilities exist, we also briefly discuss two further examples, namely of a proxy-based pseudo-SSO scheme and a local true SSO scheme.

4.1 Kerberos

Although Kerberos is formally described as a 'network authentication system' [8, 22], one can regard it as a SSO scheme. A single security domain, or *realm*, consists of a set of users, an Authentication Server, a 'Ticket Granting Server' and a set of relying SPs. The Authentication Server and Ticket Granting Server

[5] www3.ca.com/Solutions/Product.asp?ID=166

can be combined into a single entity called the 'Kerberos server' which acts as the ASP. The security infrastructure of Kerberos relies solely on symmetric cryptography; every user and SP shares a long-term secret key with the ASP.

A simplified description of SSO under Kerberos follows. The protocol is executed whenever the uses wishes to log into an SP of the realm.

1. If the user already possesses a valid 'Ticket Granting Ticket' (TGT) from a previous protocol run, this step is skipped. Otherwise, the user requests a fresh TGT from the ASP. The ASP replies with a message that contains a fresh TGT, and a 'session key' which will be used to construct an 'authenticator', i.e. a data structure which is encrypted under a session key and contains elements that protect against replay attacks. This session key is encrypted under the long-term key the user shares with the ASP (or a key derived from it). The user decrypts the session key by supplying his/her long-term key.
2. The user sends a message to the ASP that contains the TGT, an authenticator (encrypted under the aforementioned session key), and the identifier of the SP he/she wishes to access. The ASP checks the validity of the received message. If not satisfied, (if, for example, the TGT has expired) authentication fails. Otherwise the user is now deemed authenticated at the ASP.
3. The ASP replies with a message that contains a 'Service Granting Ticket' (SGT) a data structure encrypted under the key shared by the ASP and the SP in question, and a second session key which is encrypted under the session key of step 1.
4. The user now constructs a message containing the SGT and an authenticator encrypted under the second session key. This message, if constructed correctly, demonstrates the user's ability to decrypt the second session key and can be regarded as an authentication assertion. The user sends it to the SP which decrypts it and, if valid, logs the user in.

The tickets (TGT and SGT) are encrypted data structures that contain (among other things) the user identifier and network address, the server identifier, corresponding session keys and expiration timestamps. SSO is achieved by the fact that the user does not need to reenter his/her long-term key as long as the TGT remains valid.

SSO among multiple realms is achieved by setting up the required relationships and symmetric keys between the involved Kerberos servers. There is no restriction as to the type (web, FTP, etc.) of SPs that may rely on Kerberos for user authentication as long as they follow the protocol.

Since the same user identifier is used with every SP, the SSO identity/SP relationship under Kerberos is $1 : n$. Thus, unlinkability of SSO identities is not an issue. It is interesting to observe that even distinct Kerberos accounts of a given user are still linkable as tickets bind them to the user's network address. Since the authentication mechanism is based on a long-term secret key, user mobility can be supported (if the key is derived from a password, for example) but it is not suitable for use in an untrusted environment.

4.2 The Liberty Alliance

The Liberty Alliance (`www.projectliberty.org`), a consortium of over 140 member companies, recently developed a set of open specifications for web-based SSO. In Liberty terminology [12], the ASP and the user are the 'Identity Provider' and 'Principal' respectively. The specifications use the Security Assertions Markup Language (SAML), a platform-independent framework for exchanging authentication and authorisation information[6]. Liberty is based on the notion of 'trust circles' which are formed by trusted ASPs and sets of relying SPs. The relationship between ASP and SPs has to be supported by contractual agreements outside the scope of the specifications.

According to the specifications, users first authenticate themselves to the ASP, which subsequently conveys authentication assertions to the relying SPs in order to facilitate SSO. The assertions contain 'name identifiers' that allow SPs to differentiate between users. For any given user, the ASP has to use a distinct identifier with each SP of the trust circle. Thus, under Liberty, the SSO identity/SP relationship is $1:1$. Furthermore, name identifiers "must be constructed using pseudo-random values that have no discernible correspondence with the Principals identifier (e.g. username) at the Identity Provider [ASP]" [15, p.12]; SSO identities are pseudonymous and, by themselves, unlinkable. Unlinkability, however, can be compromised, as SPs may be able to correlate SSO identities based on the users' network addresses. Use of an anonymous network access scheme (section 3.2) could help with this. Profile information that individual SPs may maintain (such as shopping habits, telephone numbers or credit card details) can also compromise unlinkability. For the time being, "the only protection is for Principals to be cautious when they choose service providers and understand their privacy policies" [14, p.70]. An independent assessment of the specifications with respect to privacy appears in [20].

The Liberty specifications are authentication method neutral; the details of the particular method employed are explicitly stated in the authentication assertions [13]. This means that, under a suitable user authentication mechanism, user mobility or even use in an untrusted environment can be supported.

The Liberty Protocols and Schema Specification [15] defines 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*. All currently specified profiles rely on the Secure Socket Layer (SSL) or the Transport Layer Security (TLS) [21] protocol in order to provide secure channels between parties. Hence, a Public Key Infrastructure (PKI) must be in place. A separate PKI may be required if a profile is used that requires assertions to be digitally signed. Authentication assertions sent from the ASP to the SP are routed through the user browser via web redirects; in the 'browser/POST' profile, for example, assertions are sent within an HTTP form while in the 'browser/artifact' profile an 'artifact' is encoded in the URL that the SP can later resolve into an assertion [14].

[6] www.oasis-open.org/committees/security

According to [10], the next version of Liberty will include a framework for permissions-based attribute sharing which will allow organisations of a given trust circle to be linked together, as opposed to operating independently. Thus, the specifications are targeted towards a more comprehensive treatment of Identity Management that extends beyond the provision of SSO.

4.3 Microsoft Passport

Microsoft Passport (`www.passport.com`) is a web-based SSO service offered by Microsoft since 1999. The passport server acts as the ASP. Users register with the ASP by supplying a valid e-mail address and a password (or, if they register from a mobile phone, their phone number and a Personal Identification Number). Additional profile information, such as address, date of birth and credit card details, may also be stored in their passport accounts. Every account is uniquely identified by a 64-bit number called the 'Passport User ID' (PUID). SPs that wish to offer SSO for registered passport users need to sign a contractual agreement with Microsoft (which involves a yearly provisioning fee of $10,000 [17]), implement a special component in their web server software, and share a secret key with the ASP. Since SSL/TLS channels are required between the user and the passport server (and optionally between user and SP), an appropriate PKI must also be in place.

SSO is achieved by the following protocol, which is executed whenever the user wishes to log into an SP.

1. The user's browser is redirected to the ASP.
2. The ASP tries to retrieve a 'Ticket Granting Cookie' (TGC) from the web browser's cookie cache. If one is found, if it decrypts successfully, and if it is valid, the user is deemed authenticated and the rest of this step is skipped. Otherwise, the ASP requests the user to authenticate him/herself. Assuming successful authentication, the ASP saves a fresh TGC in the browser's cookie cache. This cookie is encrypted under a 'master key' only known to the ASP. Its function is similar to the TGT of Kerberos; there is, however, no 'authenticator' in Passport — replaying a stolen TGC results in successful impersonation.
3. The ASP saves a set of cookies in the browser's cookie cache which includes the user's PUID and other profile information the user has consented to share at registration time. This cookie set is encrypted under the secret key shared between the ASP and the SP in question. Its functionality is similar to Kerberos' SGT and acts as an authentication assertion.
4. The user's browser is redirected back to the desired SP which reads and decrypts the aforementioned cookie set and, if satisfied, logs in the user.

SSO is achieved by the fact that, as long as the TGC remains valid, the user does not need to re-authenticate (in step 1) in subsequent protocol runs. As the authentication method is password-based, user mobility is supported, but it is not suitable for use in an untrusted environment. Passport, like Kerberos, uses a

single SSO identity (the PUID) with every SP. The SSO identity/SP relationship is therefore $1 : n$ and unlinkability is not an issue.

According to [11], in the future Passport will be based on Kerberos.

4.4 An outline of a proxy-based pseudo-SSO scheme

One alternative to the true SSO schemes described above would involve a proxy acting as a gateway to service providers. The user would first authenticate to the proxy, e.g. using a one-time password scheme. The user would connect to SPs via the proxy, which would transparently 'fill in' identifiers and passwords for specific SPs, using information from its credential database.

Such an approach would have several major advantages. It would enable access to services from an untrusted PC since SP-specific credentials would not reach the user PC (see section 3.3). Like other pseudo-SSO schemes it could be deployed transparently, i.e. without requiring individual SPs to make any changes to their authentication procedures. Moreover user mobility is supported. However, like other pseudo-SSO schemes, unlinkability of SSO identities cannot be guaranteed.

4.5 A local true SSO scheme built upon TCPA

The final example scheme we consider is described in much greater detail in [18]. The Trusted Computing Platform Alliance (TCPA) is an industry consortium established to agree specifications for ways in which to include a trusted component within a computing platform. The scheme shows how to use a TCPA-compliant platform to establish a local true SSO system.

In this scheme a component within the user's platform, which we call the Authentication Service (AS), acts as the ASP; it authenticates the user and subsequently conveys assertions to SPs whenever necessary. The integrity of the user platform's software state (which includes the AS) is measured by a TCPA function called 'Integrity Metrics'. Any given SP reliably acquires these software metrics through an 'Integrity Challenge/Response' session, also specified by TCPA. This step requires a PKI to be in place. The SP then needs to verify that the given metrics represent a software configuration that is trusted for the purposes of SSO. This step includes verification of certificates supplied by the manufacturer(s) of software (including the AS) running on the user's platform.

The scheme uses TCPA-specified credentials, called 'Identity Credentials', as SSO identities. An Identity Credential is a certificate issued by a special authority and certifies that the specified platform conforms to the TCPA specifications (without, however, uniquely identifying the particular platform). Any given user can acquire an arbitrary number of such credentials for his/her platform. As these Identity Credentials act as SSO identities, SPs also need to verify them — a function which requires another PKI to be in place.

The scheme offers a $n : m$ relationship between SSO identities and SPs, allowing the assignment of roles to SSO identities. User mobility, however, is not supported per se, as SSO identities are bound to the user's platform.

5 Conclusions

We have presented an abstract taxonomy of SSO systems and have analysed the properties of the identified four main types of SSO scheme. The characteristics of these four types of scheme are summarised in Table 1. We have also presented some examples of SSO schemes in the light of the taxonomy. Unfortunately, (with the exception of the recently proposed scheme described in section 4.5) none of the true SSO schemes offers a $n : m$ SSO identity/SP relationship and therefore cannot be used to assign roles to SSO identities; a separate account has to be created at the ASP for every role. This is clearly a topic meriting further study.

Table 1. Properties of SSO systems.

	Local pseudo-SSO	*Proxy-based pseudo-SSO*	*Local true-SSO*	*Proxy-based true SSO*
Pseudonymity and Unlinkability	cannot be guaranteed	cannot be guaranteed	can be guaranteed	can be guaranteed
Anonymous Network Access	needs additional services	can be integrated	needs additional services	can be integrated
Support for User Mobility	needs additional services	under suitable authentication method	needs additional services	under suitable authentication method
Use in Untrusted Environment	not supported	under suitable authentication method	not supported	under suitable authentication method
Deployment Costs	low	low	high	high
Maintenance Costs	potentially high	potentially high	low	low
Running Costs	low	high	low	high
Trust Relationships	dynamically changing	dynamically changing	concrete and consistent	concrete and consistent

It is clear that each SSO architecture has its strengths and weaknesses, and one must carefully consider the environment before opting for a particular SSO solution. Generally speaking, pseudo-SSO schemes are probably more suitable for closed systems, where privacy protection is less of an issue. Identity Management in that context just refers to the management of the life cycle of the credentials the user maintains within the closed system. On the other hand, Identity Management in open environments like the Internet, may well need to incorporate privacy protection. SSO schemes, privacy protection services (such as the ones discussed in sections 3.2 and 3.4) and privacy-aware Identity Management schemes, such as the ones described in [2, 5, 9], could be integrated into a true SSO scheme. Indeed, it would appear that a precise indication of the

achieved privacy level is possible only with the combined use of a suitable *true* SSO scheme. A practical scheme that could deliver such an indication would potentially be useful. Moreover, it would be interesting to see how anonymous credential systems, such as the one described in [3], could be combined with authentication method-neutral SSO (possibly along with anonymous network access).

References

1. Adam Back, Ulf Möller, and Anton Stiglic. Traffic analysis attacks and trade-offs in anonymity providing systems. In I. S. Moskowitz, editor, *Information Hiding, 4th International Workshop, IHW 2001*, volume 2137 of *Lecture Notes in Computer Science*, pages 245–257. Springer Verlag, Berlin, 2001.
2. Oliver Berthold and Marit Köhntopp. Identity management based on P3P. In H. Federrath, editor, *Designing Privacy Enhancing Technologies, International Workshop on Design Issues in Anonymity and Unobservability, July 2000*, number 2009 in Lecture Notes in Computer Science, pages 141–160. Springer-Verlag, Berlin, 2001.
3. Jan Camenisch and Els Van Herreweghen. Design and implementation of the idemix anonymous credential system. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 21–30. ACM Press, New York, 2002.
4. David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.
5. Sebastian Clauß and Marit Köhntopp. Identity management and its support of multilateral security. *Computer Networks*, 37:205–219, 2001.
6. 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.
7. David M. Goldschlag, Michael G. Reed, and Paul F. Syverson. Onion routing for anonymous and private internet connections. *Communications of the ACM*, 42(2):84–88, January 1999.
8. Internet Engineering Task Force. *RFC 1510: The Kerberos Network Authentication Service (V5)*, September 1993.
9. Uwe Jendricke and Daniela Gerd tom Markotten. Usability meets security — the Identity-Manager as your personal security assistant for the internet. In *Proceedings of the 16th Annual Computer Security Applications Conference (ACSAC 2000)*, pages 344–355. IEEE Computer Society, 2000.
10. Liberty Alliance. *The Liberty Alliance News Letter*, volume 1, issue 1 edition, November 2002.
11. Liberty Alliance. *Identity Systems and Liberty Specification version 1.1 Interoperability*, January 2003.
12. Liberty Alliance. *Liberty Architecture Glossary v.1.1*, January 2003.
13. Liberty Alliance. *Liberty Authentication Context Specification v.1.1*, January 2003.
14. Liberty Alliance. *Liberty Bindings and Profiles Specification v.1.1*, January 2003.
15. Liberty Alliance. *Liberty Protocols and Schemas Specification v.1.1*, January 2003.
16. A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Boca Raton, 1997.

17. Microsoft. *Microsoft .NET Passport Review Guide*, November 2002.
18. Andreas Pashalidis and Chris J. Mitchell. Single sign-on using trusted platforms. Technical Report RHUL–MA–2003–3, Mathematics Department, Royal Holloway, University of London, March 2003.
19. Andreas Pfitzmann and Marit Köhntopp. Anonymity, unobservability, and pseudonymity — a proposal for terminology. In H. Federrath, editor, *Designing Privacy Enhancing Technologies, International Workshop on Design Issues in Anonymity and Unobservability, July 2000*, number 2009 in Lecture Notes in Computer Science, pages 141–160. Springer-Verlag, Berlin, 2001.
20. Birgit Pfitzmann. Privacy in enterprise identity federation — Policies for Liberty single signon. In *Proceedings: 3rd Workshop on Privacy Enhancing Technologies (PET 2003), Dresden, March 2003*, Lecture Notes in Computer Science. Springer-Verlag, Berlin, to appear.
21. Eric Rescorla. *SSL and TLS*. Addison-Wesley, Reading, Massachusetts, 2001.
22. J. G. Steiner, B. Clifford Neuman, and J.I. Schiller. Kerberos: An authentication service for open network systems. In *Proceedings of the Winter 1988 Usenix Conference*, pages 191–201, February 1988.
23. World Wide Web Consortium. *The Platform for Privacy Preferences 1.0 (P3P 1.0) Specification*, April 2002.