Federated Identity Management Systems: A Privacy-based Characterization*

Eleanor Birrell   Fred B. Schneider

Department of Computer Science
Cornell University
Ithaca, NY 14853
{eleanor, fbs}@cs.cornell.edu

Abstract

Identity management systems store attributes associated with users and facilitate authorization on the basis of these attributes. A privacy-driven characterization of the principal design choices for identity management systems is given, and existing systems are fit into this framework. The taxonomy of design choices also can guide public policy relating to identity management, which is illustrated using the United States NSTIC initiative.

1 Introduction

People today employ the Internet to manage finances, access employer resources, shop, and communicate. Each activity involves interacting with a *service provider*. Such interactions typically require a digital *identity* to be associated with the *user*. Identities today are, for the most part, stored and managed by each service provider. They are used to improve the user's experience, to increase the service provider's profits, or to defend against certain attacks.

Prior to the introduction of *identity management systems*, there was a broad consensus that practices concerning online identities were problematic.

(1) Each service provider maintained a set of user identities, so users had many identities (one for each service provider with which they interacted). This introduced a management burden and generated many potential points of failure.

(2) Users were not given control over attribute dissemination, leading to privacy violations and identity theft.

*Birrell is supported by a National Science Foundation Graduate Research Fellowship. Schneider is supported in part by AFOSR grants F9550-06-0019 and FA9550-11-1-0137, National Science Foundation grants 0430161, 0964409, and CCF-0424422 (TRUST), ONR grants N00014-01-1-0968 and N00014-09-1-0652, and grants from Microsoft.

Over the last fifteen years, identity management systems have evolved with the goal of addressing these problems, although mitigation of security concerns has been limited by the deployment of such systems. Identity management systems provide enhanced security and convenience by introducing a new party—an *identity provider*—that must be trusted to perform certain functions. User authentication and identity management are delegated to the identity provider, which implements mechanisms that allow users to control attribute release and mechanisms that issue *authentication assertions*—statements about user attributes. Upon receiving an *authorization request* from software executing on behalf of a user, a service provider decides whether to *authorize* a user on the basis of authentication assertions. (A detailed overview of the primary components and parties in an identity management system is given in Section 2.)

Existing work on identity management has focused primarily on individual systems. These systems[1] each focus on one of three general types of functionality:

- *Single Sign-on*: Systems designed to issue authentication assertions to multiple service providers after a single user authentication. Examples include Passport[2] [25], OpenID[3] [29], Shibboleth [34], and Facebook Single Sign-On [13].

- *Federated Identity*: Systems designed to manage multiple distinct identities for a single user and to issue authentication assertions on the basis of any of these identities. Examples include Project Liberty[4] [33], Higgins [15], PRIME [32], and CardSpace [10].

- *Anonymous Credentials*: Systems designed to provide authentication assertions that do not reveal the user's identity to the service provider. Examples include Idemix [6, 7], U-Prove [30], and P-IMS [17, 35].

In this paper, we focus on identifying key design choices that are intrinsic to identity management systems, and we explore connections between these design choices, as well as discussing the impact of these choices on system functionality. We adopt a privacy-driven approach centered around three privacy properties:

- *Undetectability* [31, 11]: Conceal user actions.

- *Unlinkability* [31, 11]: Conceal correlations between actions and/or identities (e.g., untraceability).

- *Confidentiality* [31, 11]: Enable user control over the dissemination of user attributes.

Undetectability, Unlinkability, and Confidentiality clearly are related problems (since all concern which parties have access to particular data). However, as we show below, these properties depend on orthogonal design choices and thus are best considered independently.

[1]Detailed overviews of each of the identity management systems appear in Appendix A.

[2]Microsoft's Passport is no longer supported, but this system is included for historical reasons.

[3]Although our primary focus is on systems and implementations, we discuss the OpenID specification because it has no representative implementation.

[4]Project Liberty was merged into SAML 2.0 (the specification implemented by Shibboleth and various enterprise identity solutions) and is no longer actively maintained. It is included for historical reasons, because there are no existing representative implementations.

These three properties, therefore, define a privacy-focused design space that informs choices intrinsic in building and deploying identity management systems.[5]

In Sections 3–5, we discuss how Undetectability, Unlinkability, and Confidentiality are each supported by various design choices. We also show ways in which successful instantiation of these privacy properties depends not only on system design but also on the manner in which service providers interact with an identity management system.

Section 7 illustrates the analytic value of our privacy-driven characterization by demonstrating how the privacy principles (and the design choices they define) might influence the development of future identity management systems. We focus on the National Strategy for Trusted Identities in Cyberspace (NSTIC), and we argue that some of the design choices that NSTIC advocates should be re-evaluated in light of the connections between privacy principles, design choices, and system functionality that are identified in this work.

This paper focuses primarily on the impact of privacy properties in driving the design of identity management systems, but many other factors affect both the implementation and the success of an identity management system. Economic incentives and user interface design, for example, may trump privacy concerns for some system designers. Still, privacy principles constitute an interesting lens through which to view the motivating principles that inform the design of identity management systems, since policy makers and users voice concerns about privacy. Works that adopt alternative approaches, including exploring economic incentives behind the design of identity management systems, are discussed briefly in Section 6.

2 System Components

The *parties* in an identity management system are users, identity providers, and service providers.[6]

2.1 Users

Each user (sometimes called a *subject* or *principal*) is associated with a person. A user U is characterized by an identity—a collection of attributes that represent properties about U. Attributes describe inherent qualities (e.g., $U.age$ = *25*), circumstances (e.g., $U.employer$ = *Example Co.*), behaviors (e.g., $U.shops\_at\_Amazon$ = *true*), inclinations (e.g., $U.likes\_balloon\_animals$ = *true*), or arbitrarily assigned values (e.g., $U.uid = 124$). There is no restriction on the number of attributes that comprise an identity; some identities are small (e.g., just a username and password) yet others may contain many (possibly inter-dependent) attributes. Identities can be created by the user or can be created by another party (e.g., an employer) acting on behalf of the user. Compound notions of identity (for example, the intersection of existing identities or a delegated identity) are not supported by existing identity management systems.

[5]These discussions are intended to be accessible to a reader with a background in systems security but no specialized knowledge of identity management, although it is expected that readers with relevant expertise will benefit from references to historical and current systems and specifications.

[6]We treat each party as monolithic, even though parties could be implemented by distributed systems. Such implementation details, which could affect security and reliability, are orthogonal to our analysis.

A single user can be associated with multiple identities. Identities are sometimes compared to cards in a wallet [10], because people interacting in the physical world choose from many different identifying cards (e.g., driver's license, credit cards, employer id, library card) for each given activity. Identity management systems enable users to select among multiple distinct digital identities in an analogous manner; the choice of identity can determine which attributes are disseminated and, in particular, can prevent the dissemination of extraneous attributes.

2.2 Service Providers

Service providers authorize users on the basis of authentication assertions. These authorization decisions may depend on the attributes received, the format of the assertion, or properties of the party P that issued the authentication assertion.

Currently, most service providers (e.g., Amazon, New York Times) implement their own identity management. Users are thus responsible for managing a separate identity for each service provider with which they interact. Many users—seeking convenience—simply reuse the same *authentication credentials* (e.g., username and password) with multiple different service providers. A service provider that accepts U's authentication credentials is, therefore, trusting every other service provider in the system not to impersonate users and not to release user authentication credentials (voluntarily or involuntarily).

Were identity management systems in place, service providers would have the flexibility to choose which parties to trust (that is, which authentication assertions to accept) and, because authentication assertions would be managed exclusively by the identity providers, service providers would no longer need to trust other service providers.

2.3 Identity Providers

An identity provider, which can be implemented as a stand-alone party or as a component of a user or service provider, performs two primary functions in an identity management system.

(1) It is responsible for authenticating users—i.e., determining whether a particular user is associated with a particular identity—and issuing authentication assertions in support of authenticated users. How an identity provider authenticates users could affect whether service providers accept authentication assertions issued by that identity provider.

(2) It stores collections of attributes for users and is responsible for managing these identities. Details vary across systems but, generally, an identity provider would have provisions for creating, updating, releasing, and deleting attributes and identities.

Since service providers depend on receiving authentication assertions that contain current attributes, an identity provider might not only be responsible for validating attributes initially—that is, verifying their value in connection to a real-world identity—but could also be trusted to maintain currency of those attributes. The methods used for validating attributes (whether to trust the user or whether to perform independent validation of claimed attributes) and for eliminating or updating stale attributes often determine whether a service provider will accept authentication assertions issued by some identity provider.[7]

2.4 Threat Model

Privacy properties are defined with respect to some assumptions about adversaries. These adversaries could be parties inside the system (e.g., an adversarial identity provider), or they could be external parties with access to some or all of the information available in the system (e.g., hackers who target identity provider databases, phishers, or snoopers looking over a user's shoulder). Adversaries might attempt to learn about user attributes or past actions through any available means, including forging messages or exploiting transport-layer addressing and timing. Multiple adversaries can collude with each other, and they may exchange information or messages outside the scope of the defined protocols.

A party A that trusts a party B not only will believe that B is non-adversarial but also will believe that (i) B takes reasonable measures to prevent adversaries from gaining access to privileged information and (ii) B responds to other requests for such information (e.g., legal requests) in a manner deemed reasonable by A.

3 Undetectability in Identity Management

An action is *undetectable* to a party if that party cannot distinguish whether the action has occurred. Undetectability is a privacy property in so far as it concerns a user's ability to conceal actions from other parties.

Users perform different types of actions: creating identities and updating attributes, authenticating with identity providers, and issuing authorization requests to various service providers. Most involve the user communicating with a single party; third parties can be prevented from detecting such actions by the use of direct, encrypted communication channels.

Authorization requests, however, involve the user communicating with both an identity provider and a service provider; the identity provider serves as a proxy for the user. In this section, we describe how various existing design choices—interactive authentication, active-client authentication, and credential-based authentication—impact the detectability of authorization requests.

Note that providing users with undetectable actions also relies on the cooperation of the service providers in the system. Even if a system enables undetectable actions, a service provider can share information about user actions with the identity provider. Preventing such collusion is beyond the scope of an identity management system, although it is necessary to ensure that user actions are truly undetectable.

Interactive Authentication. By storing an identity at a particular identity provider, a user is necessarily trusting that identity provider to use attributes only for purposes sanctioned by the user. If that trust extends to not revealing information about user actions

[7] In some systems, attribute validation, management, and/or storage are delegated to a separate party called the *attribute provider*.

(in particular, authorization requests) then it is unnecessary to hide authorization requests (including *context* information such as which service provider and which attributes) from the identity provider. An identity management system now can adopt *interactive authentication* for generating authentication assertions.

Interactive authentication refers to any protocol in which a service provider communicates with an identity provider to obtain authentication assertions about a user. The user might be sent a request for authentication credentials each time an authentication assertion is solicited by a service provider (as shown in Figure 3.1), or the user could authenticate once with the identity provider prior to issuing any authorization requests. Because service providers interact with identity providers either directly (e.g., SOAP[8]) or indirectly (e.g., HTTP redirects), an identity provider serves as a user proxy and, therefore, necessarily detects authorization requests and observes the context in which such requests are made. Because information gathered in this way can be valuable (e.g., for behavioral advertising), identity providers have an economic incentive to adopt an interactive authentication system despite the limited privacy offered by such a design.

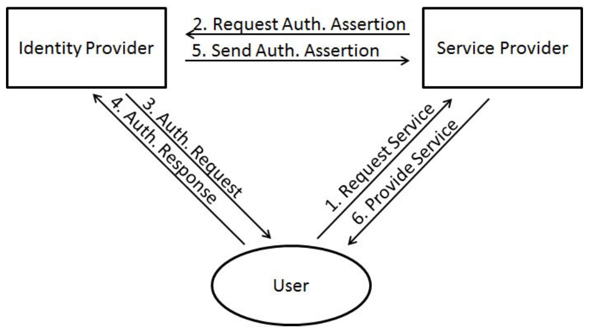


Figure 3.1: Typical Control Flow using Interactive Authentication

There are three distinct ways interactive authentication is typically implemented: authentication assertions can be signed with a shared secret key, they can be signed with a private key (presumably supported by a public-key certificate infrastructure), or they can be interactively verified. Existing identity management systems that employ interactive authentication (Passport, Project Liberty, Shibboleth, OpenID, and Facebook Single Sign-On) implement one or more of these approaches. Passport authentication assertions are encrypted with a shared DES key that is established when a service provider registers with the Passport service; authentication assertions thereafter can be verified without further interaction. Facebook authentication assertions—called *access tokens*—are digitally signed. Project Liberty and Shibboleth both support two different protocols for a passive client: an identity provider can either respond to a request with a reference (or SAML[9] *artifact*) that must be verified interactively, or it can send a digitally signed SAML response using HTTP protocols. OpenID also supports two authentication protocols: an identity provider-service provider pair can either establish a *relation* by exchanging a secret key, which is subsequently

[8]The Simple Object Access Protocol (SOAP) [36] is a protocol specification for direct communication between servers (e.g., identity providers) using remote procedure calls.

[9]Shibboleth implements the OASIS Security Assertion Markup Language Standard (SAML) [37], so all messages are formatted to comply with SAML specifications and guidelines.

used to encrypt and non-interactively verify authentication assertions, or authentication assertions can be verified interactively.

Interactive authentication limits exposure to stale attributes. Because authentication assertions are issued exclusively in response to a specific request (typically enforced using nonces and/or timestamps), attributes can be rechecked before each authentication assertion is issued. So the window for stale assertions can be limited to the communication delay in the system. This does not, however, address problems from cached attribute values or from stale values that are propagated by a service provider.

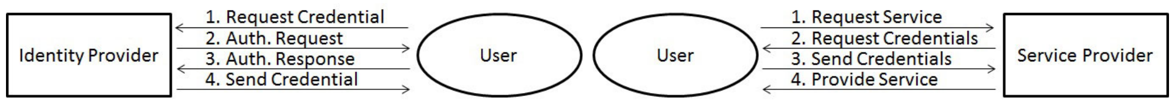
Active-Client Authentication. Although a user must trust an identity provider with attributes, that trust may not extend to arbitrary information about the user's actions. *Active-client authentication* is similar to interactive authentication, except a user (or a web-browser acting on behalf of the user) implements local functionality, including local state information and/or code for deciding which messages to send to which parties. Authorization requests remain detectable by identity providers, but the local functionality can prevent an identity provider from observing context associated with an authorization request (e.g., which service provider is contacted or which attributes are disseminated). To deploy this approach with today's browsers, a software extension would have to be downloaded.

Identity providers in an active-client system only produce authentication assertions in response to a specific request. So the window for stale assertions can be limited to the communication delay in the system.

CardSpace and Higgins both implement active-client authentication and do not disclose the identity of a service provider to the identity provider that issues an authentication assertion. Project Liberty also includes the option of employing an active client (called a Liberty-enabled client). However, Project Liberty uses the active client exclusively to free a service provider from the need to identify and locate the correct identity provider; identity providers still observe the context in which authorization requests are made.

Credential-Based Authentication. If users are unwilling to trust identity providers with information about authorization requests, including the existence or frequency of such requests, then such requests must be undetectable to identity providers. Undetectable authorization requests can be implemented using *credential-based authentication*, as illustrated in Figure 3.2. Credentials are transferable digital artifacts—issued by identity providers to authenticated users—that convey authentication assertions. The subject of these assertions is the user or identity to whom the credential was issued. And these assertions may include attributes that uniquely identify the subject of the credential. Credentials can be presented to a service provider at any time after they are issued. The protocol for presenting a credential varies. In some cases, a user might send a service provider a copy of the credential; in others, the user proves possession of such a credential.

An identity provider generates credentials without knowledge of which service provider will receive them, when, or how many times. Since credentials can be presented at any time after being issued, and since credentials can be verified non-interactively, an identity provider cannot detect an authorization request and cannot observe any information about the context in which the authorization request occurs. Some types of credentials allow the



(a) Credential Issue (b) Credential Presentation

Figure 3.2: Typical Control Flow using Credential-based Authentication

user to *selectively release* attributes by eliminating certain attributes from a credential or by replacing attributes with coarser values (e.g., replacing $U.age = 25$ with $U.age \geq 18$).

If credential transfer and delegation are considered undesirable, then it suffices to ensure that credentials are successfully presented only by the user to whom they were issued. Existing identity management systems that implement credential-based authentication use cryptography to ensure that only a party with knowledge of a particular secret (presumably known only by the subject of the credential) can successfully present a credential. The secret is not revealed during the presentation of a credential. A user U in Idemix has a master secret S_U used during the creation of pseudonyms and credentials issued to U. To present a credential, U proves knowledge of a secret S_U and a credential generated using S_U that contains the appropriate attributes. PRIME uses Idemix credentials. U-Prove uses a similar approach, except it employs per-credential secrets instead of a single master secret for each user. The protocol for generating U-Prove credentials returns two outputs to the user U: a credential c and a corresponding secret s_c which is revealed only to the user. When presenting the credential c, U provides a proof of knowledge of the secret s_c used to generate c. P-IMS uses blind signatures to produce credentials; the credential itself is therefore known only to the user and can be used as the secret. To present a P-IMS credential, the user proves knowledge of a credential satisfying the appropriate requirements.

Secrets alone cannot prevent users from delegating credentials to other parties. Idemix employs two techniques for ensuring that a credential is accepted only when presented by the subject of that credential: tieing knowledge of a secret to an external secret with real-world value (e.g., bank account information) or ensuring that a user U' who knows U's secret S_U can use all of U's credentials (including, for example, the credentials for accessing U's bank account).[10] Non-technical means (e.g., user contracts and legal accountability) can also discourage users from sharing credentials.

Since a credential can be presented long after being issued, identity management systems that employ credential-based authentication must address the issue of stale credentials. Several approaches have been suggested, including expiration dates and revocation lists (see [4, 22]); however, these approaches either require the user to place additional trust in another party or require some party to incur significant computational overhead.

[10]These techniques work because users are generally unwilling to share valuable information and will, therefore, not reveal the secret S_U.

4 Unlinkability

Unlinkability is a privacy property that concerns hiding correlations between actions and/or identities. For each type of linking—between two identities, between an action and an identity, and between two actions—the level of trust that users place in other parties determines the most appropriate design choice.

4.1 Identities Linked with Other Identities

The first question is whether users trust identity providers with connections between different identities and, if not, then how to prevent such linking. But note that providing users with unlinkable identities implicitly relies on the assumption that trusted identity providers do not collude with each other to enable linking of distinct identities. Preventing such collusion is beyond the scope of today's identity management systems.

Centralized. If identity providers can be trusted with identity linking, then a *centralized identity management system*—one in which a single, dedicated identity provider manages all identities for all users—is a sensible design choice. A user who opts-in must trust the identity provider, and all user attributes would have to be disclosed to this identity provider. Even when a user chooses to create multiple distinct identities (e.g., a personal identity and a work identity), patterns in attribute value or use could allow the identity provider to link multiple identities to the same individual because all identities are stored by the same identity provider. Since information that links user identities and actions performed under those identities is economically valuable, identity providers have an incentive to favor a centralized identity management system. Passport and Facebook Single Sign-On are examples of centralized systems.

Federated. A *federated identity management system* instantiates an intermediate design point in which a user chooses which identity providers to trust with links between certain identities associated with that user. Project Liberty supports this. There, identity providers that have established business relations form *circles of trust.* Within a circle of trust, a user can opt to *federate* two identities, in which case the identity providers exchange information and the identities are linked.

Decentralized. If identity providers only can be trusted with attributes that are specifically released to them and not trusted with identity linking, then a *decentralized identity management system*—in which multiple, distinct identity providers each function separately, possibly using different protocols, and may not even be aware of each other—is the most appropriate choice. A user can create one or more identities with any identity provider in the system. This architecture not only allows users to choose which identity providers to trust with which attributes, but it also allows a user to distribute sensitive attributes across distinct identity providers, thereby ensuring unlinkability of distinct identities. This approach has been almost universally adopted by existing identity management systems (e.g., Idemix, Shibboleth, Higgins, PRIME, OpenID, CardSpace, U-Prove, and P-IMS).

4.2 Identities Linked with Actions

Another question is whether service providers can be trusted with information that links an authorization request to information that uniquely identifies the issuer of that request or to an identity associated with the issuer. Of course, support for anonymous actions implicitly relies on the cooperation of the service providers in the system. A service provider always can request uniquely identifying attributes, thus undermining efforts to create anonymity. Ensuring that service providers support anonymous users (by making attribute-based authorization decisions rather than requiring a unique identifier) is a significant problem that is beyond the scope of identity management systems.

Pseudonymous Authorization. If service providers are trusted to link authorization requests to identities, then all such requests for a given identity could be issued with a unique identifier or *pseudonym*. Pseudonyms are typically (although not necessarily) opaque, globally-unique identifiers. They are commonly used to facilitate single sign-on. Permanent (or long-lived) pseudonyms enable linking, so identity providers and service providers have an economic incentive to favor pseudonymous authorization.

Pseudonymous authorization is implemented by Passport, Project Liberty, and OpenID. Passport credentials all contain the opaque, globally-unique Passport Unique Identifier (PUID). In Project Liberty, federated parties exchange user handles that are locally unique. OpenID users submit a globally-unique identifier (e.g., `username@example.com`) to the service provider to initiate authorization.

Anonymous Authorization. If service providers are not trusted with links between authorization requests and identities, then an identity management system can employ *anonymous authorization*. Anonymous authorization does not ensure that a service provider never links an authorization request to information that uniquely identifies the identity (for example, a particular combination of attributes); it only ensures that such linking cannot occur unless such a combination of attributes is explicitly required by the service provider.

Anonymous authorization can be implemented simply by eliminating all unique identifiers (at least, those that are not explicitly required by the service provider) from messages or credentials. Shibboleth, for example, supports anonymous authorization (although a user can choose to reveal a persistent identifier). Project Liberty allows a service provider to request an anonymous, temporary, identifier for some user if the service provider elects to support anonymous authorization. U-Prove credentials have unique identifiers, however, the identifier is tied to the credential and not the identity, so never re-using a credential with the same service provider is sufficient to achieve anonymous authorization. Idemix and P-IMS both employ zero-knowledge proofs[11] to allow a user to demonstrate the existence of a credential.

Anonymous authorization interferes with standard notions of accountability, yet many communities believe that individuals should be held accountable for unacceptable online behaviors (e.g., fraud, accessing illegal materials). To support both anonymous authorization

[11] A zero-knowledge proof, by definition, reveals nothing other than the veracity of the claim (in this case, knowledge of a valid credential) and, therefore, supports anonymous authorization.

and accountability, a *deanonymizing party* can be implemented. It would (under certain circumstances) link an action to the identity behind it. So users are anonymous under normal circumstances, but an authorized agency can, when necessary, determine the individual responsible for an online action.

In Shibboleth, the identity provider observes and logs all user actions, so it can act as the deanonymizing party. In P-IMS, authorization is granted to a particular *persona*; the identity provider that issued a credential or persona can determine the identity or persona to which it was issued, and therefore identity providers collectively perform the functionality of the deanonymizing party. Deanonymization is more difficult in Idemix (where credential presentation consists of a zero-knowledge proof) and U-Prove (where the issuing identity provider does not learn the credential identifier). However, users in either system can choose to include identifying information that is encrypted under the public key of a designated deanonymizing party (and a proof that the ciphertext was correctly generated), so that accountability can be enforced by a third party deanonymizer.

4.3 Actions Linked with Other Actions

A final dimension to linking concerns whether service providers (either alone or in cooperation with an identity provider) are trusted with information that links various actions attributed to the same identity or credential and, if not, how such linking can be prevented.

Linking Multiple Authorization Requests. If a service provider can link authorization requests to an identity (or an identifier for an identity, e.g., pseudonymous authorization), then, by transitivity, it can link all of the authorization requests that were issued using the same identity. If the system instead employs anonymous authorization, whether or not the service provider can link multiple authorization requests to each other depends on the implementation. Existing interactive, anonymous authorization protocols (i.e., Shibboleth) provide no persistent identifier across authentication assertions, so actions cannot be linked to each other. To show an Idemix credential, the user employs a zero-knowledge proof, so two authorization requests (even when issued using the same credential) cannot be linked. However, U-Prove credentials have a unique identifier, so requests containing the same credential can be linked (although requests issued with different credentials cannot). And P-IMS authorization protocols require the user to reveal the claimed persona, so requests issued with the same persona can be linked (although, again, requests issued with different personae cannot).

Linking Credential Issue with Credential Presentation. Another type of action linking is feasible if a service provider and an identity provider can cooperate to link a credential issued by the identity provider to its use in an authorization request. Credentials for which these actions cannot be linked are sometimes called *untraceable*. In systems that employ interactive or active-client authentication, such links can be established. (Even if the authorization is anonymous, the link can typically be established by comparing time stamps.) Since the same persona is used during the issuing protocol and the use protocol, this link can also be established by identity providers and service providers in P-IMS. However, both

U-Prove and Idemix implement untraceability.

5 Confidentiality in Identity Management

Identity management systems should enable users to control which parties learn specific attributes. Three channels could allow a party P to learn attributes about a user U:

- *Intentional channels*: P receives the attributes directly from U or from some identity provider *IDP* acting on behalf U.
- *Attribute forwarding*: P acquires the attributes from another party P'.
- *Attribute inference*: P deduces the attributes from other known or observed information.

Intentional channels are instigated by users. Design choices here focus on how users control what attributes to release, when, and to which parties. Attribute forwarding and attribute inference are not controlled by users, so design choices concern whether and how to control or prevent the release of attributes over these channels.

5.1 Release over Intentional Channels

We begin with different types of user control over the dissemination of attributes over intentional channels. These design choices can be applied both to direct release of attributes and to dissemination of attributes by an identity provider acting on behalf of a user. Of course, the user must trust the service provider with all disseminated attributes.

Instance-based Attribute Release. Under this approach, the user explicitly specifies the parties to which each attribute may be released. Instance-based attribute release is extremely flexible. However, the approach can be inconvenient, since it requires the user to make many decisions and since determining which service providers to trust can be difficult.

Nonetheless, many existing identity management systems today provide instance-based attribute release to control the dissemination of attributes to service providers. CardSpace and Higgins service providers send the user a policy (described using HTML or WS-Security-Policy) specifying required attributes and authentication assertion types. The user must then (interactively) decide whether to share the requested attributes with the service provider and, if so, which available identity to use. Depending on the other design choices made by the identity management system, the choice of identity could determine what information various identity providers can observe and/or which attributes the service provider receives. Facebook Single Sign-on presents the user with permissions requested by the service provider, which the user must choose to accept or deny. Mechanisms for controlling attribute release are beyond the scope of the OpenID specification, but Google's OpenID identity provider does give users an option to permit or deny the release of each attribute required by the service provider (with an option to remember selected permissions for future interactions with that service provider). Existing credential-based identity management systems (e.g., Idemix, U-Prove, and P-IMS) leave unspecified the approach to attribute release; some of these systems allow a user to share an entire credential or to share derived credentials that convey subsets of the attributes and/or coarser attribute values.

Policy-based Attribute Release. With this approach, a user defines a *policy*, and an attribute is released to a service provider if and only if the specified policy is satisfied for that attribute. The policy can involve attribute type, attribute value, party identity, or properties of the party when defining the conditions for attribute release. In a typical implementation, software (e.g., the web browser) will automatically determine which parties should receive specific attributes on the basis of the user's policy. Although policy-based attribute release is typically less expressive than instance-based (depending on the language in which policies are written), policy-based release simplifies the user experience by automating the dissemination of attributes. Policy-based attribute release is employed by Passport, Shibboleth, and PRIME.

Passport employs a very restrictive language for expressing user policies. A user tags each attribute with a label (e.g., `public` or `private`). Attributes are then automatically released to other parties based on these tags. Specifically, Passport sends attributes tagged `public` to any service provider with which the user chooses to interact.

In Shibboleth, attribute release is controlled by an *attribute filter*, which is a collection of policy rules. Each *policy rule* consists of a single *requirement rule*—it determines whether the policy rule is binding for a particular authentication request—and zero or more *attribute rules*—these specify which attributes are governed by the policy rule and whether or not those attributes will be released. Each policy rule is expressed using a flexible policy language and can depend on each of the parties involved as well as attribute type and value.

The PRIME system makes extensive use of policies that govern data access, data release, and data handling. A user and a service provider both define policies. Starting from these, the system automatically negotiates conditions of data release; this negotiation can, but does not necessarily, include active user input. A successful negotiation finds a solution that satisfies both parties' policies; such a solution is required before any attribute will be disclosed. PRIME software, running locally at the service provider, is designed to automatically enforce any conditions that are agreed during the negotiation.

5.2 Release by Attribute Forwarding

Attribute forwarding occurs when a party P acting on its own initiative discloses an attribute to another party P'. An example is when a service provider that sells goods forwards a user's address to a service provider that arranges delivery. Attribute forwarding is invisible to the user, but some identity management systems introduce mechanisms that allow users to prevent or control such disclosures, nevertheless.

Deniable Attributes. *Deniable attributes* cannot be validated without cooperation from an identity provider or user. This embodies the philosophy that the significant concern about attribute release is whether parties can prove a user satisfies some particular attribute—not whether those parties can merely learn or claim the user satisfies those attributes. The approach is widely adopted in existing identity management systems.

Systems that employ interactive authentication implement deniable attributes by encrypting assertions with a shared secret key (Passport and OpenID) or by sending service providers a reference that will be verified interactively only with explicit user permission (Project Liberty and Shibboleth). Systems that employ credential-based authentication can implement deniable attributes by requiring a party to know a particular secret before a credential can be validated (Idemix, U-Prove, and P-IMS).

Obligations. Another approach to control over attribute forwarding involves tagging released attributes with *obligations* that describe if and how the attribute can be used. An obligation could, for example, describe circumstances under which the attribute may be forwarded to a third party. In order to be an effective control, obligations must be obeyed by recipients of the tagged attributed. Thus, a user must trust those recipients; such trust can be developed on the basis of previous experience or reputation, assertions generated by trusted hardware, and/or (if deviations can be detected) on the basis of legal accountability.

PRIME supports obligations to implement control over data use (including attribute forwarding); a policy negotiation phase occurs prior to attribute release, and all released attributes are tagged with obligations that are mutually agreed upon during that phase. Obligations can specify notification requirements, deletion requirements, and limitations on forwarding or other uses of attributes. PRIME relies on a combination of legal accountability and secure hardware as the basis for users to trust that obligations are enforced; legal accountability ensures service providers use correctly-installed trusted hardware modules, the trusted hardware generates cryptographic assertions that tagged attributes are accessed only by PRIME software, and legal accountability can be invoked if correct assertions are not received.

5.3 Attribute Inference

Attribute inference occurs when a party P can deduce the value of an attribute from other information known to P. Attributes can be inferred from user behavior, including websites visited and items viewed or purchased, or from other attributes.[12] One standard defense against attribute inference is to minimize the amount of information that P learns—this can be done by limiting the types of user behavior that can be detected or observed by P (as discussed in Section 3) and by limiting the attributes that P can learn over other channels (as discussed in Sections 5.1 and 5.2). If a party has prior access to information about particular user action or attribute, then it is impossible to prevent that party from learning whatever attributes can be inferred from that information alone. However, in many cases, accurate attribute inference requires linking many different pieces of information and even performing statistical analysis. A second standard defense is, therefore, to prevent linking using one of the techniques discussed in Section 4.

[12] Preventing attribute inference is related to the problem of privacy-conscious information disclosure. There has been extensive work on this subject in the context of databases, culminating in the notion of differential privacy [12] for database queries—the goal of which is to prevent an adversary from inferring attributes that could not be deduced from previously available information.

6 Prior Characterizations of Identity Management

In an effort to understand the failures (and limited successes) of preceding identity management systems, Cameron [8] proposed seven *laws of identity* that he claims are essential for an identity management system to succeed. In terms of the landscape sketched in this paper, many of Cameron's laws advocate for particular design choices associated with the various privacy properties.

(1) *User Control and Consent*: An identity management system must obtain a user's consent to reveal information that identifies the user.

(2) *Minimal Disclosure for a Constrained Use*: An identity management system that discloses less identifying information and imposes more limits on its use should be preferred.

User Control and Consent (1) and Minimal Disclosure for a Constrained Use (2) are what we have termed confidentiality properties. In articulating these laws, Cameron argues that users should control attribute dissemination. In particular, an identity management system should provide a user with information (e.g., an attribute-use policy) that enables the user to make informed decisions about attribute dissemination. With (2), Cameron also argues for mechanisms that disseminate coarser versions of attributes and for mechanisms that restrict attribute use (e.g., obligations), although his work does not include examples of any such mechanisms.

(3) *Justifiable Parties*: An identity management system must be designed so that identifying information is disclosed only to parties having a necessary and justifiable need.

Justifiable Parties (3) is an argument that users should be aware of the parties with which they interact or share information, and that information disclosure should be limited to necessary parties. This rule implies that user actions should be undetectable (e.g., as supported by a credential-based system), since it is not necessary for identity providers to obtain information about authorization requests. In practice, an active-client system in which the context of a user action is unobservable would suffice to address the privacy concerns that motivate this law.

(4) *Directed Identity*: An identity management system must support global identifiers for use by public entities and local identifiers for use by private entities.

Directed Identity (4) is concerned with linking user actions across multiple service providers. Cameron argues in favor of local pseudonyms over the use of universal pseudonyms. But as stated, (4) precludes the possibility of an anonymous authorization system, although such systems are consistent with the underlying motivations.

(5) *Pluralism of Operators and Technologies*: An identity management system must support interoperability of multiple identity technologies run by different identity providers.

Pluralism of Operators and Technologies (5) precludes centralized systems, and it advocates that different service providers implement different types of authentication assertions.

(6) *Human Integration*: An identity management system must employ unambiguous human-machine communication mechanisms that prevent identity-based attacks (e.g., phishing and impersonation).

(7) *Consistent Experience Across Contexts*: An identity management system must provide a simple, consistent experience to users while supporting multiple operators and technologies.

Human Integration (6)—which argues that identity management systems should employ a clear, secure user-interface—and Consistent Experience Across Contexts (7)—which argues that this interface should be consistent across the various parties in the system—are universally accepted as good design principles but are orthogonal to the privacy-centered characterization that is the focus of our work.

In a follow-up paper, El Maliki and Seigneur [24] give overviews of Project Liberty, Shibboleth, OpenID, CardSpace, and Higgins and then evaluate these systems relative to Cameron's laws of identity. They conclude that Higgins most closely embraces the design choices promoted by Cameron's laws of identity.

Bhargav-Spantzel et al. [1] take a sociological approach to identifying key privacy principles for an identity management system. After discussing numerous user surveys, they emphasize that user's privacy goals differ, depending on the type of attributes and the receiving party. They argue that identity management systems should provide information about attribute validation, implement protection against false attributes and identity theft, and eliminate invisible channels for attribute dissemination (i.e., attribute inference), and they critique CardSpace, Project Liberty, and Shibboleth with respect to the identified privacy concerns. They also argue that new identity management systems should be developed that address these concerns.

Other surveys of identity managements systems focus on functionality (e.g., attribute namespace and propagation [16], ease of deployment [26], control [9]) or economic incentives [19, 21].

7 The Framework Applied to Public Policy

The preceding taxonomy of design choices is not only descriptive, but it has analytic value in the design of an identity management systems intended to serve public policy goals. As an illustration, we consider the National Strategy for Trusted Identities in Cyberspace (NSTIC), initiated by the United States federal government in April, 2011.[13]

NSTIC is intended to foster the development of an *identity ecosystem* that makes online transactions more secure, enhances consumer privacy, and encourages the development of future, identity-driven online services. The NSTIC strategy document [27] highlights the need for new, secure, private solutions to enable validation, authentication, authorization,

[13]NSTIC is a current White House initiative and thus its proposals and recommendations continue to be revised. For the purpose of this discussion, we focus on the most recent strategy document. Although the content is subject to change, it is a useful example to illustrate how our approach can be leveraged to extract insights pertaining to current public policy initiatives.

and accountability. And it identifies four principles to guide the design of an identity ecosystem: (1) privacy-enhancing, (2) secure and resilient, (3) interoperable, and (4) easy to use. We now show how the design choices described in this paper can inform the design of such systems.

Noting that people currently use drivers' licenses as a real-world assertion without the DMV or the state government being aware of their actions, NSTIC recommends implementing a system in which user actions are undetectable by an identity provider. Undetectable actions are also consistent with NSTIC's articulated goal of enhancing user privacy by minimizing the release of information about the user. As noted above in Section 3, credential-based systems implement undetectable actions but no other design choice does. Credentials are also employed by all existing identity management systems that focus on anonymity (a feature that NSTIC recommends identity management systems support). We thus should not be surprised to find that an NSTIC-compliant ecosystem explicitly embraces credential-based authentication.

NSTIC advocating for a credential-based system is consistent with other goals espoused by the strategy document. NSTIC stipulates both that user privacy be enhanced by releasing minimal information about attributes and that the system be easy to use. This balance between expressiveness and convenience best materialized with a system that uses policy-based attribute release. Since it is possible to implement credentials that permit selective release of attributes and support arbitrary claims about attributes (e.g., Idemix credentials), policy-based access control can be incorporated into a credential-based system (although no system currently combines these two design choices). Moreover, the goals of privacy-enhancement and release-minimization imply that techniques discussed in Section 5 should be employed to control or minimize the release of information through invisible channels; these techniques also are consistent with a credential-based system. NSTIC envisages a system in which multiple identity management technologies issued by multiple identity providers are recognized and accepted, interoperably by service providers in the system; this implicitly assumes a decentralized system. NSTIC also specifically asserts that anonymous authorization should be supported and that linking between actions should be minimized. These goals, too, can be supported by a credential-based system.

However, the NSTIC documentation asserts that an identity ecosystem should be resilient to the loss, compromise, or theft of user credentials. This goal is difficult to achieve in a credential-based system, because it requires means to revoke credentials. And existing approaches to revocation conflict with NSTIC's other goals. If credentials are issued for a specific service provider, then they can be revoked by contacting that service provider. However, such a design allows the identity provider to learn information about user actions that could violate the privacy-goals of the system. If credentials are instead issued with short expiration dates, then credential loss is only a vulnerability for a short interval. However, this allows identity providers to observe some information about patterns in user authorization and/or requires significant computational overhead. If neither of these approaches is adopted, then the inability to revoke credentials requires a user to contact all service providers to ensure that lost or stolen credentials are not fraudulently used. Such a task is infeasible and certainly violates the ease-of-use goal. It is, therefore, not possible to implement a credential-based system that is efficient, easy-to-use, and resilient to the loss of user credentials.

Moreover, in order to support flexible credentials that enable partial release of identities, the user must perform some computations (e.g., selective release) locally. Given the limitations of current web-browsers, this implies downloading and installing specialized software, a requirement that conflicts with NSTIC's goals of facilitating deployment and ensuring ease-of-use.

However, if a system designer is resigned to deploying a system that makes certain decisions locally (and the difficulties that accompany this choice), then it is possible to abandon the credential-based approach in favor of an active-client system. Although this makes user actions detectable to identity providers (as discussed in Section 3), active-client systems can ensure that the context and details of those actions are still unobservable, a standard that appears consistent with the stated goals of NSTIC. Moreover, an active-client system would ameliorate the security and resiliency concerns that arise from a credential-based system while maintaining compatibility with the other design choices implied by NSTIC's goals and discussion.

So, the goals outlined by NSTIC imply that active-client systems (and the software to support them) would be a profitable focus for future research on private, secure, interoperable, and easy-to-use identity management systems.

A Details of Existing Identity Management Systems

This appendix discusses historical and current identity management solutions in chronological order, and it sketches design choices made by each. We discuss Passport, Project Liberty, Idemix, Shibboleth, Higgins, PRIME, OpenID, CardSpace, U-Prove, P-IMS, and Facebook Single Sign-On. Although our primary focus is implementations rather than specifications, we do discuss identity management specifications for which there are no representative implementations (e.g., Project Liberty, OpenID). Specifications for which a representative implementation exists (e.g., SAML, OASIS-IMS, WS-InfoCard) are covered by the systems we discuss that implement those specifications. All of these design choices are summarized in Table A.1, and descriptions of the various systems are given in Subsections A.1 - A.10.

A.1 Passport (1999)

Passport is an identity management system introduced by Microsoft in 1999, but it is no longer actively supported. Microsoft ran the sole identity provider in the system, and service providers joined the Passport system by registering with Microsoft, at which time they received a unique SiteID and a DES encryption key. After registering, a service provider could delegate identity management and user authentication to Microsoft.

Like many of the early identity management systems, Passport employed interactive authentication (thereby allowing the identity provider to both detect user actions and observe the context in which those actions occur). When a user requested access to a protected resource or service, the service provider redirected the request to a Passport server (locations were periodically published) and included the SiteID and a return URL as parameters. The Passport server verified the SiteID and returned URL against the list of registered

			Pass.	Lib.	Ide.	Shibb.	Higg.	PRIME	OpenID	CardSp.	U-Pr.	P-IMS	FB
Undetect.		Interactive	√	√	×	√	×	×	√	×	×	×	√
		Active-Client	×	√	×	×	√	×	×	√	×	×	×
		Credential	×	×	√	×	×	√	×	×	√	√	×
Unlink.	Identities	Centralized	√	×	×	×	×	×	×	×	×	×	√
		Federated	×	√	×	×	×	×	×	×	×	×	×
		Decentralized	×	×	√	√	√	√	√	√	√	√	×
	Ident./Action	Pseudo.	√	√	×	√	?	√	√	?	×	×	?
		Anonymous	×	×	√	√	?	√	√	?	√	√	×
	Actions	Auth. Req.	√	√	×	?	?	×	?	?	√	√	√
		Issue/Use	√	√	×	√	?	×	√	?	×	√	√
	Intentional	Instance	×	?	√	√	√	×	?	√	√	√	√
		Policy	√	?	×	√	×	√	?	×	×	×	×
	Forwarding	Deniable	√	√	√	√	?	√	√	?	√	×	×
		Obligations	×	×	×	×	?	√	×	?	×	×	×
	Inference	Actions	√	√	×	√	?	×	√	?	√	√	√
		Attributes	√	√	√	√	?	×	√	?	√	√	√

Table A.1: Design Choices made by Existing Identity Management Systems

service providers; if they matched, then Passport authenticated the user (either by acquiring a valid email-password combination and setting a DES-encrypted cookie or by reading a previously-defined cookie). Passport then encrypted the user's Passport Unique Identifier (PUID), timestamp, and public profile information under the service provider's DES key and redirected the user's browser to the return URL, with the encrypted information included as a parameter. The service provider decrypted the information and used it to make an authorization decision.

The centralized Passport identity provider stored a database of user identities. Each identity consisted of a unique identifier (PUID), authentication credentials (email, password, and an optional four-digit security key), and attributes. Nothing prevented Microsoft from linking different identities on the basis of source, usage, or attribute. Authentication assertions issued by Passport servers contained the unique PUID (so the system is pseudonymous) and, therefore, service providers could link an authorization request to the identity issuing that request, and could link two requests made with the same identity.

Passport provided minimal support for confidentiality. Since Microsoft ran the only identity provider, all attributes were stored on Microsoft servers; attributes were sent to a service provider on the basis of tags that labeled the attribute as either `public` or `private`. Attributes could be collected by Passport during registration (e.g., email address) or could be contributed by a service provider. All attributes (except PUID) were `private` by default, but a user could voluntarily set attribute tags to `public`.

Initially, the only attribute was an email address, but users and service providers could add other attributes. Certain types of information—e.g., financial information—were tagged as *wallet attributes* by Passport. Wallet attributes were disclosed only after receiving explicit, interactive, permission from the user. Attributes were encrypted using a private, shared key, so they were deniable. However, there was no support for obligations, and there was no protection against attackers inferring attributes from user actions or from other attributes.

Passport was never as widely adopted as originally hoped. Several theories attempt to explain why. The most prevalent concerns the limited support Passport offered for user privacy—in particular, user actions were detectable, observable, and linkable. Moreover, Microsoft, as the exclusive identity provider in the system, was in a position to take advantage of these privacy vulnerabilities. The discovery of several security vulnerabilities in the Passport implementation [20] also undermined confidence in Passport.

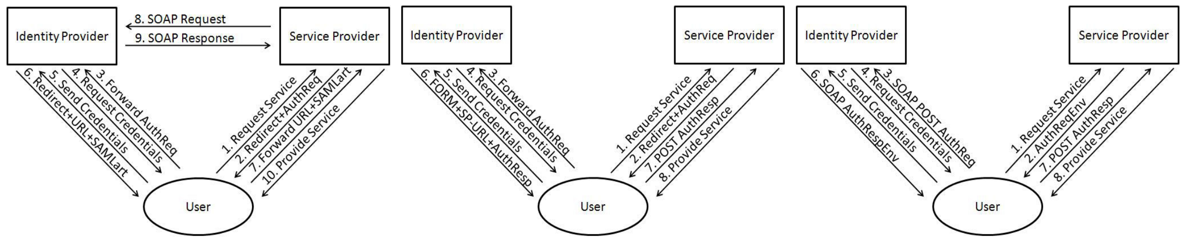
A.2 Project Liberty (2001)

The Liberty Alliance was a group of companies formed in 2001 to define standards that would enable individuals and businesses to engage in secure, private transactions by leveraging federated identity management systems. The standards were collectively referred to as Project Liberty, and involved three phases, released incrementally:

- **Phase 1:** Identity federation and single sign-on,
- **Phase 2:** A framework for building identity services (including attribute sharing),
- **Phase 3:** Interoperable identity services (including services to manage identities and profiles).

In 2005, the Project Liberty specifications were incorporated into SAML 2.0, a specification implemented by Shibboleth 2.0 and various enterprise identity solutions. The organization aspects of the Liberty Alliance were subsequently subsumed by the Kantara Initiative.

Since Project Liberty was a set of specifications rather than a specific implementation, it admitted multiple designs choice. It specified three possible control flows that supported authentication assertions: Artifact-based, POST-based, and Liberty-Enabled Client-based. The first two approaches were interactive, while the third was a form of active-client authentication. All three approaches allowed identity providers to detect user actions and to observe the context in which those actions occur. The three specified control flows are summarized in Figure A.1.



(a) Artifact-based (b) Post-based (c) LECP-based

Figure A.1: Types of Delegated Authentication Specified in Project Liberty

The Liberty Alliance envisioned a world in which businesses would form *circles of trust* based on contractual agreements and in which users would have an option to link or federate local identities within an established circle of trust. If a user chose to federate an identity, then the identity provider published an *introduction* to service providers in the same circle of trust (how this was implemented was not fixed by the Liberty Project specification, but use of a common domain was suggested). Service providers could *federate* or link a local identity with the identity stored at the identity provider using such introductions. Identities that were stored with un-federated identity providers (e.g., those in different circles of trust) could not.

By default, the authorization protocol was pseudonymous (the user is referred to by a local, shared handle), and user actions could be linked to an identity. However, the specification also included an option for the service provider to enable anonymous authorization using a temporary identifier. Whether a service provider could link two actions depended on whether the service provider opted for pseudonymous authorization; the use of an authentication credential could always be linked to its issuance.

The Project Liberty specifications allowed authentication assertions to include information about the authentication context [23] and user attributes [18]. How the system implemented user control over attribute confidentiality was beyond the scope of the Project Liberty specification, but the documentation recommended that disclosure of user attributes be governed by identity provider policies, user permissions, and interaction with the user. In the Bluewin [2] implementation of Project Liberty, users were asked to consent to attribute sharing when federating identities and when accessing the provider in question [38]. The specification did include some protection against attribute disclosure by third parties (i.e., attribute forwarding); since users were referred to by local handles, authentication responses (including any asserted attributes) were deniable. Project Liberty provided no mechanism to prevent attribute inference.

A.3 Idemix (2001)

Idemix is an anonymous credential system first proposed by researchers at IBM Zurich in 2001. Idemix is currently in a pilot phase, where it is employed in government, banking, and telecommunications; it is available as a Java library. Enhanced versions have also been used in the PRIME and Higgins projects.

All identity providers and service providers in the system generate and publish public keys that can be used for encrypting messages and verifying digital signatures. Each user U generates a secret key S_U that is used to generate pseudonyms for that user. U can establish an identity $N_{IDP}(S_U)$ that consists of attributes $\{attr_1, \ldots, attr_n\}$ with an identity provider IDP.

Idemix is a credential-based system, so user actions are undetectable to the identity provider. A user U obtains a credential $C(IDP, N_{IDP}, \{attr_1, \ldots, attr_n\})$ from an identity provider IDP by presenting authentication credentials for identity N_{IDP}. Credentials can be issued as one-time credentials, can be issued with an expiration date, or can be issued with a parameter that can be validated using a global revocation manager. To subsequently access a service provider SP, a user U demonstrates possession of an appropriate credential. If a local pseudonym $N_{SP}(S_U)$ has previously been established with the service provider, then U proves knowledge of a credential that incorporates the desired set of attributes and was issued to a pseudonym N' that is linked to the same secret key as the local pseudonym N_{SP}; if not, then U simply proves knowledge of a credential that incorporates the desired set of attributes and was issued to a pseudonym N' for which U knows the corresponding secret key.

Because the proof of knowledge employed during credential presentation requires the presenter to know secret key S_U, the Idemix protocol can ensure that any user U' who can successfully present a credential issued to a user U can successfully use all of U's credentials. This functionality is intended to prevent users from sharing the secret key with others. The secret key S_U can also be tied to knowledge of an external secret (e.g., bank account information), providing an additional incentive for users to keep their credentials private.

Idemix is designed to minimize linking. Different identities cannot be linked, actions cannot be linked to identities (because credentials are anonymous), and it is impossible for two actions to be linked (even if a user presents the same credential multiple times and even if service providers collude with identity providers). However, in order to support accountability, a user can voluntarily allow a trusted party (the identity provider or a third-party) to link the use of a credential to a pseudonym N_{IDP} or other identifying information by encrypting it under the trusted party's secret key and providing this information (along with a proof that it was correctly generated) when a credential is presented. Since service providers make authorization decisions, a service provider can choose to authorize only those users who submit de-anonymizable credentials.

Idemix implements instance-based attribute release; a user must make individual decisions regarding which attributes are stored with each identity provider and which attributes are disclosed to which service provider. Idemix credentials do, however, support selective release of attributes. Credentials can be used to assert subsets of the attributes or to assert "coarser" versions of the attributes. For example, if the credential were issued to assert that $N.age$ = *25* and $N.employer$ = *Example Co.* then it could be used assert only that $N.age \geq 18$; this is implemented by proving that a parameter in the credential is within a range rather equal to a specific value. Although Idemix credentials are deniable (since the successful proof of an attribute or claim requires knowledge of the secret S_U), Idemix provides no mechanisms to protect the confidentiality of attributes against inference.

A.4 Shibboleth (2003, v2.0 released 2006)

Shibboleth is a standards-based, open-source software package developed by Internet2 that implements the SAML standard to provide federated single sign-on and attribute exchange; it also provides functionality to manage attribute release. The first version was released in 2003, and Shibboleth 2.0 (which implements SAML 2.0 for its core protocols) was released in 2006. It is deployed primarily in academic environments; universities function as Shibboleth identity providers, and services to which they have subscribed (e.g., JSTOR) use the Shibboleth service provider implementation to control access to protected resources.

Shibboleth implements interactive authentication. When a user attempts to access a protected resource at a service provider, the service provider first identifies identity provider(s) with which the user is affiliated. This identification is done either by interacting directly with the user (e.g., presenting a list of identity providers and asking the user to select one) or by interacting with a centralized OASIS discovery service [39]. The service provider then redirects the request to the appropriate identity provider, and the user authenticates to that identity provider; the authentication method can either be specified by the service provider or can be determined by the identity provider. After a successful authentication, the identity provider sends a signed, encrypted authentication assertion (or a reference to it) back to the service provider via the user. The service provider verifies and unpackages the authentication assertion and uses it to make an authorization decision.

Whether Shibboleth is vulnerable to linking depends on some settings. A user can prevent different identities from being linked (since Shibboleth is a decentralized system), or the user can choose to store distinct identities with the same identity provider. Shibboleth supports both pseudonymous and anonymous authorization (anonymous authorization can be required by any party involved in the interaction). If anonymous authorization is chosen, then actions cannot be linked to identities and authorization requests cannot be linked to each other (although the use of an authentication assertion can be linked to its issuance if the service provider and identity provider collude).

In comparison to earlier systems, Shibboleth provides sophisticated protection for attribute confidentiality. A user makes individual, itemized decisions about which attributes to store with each identity provider, and Shibboleth then supports expressive policies governing disclosure of these attributes to service providers. Which attributes are disseminated (that is, which attributes are included in an authentication assertion) is determined by an attribute filter. An attribute filter is a collection of policy rules. Each policy rule consists of a single requirement rule—which determines whether the given policy rule is applied to a particular authentication request—and zero or more attribute rules—which specify what attributes are affected by the policy rule and whether or not they will be released or whether release is contingent on explicit, interactive user consent. Policy rules can depend on requester, issuer, principal, or attribute, and each of these can depend on exact matches, matching a regular expression, format, attributes of the party in question, regular expressions for attributes of the party in question, or arbitrary scripts. In order to minimize attribute forwarding, an identity provider may make assertions deniable by sending a reference rather than a signed assertion. However, no mechanism prevents attribute inference.

A.5 Higgins (2003, v1.0 released 2008)

Begun as an effort by Paul Trevithick to implement a dashboard for managing multiple distinct identities, Higgins developed into an open-source Information Card system which simplifies the development and implementation of identity solutions. Currently, the primary goal of Higgins is to encourage deployment of identity management systems (and the Higgins Information Card functionality, in particular) by facilitating installation and by supporting multiple platforms (including mobile systems). Higgins 1.0 implements the OASIS Identity Management interoperability protocol and performs the functionality of an identity card selector; it was released by the Eclipse Foundation in 2008. Higgins 2.0 (currently under construction) adds support for a personal data store.

Higgins is an active-client system in which the user decides what messages are sent to which parties (currently by running a browser extension and/or native applications). After establishing an identity with an identity provider, a user receives an *information card* associated with that identity. An information card is a reference to an identity that lists the types of security tokens (e.g., x.509 certificates, Idemix credentials, etc.) and the types of attributes that can be obtained from the identity provider. Depending on the Higgins version, information cards can be stored locally or on a dedicated Higgins server. A user can always become an identity provider by creating personal or self-issued information cards for attributes being claimed (e.g., username, address). When a user employs a particular information card, an authentication assertion corresponding to that identity is requested from the appropriate identity provider. Identity providers, therefore, detect user actions, but they cannot necessarily observe the context in which those actions occur.

Higgins is a decentralized system, so distinct identities established with distinct identity providers cannot be linked by any single party in the system (other than the user who established those identities). Whether user actions can be linked to identities or to other actions depends on the type of authentication assertion that is employed, something not specified by Higgins.

Confidentiality of attributes is controlled by an instance-based attribute release mechanism. A user explicitly decides which attributes to share with each identity provider when establishing a digital identity. Subsequently, during authorization, the service provider releases a list of requirements (specified either using WS-SecurityPolicy or HTML) and the user must decide whether to release the requested attributes and, if so, which identity to use. Once the user selects an identity (or information card), a request is sent to the corresponding identity provider, and it issues an authentication assertion for the requested attributes (in the required format, if specified). Whether and how the release of attributes across invisible channels is controlled depends on the format of the authentication assertion, which is not specified by Higgins.

A.6 PRIME (2004, v3.0 released 2008)

PRIME is the result of the four-year Privacy and Identity Management for Europe project that ran from 2004 to 2008; the system is a research prototype. The primary focus of PRIME is enabling users to effectively protect and control personal information.

PRIME assumes that users will serve as identity providers for self-asserted attributes; the system includes middleware that users can download and run for this purpose. Distinct, third-party identity providers—that independently validate attributes—are intended to issue high-assurance authentication assertions for use with service providers that do not trust self-asserted attributes.

PRIME is a credential-based identity management system. Prior to issuing an authorization request, a user interacts with zero or more identity providers to obtain authentication assertions for various attributes. The details concerning how a third-party identity provider validates attributes and authenticates users are not specified. Authentication assertions are implemented as digitally signed statements or Idemix credentials.

After receiving an authorization request, a service provider responds with a *Data Handling Policy*, which describes types of authentication assertions required for an affirmative authorization decision. The user then describes conditions under which the user will release various attributes, and PRIME automatically determines whether there exist attributes, formats, and conditions under which both parties would satisfied. The conditions could require that released attributes be tagged with obligations specifying notification requirements, deletion requirements, and other limitations on attribute use. Any obligations are automatically enforced by PRIME.

If an agreement is reached, then the user presents an authentication assertion containing appropriate (tagged) attributes. That assertion also contains a pseudonym, which can be relationship-based, role-based, role-relationship-based, or session-based, depending on preferences of the user and the service provider. Pseudonyms can be arbitrarily assigned strings or can be Idemix pseudonyms. The degree to which linking is possible depends on the nature of the pseudonym.

A.7 OpenID (2005, v2.0 ratified 2007)

OpenID is an open standard for decentralized identity management that was developed in 2005; the specification for OpenID 2.0 adds support for attribute management and was ratified in 2007. OpenID is now provided and accepted by many websites, including Google, IBM, Yahoo!, AOL, PayPal, VeriSign, MySpace, LiveJournal, and Steam.

OpenID specifies an interactive authentication protocol. The protocol allows identity providers to detect (and observe the context of) user actions. Authentication is initiated when a user requests a protected service from a service provider and provides an OpenID identifier. Such an identifier is a URL or XRI (e.g., `username.identityprovider.com`). The endpoint for the identity provider associated with the given identifier is determined by performing XRI resolution, employing the Yadis protocol, or by retrieving a document from the specified URL. The service provider and identity provider then optionally establish a *relation* (if there is none) by exchanging secret keys. Next, the service provider redirects the request to the identity provider with an OpenID authentication request. The identity provider authenticates the user and then redirects the request back to the service provider with a (signed) OpenID assertion. The service provider verifies the assertion and signature (using the shared secret key or via direct communication with the identity provider) and then makes an authorization decision.

The OpenID specification describes a decentralized system in which different identities associated with the same user cannot be linked. It supports both pseudonymous and anonymous modes of authorization (which method is employed is specified by the service provider). When anonymous authorization is employed, user authorization requests can be linked to a corresponding authentication assertion, but two distinct authorization requests made by the same user cannot be linked to each other.

OpenID 2.0 [5] introduces support for attributes [14], but protocols to enforce confidentiality are not specified. Attribute types are given as URIs, and attributes are arbitrary data encoded as UTF-8 strings. How attributes are created and which identity providers are given access to those attributes is not described by the specification, but it is suggested that all parties in the system (including identity providers and service provides) be allowed to create attributes associated with a user and to make itemized decisions regarding which identity providers store those attributes. A service provider can specify requested attribute types (designated either *required* or *if_available*); how an identity provider determines which attributes to release is not covered in the specification, but the Google OpenID identity provider gives the user the (interactive) option to permit or deny the release of each attribute on the required list (with an option to remember the selected permissions for future interactions with that service provider). OpenID attributes are deniable, but no other mechanisms for preventing or minimizing the release of attributes across invisible channels are specified.

A.8 CardSpace (2006)

CardSpace is a proprietary identity management metasystem released by Microsoft in 2006 as part of the .NET Framework 3.0. It was designed to facilitate management of multiple digital identities or "identity cards" and is consistent with the OASIS IMI standard. CardSpace was designed to work with arbitrary token formats (e.g., OpenID, SAML, U-Prove), but it was compatible only with Internet Explorer run on Windows operating systems. In 2011, Microsoft announced that it would not be releasing CardSpace 2.0 and would instead focus its attention on its new U-Prove system (described in Section A.9).

CardSpace is an active-client system. A user (or a browser acting on behalf of the user) decides what messages are sent to which parties in the system. Given the limitations of current web browsers, a user must install the CardSpace client locally in order to use the identity management system.

Upon establishing an identity with an identity provider, the user is given an information card. An information card is a locally-stored reference to an identity that lists the available types of credential, the list of attributes that can be obtained with that identity, a URL for requesting credentials, a time-stamp, and a globally unique identifier; information cards are signed by the issuing identity provider. CardSpace users can also serve as identity providers for self-asserted attributes (e.g., username).

Since CardSpace is an active-client system, identity providers can detect user actions, but they cannot necessarily observe the context in which those actions occur. The default implementation allows identity providers to observe which attributes are released at what points in time. However, the identity provider cannot observe which service provider receives those attributes. CardSpace identity providers may optionally require a service provider's identity (information which is necessary for certain types of credential).

Since CardSpace is an active-client system in which users store information locally, it must implement means to support users who use multiple devices. To address this, information cards can be copied onto a portable storage medium. However, in released versions of CardSpace, the user must download the information cards onto the local device prior to use.

CardSpace is a decentralized system. So, distinct identities established with distinct identity providers cannot necessarily be linked by any single party (other than the user who established those identities). Whether user actions can be linked to identities or to other actions depends on the type of credential that is employed and is not specified by CardSpace.

Confidentiality of attributes is controlled by an instance-based access control mechanism. A user explicitly decides which attributes to share with each identity provider when establishing a digital identity. In response to an authorization request, a service provider releases a list of requirements (specified either using WS-SecurityPolicy or HTML) and the user must decide whether to release the requested attributes and, if so, which identity to use (the CardSpace system, running locally on the user's machine, determines which identities can fulfill the specified requirements). Once a user selects an identity (or information card), a request is sent to the corresponding identity provider, and the identity provider issues a credential for the requested attributes (in the required format, if specified). Whether and how the release of attributes across invisible channels is controlled depends on the type of credential that is used and, therefore, is not specified by CardSpace.

A.9 U-Prove (2007)

U-Prove is an identity management solution based on cryptographic protocols developed by Stefan Brands [3]. First released by Credentica in 2007, U-Prove was subsequently acquired and re-branded by Microsoft. It is now available as an open-source implementation.

U-Prove is a credential-based identity management system where the credentials (called *tokens* in the U-Prove documentation) consist of digitally signed collections of attributes. More specifically, a U-Prove token consists of a unique identifier, a public key that encodes the attributes, a token information field that includes usage restrictions and expiration, the issuing identity provider's blind signature of the previous elements, and prover-asserted information (e.g., self-asserted attributes or a service provider-supplied nonce). In order to obtain a U-Prove token for a particular identity, a user authenticates with an identity provider and then participates in the interactive U-Prove issuance protocol, which returns a token and a private key (known only to the user). A U-Prove token can only be used by parties that know this private key. If token delegation and transfer are undesirable, then users can be discouraged from sharing private keys by linking the value to that of an external secret (e.g., bank account information). Since signatures and, therefore, tokens can be verified non-interactively, U-Prove is a credential-based system in which user actions are undetectable to the identity provider.

U-Prove is a decentralized identity management system. Users can, therefore, establish distinct identities with distinct identity providers, preventing those identities from being linked. U-Prove also is an anonymous authorization system, and user actions cannot be linked to the identity to which a credential was issued. Because token presentation involves sending the service provider a unique token identifier, the service provider can link different sessions in which the user was authorized using the same token. However, uses of two different tokens cannot be linked. Since U-Prove employs blind signatures to generate tokens, U-Prove tokens are untraceable (an identity provider cannot link the use of a token to the token that was issued).

U-Prove provides minimal technical support for confidentiality of user attributes. A user must make instance-based attribute release decisions to determine which attributes are disclosed to each identity provider. The mechanism used to control disclosure of attributes to service providers is not discussed in the U-Prove documentation. However, in the version integrated into CardSpace, the service provider requests a collection of attributes whenever a user wants to access a protected resource. The user responds by sending a U-Prove credential that includes the requested attributes as well as a zero-knowledge proof that the credential was generated using those attributes (and possibly other undisclosed attributes) and a private key known to the user. Observe, this allows release of subsets of the attributes in a given U-Prove credential. Attributes are deniable (since zero-knowledge proofs, by definition, cannot be reproduced), but U-Prove provides no mechanisms to prevent or control attribute inference.

A.10 P-IMS (2008)

Persona-based Identity Management (P-IMS) is an identity management system proposed by researchers at Queen's University in 2008. The primary goal of P-IMS is to find a balance between anonymity and accountability in a credential-based system. P-IMS currently exists only on paper.

P-IMS is designed to exploit the features of an identity-based signature scheme. ID-based signatures allow a party in the system to sign messages in a manner that permits other parties to verify the signature using only an identifier associated with the signer; private signing keys are issued to an identifier by a designated *key generator*. In the P-IMS system, each identity provider functions as a key generator, and the cryptographic identifiers are hashes of the (encrypted) list of attributes associated with an identity. To create an identity (which is called a *persona* in the P-IMS documentation), a user gives an identity provider a list of attributes that the user wishes to associate with that identity; the user may also optionally present credentials that attest to one or more of the claimed attributes. The identity provider validates the claimed attributes (a process that could require real-world interactions and/or verification of the presented credentials) prior to creating an identity with those attributes.

P-IMS is a credential-based authentication system, so user actions are undetectable to the identity provider. Upon receiving a request, an identity provider issues a credential for an established identity I (which is a collection of attributes along with any credentials P presented in support of those attributes) by generating the private key PK_I associated with identifier $hash(I)$ and sending $C = (I, PK_I)$ to the user. To present a credential, a user sends the signed identity (where the signature is generated using the key PK_I) or a signed list of label-encrypted attributes (along with labels for the desired attributes) to a service provider. The latter allows the user to selectively disclose attributes from a pre-defined credential, since label-encrypted ciphertexts require both the private key and a ciphertext-specific label in order to decrypt the ciphertext. After a credential has been presented, a service provider can non-interactively verify the credential by verifying the signature using the appropriate identifier. Non-interactive verification ensures that authorization requests remain undetectable to the identity provider.

Since P-IMS is envisioned as a decentralized identity management system with many identity providers, distinct identities associated with the same user may be stored with distinct identity providers, thereby minimizing the threat of identity linking. Moreover, P-IMS is an anonymous authorization system; credentials are not labeled with a name or pseudonym, so actions cannot be inherently linked to the user behind them. However, all authorization requests that are invoked using the same credential include the same identity (or label-encrypted list of attributes). Therefore, a service provider can link two actions that were invoked using the same credential.

P-IMS provides limited support for attribute confidentiality. Attribute release to both identity provider and service provider is controlled using an instance-based approach, although label-encryption does permit a user to release subsets of the attributes asserted in a credential. No mechanisms prevent or limit communication of attributes across invisible channels: attributes are not deniable (a credential can be non-interactively verified by any party), and there is no support for obligations. There are also no mechanisms designed to prevent or minimize attribute inference.

A.11 Facebook Single Sign-On (2008, released 2010)

Facebook is commonly considered to be a service provider. However, with the introduction in 2008 of Facebook Connect—a proprietary single sign-on system—Facebook became an identity providers as well. Its current identity management system, Facebook Single Sign-On, is a component of Facebook's Open Graph platform that implements the OAuth 2.0 [28] authorization specification. By leveraging the large Facebook user base and by providing economic incentives for service providers (namely access to large quantities of user data, including the social graph), Facebook Single Sign-On has become one of the most widely-deployed identity management systems [21].

Facebook Single Sign-On implements an interactive authentication protocol specified by OAuth 2.0. When a user attempts to access a protected resource, the service provider presents a login page that includes the option to authenticate with a Facebook account. When a user selects this option, the browser opens a Facebook window that asks for authentication credentials (email address and password) and any requested permissions. Facebook also sets a session cookie at this time. By default, basic information (which includes name, picture, friends list, and any information the user has made public) is requested; applications can also request additional permissions. After a user has authenticated with Facebook and granted any requested permissions, Facebook issues an *access token*—a signed string containing agreed permissions and an expiration date—which is forwarded to the service provider. The service provider can use the access token to make requests to Facebook's APIs, and it can make authorization decisions on the basis of the received information.

Facebook Single Sign-on is a centralized identity management system: Facebook is the sole identity provider. However, linking between multiple identities is not considered a privacy concern, since the Facebook terms of service limit a user to a single identity. Facebook Single Sign-On does not prevent linking between actions and identities or between different actions because it does not support anonymous authorization. Any service provider that receives an access token can access the user's "real-world" name.

Facebook offers little support for confidentiality of user attributes; if a user wishes to interact with a service provider using Facebook Single Sign-on then that user must grant the service provider all requested permissions. After permissions are granted, the service provider can use the resulting access token to make arbitrary calls to the Facebook APIs up until the token expires, and the resulting information can be used for any purpose.

Acknowledgements

We are grateful to Donna Dodson, Jeff Friedberg, Susan Landau, Naomi Lefkovitz, and Deirdre Mulligan for many helpful comments on an earlier version of this paper.

References

[1] Abhilasha Bhargav-Spantzel, Anna C. Squicciarini, Matthew Young, and Elisa Bertino. Privacy requirements in identity management solutions. In *Proceedings of the 2007 conference on Human interface: Part II*, pages 694–702, 2007.

[2] Bluewin. `http://www.bluewin.ch`.

[3] Stefan Brands. *Rethinking public key infrastructures and digital certificates—building in privacy.* PhD thesis, Eindhoven University of Technology, September 1999.

[4] Stefan Brands, Liesje Demuynck, and Bart De Decker. A practical system for globally revoking the unlinkable pseudonyms of unknown users. In *Proceedings of the 12th Australasian conference on Information security and privacy*, ACISP'07, pages 400–415, 2007.

[5] Johnny Bufu, Josh Hoyt, and David Recordon. OpenID authentication 2.0. `http://openid.net/specs/openid-authentication-2_0.html`, December 2007.

[6] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *Advances in Cryptology EUROCRYPT 2001*, pages 93–118, 2001.

[7] 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*, CCS '02, pages 21–30, 2002.

[8] Kim Cameron. The laws of identity. `http://www.identityblog.com/stories/2005/05/13/TheLawsOfIdentity.pdf`, 2005.

[9] Yuan Cao and Lin Yang. A survey of identity management technology. In *IEEE International Conference on Information Theory and Information Security*, ICITIS '10, pages 287–293, December 2010.

[10] David Chappell. Introducing Windows CardSpace. `http://msdn.microsoft.com/en-us/library/aa480189.aspx`, April 2006.

[11] Mina Deng, Kim Wuyts, Riccardo Scandariato, Bart Preneel, and Wouter Joosen. A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements. *Requir. Eng.*, 16(1):3–32, March 2011.

[12] Cynthia Dwork. Differential privacy. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *Automata, Languages and Programming*, volume 4052 of *Lecture Notes in Computer Science*, pages 1–12. Springer Berlin / Heidelberg, 2006.

[13] Facebook single sign-on. `http://facebook.com/`.

[14] Dick Hardt, Johnny Bufu, and Josh Hoyt. OpenID attribute exchange 1.0. `http://openid.net/specs/openid-attribute-exchange-1_0.html`, December 2007.

[15] Higgins. `http://www.eclipse.org/higgins/`.

[16] Wolfgang Hommel and Helmut Reiser. Federated identity management: Shortcomings of existing standards. In *9th IEEE Symposium on Integrated Network Management*, 2005.

[17] Mohammed Hussain and David Skillicorn. Persona-based identity management: A novel approach to privacy protection. In *Proceedings of the 13th Nordic Workshop on Secure IT Systems*, WWSecIT '08, pages 201–212, 2008.

[18] Sampo Kellomaki and Rob Lockhart. Liberty ID-SIS personal profile service specification. `http://projectliberty.org/liberty/content/download/1028/7146/file/liberty-idsis-pp-v1.1.pdf`.

[19] Sven Koble and Rainer Böhme. Economics of identity management: a supply-side perspective. In *Proceedings of the 5th international conference on Privacy Enhancing Technologies*, PET'05, pages 259–272, 2006.

[20] David P. Kormann and Aviel D. Rubin. Risks of the Passport single sign-on protocol. *Comput. Netw.*, 33(1-6):51–58, June 2000.

[21] Susan Landau and Tyler Moore. Economic tussles in federated identity management. In *Tenth Workshop on the Economics of Information Security*, WEIS '11, 2011.

[22] Jorn Lapon, Markulf Kohlweiss, Bart De Decker, and Vincent Naessens. Analysis of revocation strategies for anonymous Idemix credentials. In *Communications and Multimedia Security*, pages 3–17, 2011.

[23] Paul Madsen. Liberty ID-FF authentication context specification. `http://projectliberty.org/liberty/content/download/1209/7927/file/draft-liberty-authentication-context-v2.0-01.pdf`.

[24] Tewfiq El Maliki and Jean-Marc Seigneur. A survey of user-centric identity management technologies. In *International Conference on Emerging Security Information, Systems, and Technologies*, pages 12–17, 2007.

[25] Microsoft .NET Passport. `http://www.passport.net`.

[26] Annu Myllyniemi. Identity management systems: A comparison of current solutions. `http://www.tml.tkk.fi/Publications/C/22/papers/Myllyniemi_final.pdf`, December 2006.

[27] National strategy for trusted identities in cyberspace: Enhancing online choice, efficiency, security, and privacy. `http://www.whitehouse.gov/sites/default/files/rss_viewer/NSTICstrategy_041511.pdf`, April 2011.

[28] OAuth 2.0. `http://tools.ietf.org/html/draft-ietf-oauth-v2-31`.

[29] Openid. `http://openid.net/`.

[30] Christian Paquin. U-Prove technology overview v1.1. `http://research.microsoft.com/pubs/166980/U-ProveTechnologyOverviewV1.1.pdf`, February 2011.

[31] Andreas Pfitzmann and Marit Hansen. A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management. Technical report, TU Dresden and ULD Kiel, April 2010.

[32] PRIME: Privacy and identity management for Europe. `http://www.prime-project.eu`.

[33] Project liberty. `http://projectliberty.org`.

[34] Shibboleth. `http://shibboleth.internet2.edu/shib-v2.0.html`.

[35] David Skillicorn and Mohammed Hussain. Personas: Beyond identity protection by information control. A Report to the Privacy Commissioner of Canada, March 2009.

[36] Simple Object Access Protocol (SOAP). `http://www.w3.org/TR/soap12-part1/`.

[37] OASIS Standard. Security assertion markup language (saml), November 2002.

[38] Technical information on Bluewin identity provider. `http://projectliberty.org/liberty/content/download/378/2693/file/IdP_Public_TechWhitePaper_Englishv2.3.pdf`, June 2004.

[39] Rod Widdowson and Scott Cantor. OASIS identity provider discovery service protocol and profile. http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-idp-discovery.pdf, March 2008.