Federated Identity Management Systems: A Privacy-Based Characterization

Eleanor Birrell and Fred B. Schneider | Cornell University

Identity management systems store attributes associated with users and employ these attributes to facilitate authorization. A privacy-driven taxonomy of design choices found in these systems can help technical experts consulting on public policy relating to identity management.

People use the Internet to manage finances, access employer resources, shop, and communicate. Each activity involves interacting with a service provider. Such interactions typically require that each user have a digital identity. For the most part, each service provider stores and manages such identities, which are used to improve the user's experience, increase the service provider's profits, and defend against certain attacks.

Prior to the introduction of identity management systems, many believed that practices concerning online identities were problematic because

- each service provider maintains a set of user identities, so users have many identities (at least one for each service provider with which they interact), which becomes a management burden and creates potential points of failure; and
- users aren't given control over their attributes' dissemination, leading to privacy violations and identity theft.

Over the past 15 years, identity management systems evolved with the goal of addressing these problems, although actual mitigation of security concerns has been limited by the sparse deployment of such systems. Identity management systems enhance 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 this identity provider, which implements mechanisms that let users control attribute release and mechanisms that issue *authentication assertions*—statements about user attributes. Upon receiving an authorization request from software executing on a user's behalf, a service provider decides whether to authorize the user on the basis of the accompanying authentication assertions.

Existing work on identity management focuses primarily on individual systems, each of which focuses on one of three general types of functionality:

- *Single sign-on*. These systems issue authentication assertions to multiple service providers after a single user authentication. Examples include Passport, OpenID (http://openid.net), Shibboleth (http://shibboleth.net), and Facebook Single Sign-On.
- *Federated identity*. These systems manage multiple distinct identities for a single user and issue authentication assertions on the basis of any of these identities. Examples include Project Liberty (http://projectliberty.org), Higgins (www.eclipse.org/higgins), PRIME (www.prime-project.eu), CardSpace, and Client-Side Federation.[2]
- *Anonymous credentials*. These systems provide authentication assertions that don't reveal the user's identity to a service provider. Examples include Idemix,[3,4] U-Prove, and P-IMS.[5,6]

In this article, we identify key design choices intrinsic

 Copublished by the IEEE Computer and Reliability Societies

Table 1. Existing identity management systems' design choices.

			Passport	Liberty	Idemix	Shibboleth	Higgins	PRIME	OpenID	CardSpace	U-Prove	CS-Federation	P-IMS	Facebook SSO
Undetectability		Interactive	■	■		■			■					■
		Active client		■			■			■		■		
		Credential			■			■			■		■	
Unlinkability	Identities	Centralized	■											■
		Federated		■								■		
		Decentralized			■	■	■	■	■	■	■		■	
	Identity/action	Pseudonymous	■	■		■	?	■	■	?		■		?
		Anonymous			■	■	?	■	■	?	■		■	
	Actions	Authorization request	■	■		?	?		?	?	■	■	■	■
		Issue/use	■	■		■	?		■	?			■	■
Confidentiality	Intentional	Instance		?	■	■	■		?	■	■	■	■	■
		Policy	■	?		■		■	?					
	Forwarding	Deniable	■	■	■	■	?	■	■	?	■			
		Obligations					?	■		?				
	Inference	Actions	■	■		■	?		■	?	■	■	■	■
		Attributes	■	■	■	■	?		■	?	■	■	■	■

to existing identity management systems. We focus on connections between the design choices identified in current systems and discuss the impact of these choices on system functionality. We adopt a privacy-driven approach, which centers on three privacy properties: [7,8]

- *undetectability*—concealing user actions,
- *unlinkability*—concealing correlations between combinations of actions and identities (for example, untraceability), and
- *confidentiality*—enabling users' control over dissemination of their attributes.

Undetectability, unlinkability, and confidentiality are related properties—all concern which parties have access to particular data. However, these properties depend on orthogonal design choices and thus are best considered independently. The three properties define a privacy-focused design space that informs choices intrinsic to building and deploying identity management systems. Table 1 shows a chart depicting various design choices that impact these properties, listing what choices existing systems have made. (For more information on the characterizations of identity management, see the related sidebar.)

We focus primarily on these privacy properties' impact on identity management system design; however, other factors affect both the implementation and success of an identity management system. For example, economic incentives and user-interface design might trump privacy concerns for some systems. Still, privacy principles constitute an interesting lens through which to view the motivating principles that inform identity management system design, because policymakers and users often voice privacy concerns.

System Components

Although we identify and discuss high-level design choices informally rather than conduct a formal treatment, a good place to start is with careful definitions for the components and actors of concern. These definitions are standard.[9]

The *parties* in an identity management system are users, service providers, and identity providers. For convenience, we treat each party as monolithic, even though it could be implemented by a distributed system.

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.

Other Characterizations of Identity Management

In an effort to understand the failures (and limited successes) of preceding identity management systems, Kim Cameron proposed seven *laws of identity* that he claims are essential for successful identity management systems.[1]

- *User Control and Consent*: An identity management system must obtain a user's consent to reveal information that identifies the user.
- *Minimal Disclosure for a Constrained Use*: An identity management system that discloses less identifying information and imposes more limits on its use is preferred.
- *Justifiable Parties*: An identity management system must be designed so that identifying information is disclosed only to parties having a necessary and justifiable need.
- *Directed Identity*: An identity management system must support global identifiers for use by public entities and local identifiers for use by private entities.
- *Pluralism of Operators and Technologies:* An identity management system must support interoperability of multiple identity technologies run by different identity providers.
- *Human Integration*: An identity management system must employ unambiguous human-machine communication mechanisms that prevent identity-based attacks (for instance, phishing and impersonation).
- *Consistent Experience across Contexts*: An identity management system must provide a simple, consistent experience to users while supporting multiple operators and technologies.

In terms of the landscape we sketch in the main text, many of Cameron's laws advocate for particular design choices associated with the various privacy properties.

User Control and Consent and Minimal Disclosure for a Constrained Use are what we have termed *confidentiality properties*. In articulating these laws, Cameron argues that users should control attribute dissemination. In particular, identity management systems should provide users with information, such as an attribute-use policy, that enables them to make informed decisions about attribute dissemination. With the second law, Cameron also argues for mechanisms that disseminate coarser versions of attributes and for mechanisms that restrict attribute use (for example, obligations), although his work doesn't include examples of any such mechanisms.

Justifiable Parties 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 (for example, as supported by a credential-based system), because identity providers don't need 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.

Directed Identity 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 given, Directed Identity precludes the possibility of an anonymous authorization system, although such systems are consistent with the underlying motivations.

Pluralism of Operators and Technologies precludes central-

Attributes describe inherent qualities (for example, *U.age = 25*), circumstances (for example, *U.employer = Example Co.*), behaviors (for example, *U.shopping = true*), inclinations (for example, *U.likes_animals = true*), or arbitrarily assigned values (for example, *U.uid = 124*). There's no restriction on the number of attributes comprising an identity; some identities are small (for example, just a username and password), and others might contain many, possibly interdependent attributes. Identities can be created by the user or by another party acting on the user's behalf, such as an employer. Existing identity management systems don't support compound notions of identity (for example, the intersection of existing identities or a delegated identity).

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

Service Providers

Service providers authorize users on the basis of authentication assertions. The authorization decision might depend on the attributes received, the format of the authentication assertion, or properties of the party *P* that issued the authentication assertion.

Most service providers (for example, Amazon or *The New York Times*) currently 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 with multiple service providers. Therefore,

ized systems and advocates that different service providers implement different types of authentication assertions.

Human Integration—which argues that identity management systems should employ a clear, secure user interface—and Consistent Experience across Contexts—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 follow-up research, Tewfiq El Maliki and Jean-Marc Seigneur gave overviews of Project Liberty, Shibboleth, OpenID, CardSpace, and Higgins and evaluated these systems relative to Cameron's laws of identity.[2] They concluded that Higgins most closely embraces the design choices that Cameron's laws promote.

Abhilasha Bhargav-Spantzel and her colleagues took a sociological approach to identifying key privacy principles for identity management systems.[3] After discussing numerous user surveys, they emphasized that users' privacy goals differ, depending on the type of attributes and the receiving party. They argued 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 (that is, attribute inference); they critiqued CardSpace, Project Liberty, and Shibboleth with respect to the identified privacy concerns. They also argued for the development of new identity management systems that address these concerns.

Other surveys of identity management systems focus on functionality (for example, attribute namespace and propagation,[4] ease of deployment,[5] or control[6]) or economic incentives.[7,8]

References

1. K. Cameron, "The Laws of Identity," IdentityBlog, 2005; www.identityblog.com/stories/2005/05/13/TheLawsOfIdentity.pdf.
2. T. El Maliki and J.-M. Seigneur, "A Survey of User-Centric Identity Management Technologies," *Proc. Int'l Conf. Emerging Security Information, Systems, and Technologies*, IEEE, 2007, pp. 12–17.
3. A. Bhargav-Spantzel et al., "Privacy Requirements in Identity Management Solutions," *Proc. 2007 Conf. Human Interface: Part II*, Springer-Verlag, 2007, pp. 694–702.
4. W. Hommel and H. Reiser, "Federated Identity Management: Shortcomings of Existing Standards," *Proc. 9th IEEE Symp. Integrated Network Management*, IEEE CS, 2005; http://dorii.eu/pub/Publikationen/hore05a/PDF-Version/hore05a.pdf.
5. A. Myllyniemi, "Identity Management Systems: A Comparison of Current Solutions," Dec. 2006; www.tml.tkk.fi/Publications/C/22/papers/Myllyniemi_final.pdf.
6. Y. Cao and L. Yang, "A Survey of Identity Management Technology," *IEEE Int'l Conf. Information Theory and Information Security* (ICITIS 10), IEEE CS, 2010, pp. 287–293.
7. S. Koble and R. Böhme, "Economics of Identity Management: A Supply-Side Perspective," *Privacy Enhancing Technologies* (PET 05), Springer Berlin Heidelberg, 2006, pp. 259–272.
8. S. Landau and T. Moore, "Economic Tussles in Federated Identity Management," *First Monday*, vol. 17, no. 10; http://firstmonday.org/ojs/index.php/fm/article/view/4254.

accepting *U*'s authentication credentials constitutes trusting every other service provider in the system to not impersonate users or release user authentication credentials, voluntarily or involuntarily.

With identity management systems, service providers would have the flexibility to choose which parties to trust (that is, which authentication assertions to accept), and because only identity providers would manage authentication assertions, service providers would no longer need to trust other service providers.

Identity Providers

An identity provider can be implemented as a standalone party or as a component of a user or service provider. It performs two primary functions:

- Authenticating users—that is, determining whether a particular user is associated with a particular identity—and issuing authentication assertions in support of authenticated users. The manner in which an identity provider authenticates users could affect whether service providers accept that identity provider's authentication assertions.
- Storing collections of attributes for users and managing these identities. Details vary across systems, but generally, an identity provider would have provisions for creating, updating, releasing, and deleting attributes and identities. (In some systems, attribute validation, management, and storage are delegated to a separate party called the *attribute provider*.)

Because service providers depend on receiving authentication assertions that contain up-to-date 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 also be trusted to maintain currency of those attributes. The methods used to validate attributes—including whether to trust the user or to perform independent validation of claimed attributes—and to eliminate

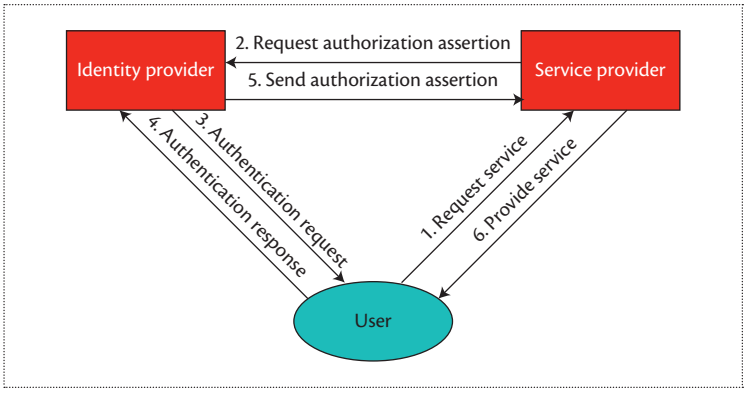

Figure 1. Typical control flow using an interactive design. Service providers interact with an identity provider during each authorization phase.

or update stale attributes often determine whether a service provider will accept some identity provider's authentication assertions.

Threat Model

Privacy properties are defined with respect to assumptions about adversaries. The adversaries could be parties inside the system, such as adversarial identity providers, or they could be external parties with access to some or all of the information available in the system, such as hackers who target identity provider databases, phishers, or snoopers looking over a user's shoulder. Adversaries 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 (including collusion between adversarial identity providers and service providers, as considered in some systems[2]), and they might exchange information or messages outside the scope of the defined protocols.

A party *A* that trusts a party *B* will believe not only that *B* is nonadversarial but also that *B* takes reasonable measures to prevent adversaries from gaining access to privileged information, including precautions to prevent or mitigate against code vulnerabilities, and that *B* responds to requests by others for such information—for example, legal requests and subpoenas—in a manner *A* deems reasonable. Trust is not necessarily permanent; *A* might cease to trust *B* if evidence emerges that indicates *B* might be untrustworthy.

Undetectability of Authorization Requests

Because authorization requests involve the user communicating with both an identity provider and a service provider, the identity provider becomes a proxy for the user. We identify three possible design choices associated with this interaction—*interactive*, *active client*, and *credential based*—and describe how they impact the authorization requests' detectability. A request is considered undetectable to a party if that party can't distinguish whether the action has occurred. Undetectability is a privacy property insofar as it concerns users' ability to conceal actions from other parties.

Of course, whether a request is detectable depends on the adversary's observational and computational powers. In this section, let's assume that the adversary performs the role of the identity provider and has no additional observational capabilities, so eavesdropping on communication between other parties and collusion with service providers are ruled out. Even if a system supports requests now being considered undetectable, an adversary might learn information about user authorization requests through other means. However, prevention of such attacks is beyond the scope of identity management systems (although such prevention is necessary for ensuring that user actions are truly undetectable).

Interactive

By storing an identity at a particular identity provider, users are trusting that it will disseminate attributes only for user-sanctioned purposes. If that trust extends to not revealing information about user actions (in particular, authorization requests), then hiding authorization requests (including context information such as which service provider and which attributes) from the identity provider is unnecessary. Identity management systems can then interactively generate authentication assertions.

Interactive systems employ a protocol in which service providers communicate with identity providers to obtain authentication assertions about users. Users might be sent a request for authentication credentials each time a service provider solicits an authentication assertion (as Figure 1 shows) or could authenticate once with the identity provider prior to issuing any authorization requests. Because service providers interact with identity providers either directly (for example, using SOAP) or indirectly (for example, using 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. Information gathered in this way can be valuable (for example, for behavioral advertising), so identity providers have an economic incentive to favor an interactive system despite the limited privacy the design offers.

Interactive systems are typically implemented in one of three ways: authentication assertions can be augmented with an HMAC, signed with a private key (presumably supported by a public-key certificate infrastructure), or validated interactively. Existing

interactive identity management systems implement one or more of these approaches. For example, Passport authentication assertions are encrypted using a shared Data Encryption Standard key that's established when service providers register with the Passport service; authentication assertions thereafter can be validated 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: identity providers can either respond to a request with a reference or Security Assertion Markup Language (SAML) artifact that must be validated interactively, or they can send a digitally signed SAML response using HTTP protocols. OpenID also supports two authentication protocols: either an identity provider–service provider pair can establish a relation by exchanging a secret key, which is subsequently used to encrypt and noninteractively validate authentication assertions, or authentication assertions can be validated interactively.

Interactive designs limit exposure to stale attributes. Because authentication assertions are issued exclusively in response to a specific request (typically enforced using nonces or time stamps), the identity provider can recheck attributes before each authentication assertion is issued. So, the window for stale assertions can be limited to the system's communication delay. However, this doesn't address problems from cached attribute values or from stale values propagated by service providers.

Active Client

Although users must trust an identity provider with attributes, that trust might not extend to arbitrary information about user actions. Active-client systems are similar to interactive designs, except users—or Web browsers acting on users' behalf—implement local functionality, including local state information or code for deciding which messages to send to which parties. Identity providers can still detect authorization requests, but the local functionality can prevent identity providers from observing context associated with an authorization request, for example, which service provider is contacted or which attributes are disseminated (see Figure 2). To adopt this approach with today's browsers, users would have to download a software extension.

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

CardSpace and Higgins implement active clients and don't disclose the service provider's identity to the identity provider that issues an authentication assertion. Client-Side Federation uses an active client together with a newly defined cryptographic construct—an invariable

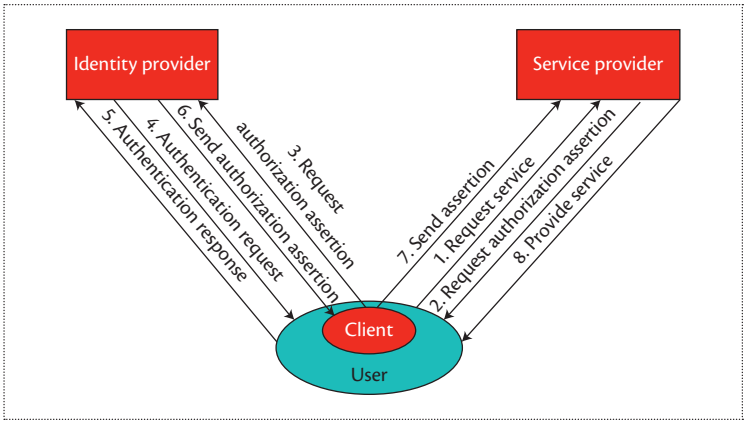

Figure 2. Typical control flow using active-client authentication. The client, acting on the user's behalf, interacts with the identity provider and the service provider, eliminating the need for direct communication between these parties while ensuring fresh authorization assertions.

partially blind signature scheme—to ensure that an identity provider doesn't learn unnecessary information about user authorization requests, even if the identity provider colludes with service providers. Project Liberty also includes the option of employing an active client; however, it 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

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 credentials, as Figure 3 illustrates. Credentials are transferrable digital artifacts—issued by identity providers to authenticated users—that convey authentication assertions. Examples of credentials include digitally signed strings,[10] X.509 certificates,[11] and SAML assertions. The subject of these authentication assertions is the user or identity to whom the credential was issued. And these authentication assertions might include attributes that uniquely identify the subject of the credential. Credentials can be presented to a service provider at any time after being 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.

Identity providers generate credentials without knowledge of which service providers will receive them, when, or how many times. Because credentials can be presented at any time after being issued and can be validated without identity providers' participation

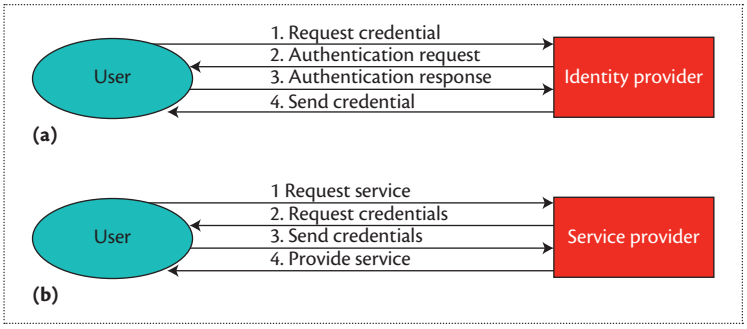

Figure 3. Typical control flow using credential-based authentication. (a) Credential issue and (b) credential presentation. Because the two phases occur independently, this design minimizes the information about authorization requests that is detectable by the identity provider.

(although interaction with a certificate authority or other components of a trust hierarchy might be required), identity providers can't detect an authorization request or observe information about the context in which the authorization request occurs. Some types of credentials let users selectively release attributes by eliminating certain attributes from a credential or by replacing attributes with coarser values—for example, replacing *U.age = 25* with *U.age > 18*.[3,5,12]

If credential transfer and delegation are undesirable, then it is necessary to ensure that credentials are validated only when presented by the user to whom they were issued. Existing identity management systems that implement credentials use cryptography to ensure that only a party with knowledge of a particular secret—presumably the subject of the credential—can present a credential in a manner that will be validated. The secret isn't revealed during the presentation of a credential. In Idemix, a user U 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 proves knowledge of secret s_c. P-IMS uses blind signatures to produce credentials; the credential itself is therefore known to the user only and can be used as the secret. To present a P-IMS credential, the user proves knowledge of a credential satisfying appropriate requirements.

Secrets alone can't prevent users from delegating credentials to other parties. Idemix employs two techniques to ensure that a credential is accepted only when the subject presents that credential: tying knowledge of a secret to an external secret with real-world value, such as 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. These techniques work because users are generally unwilling to share valuable information and, therefore, won't reveal the secret S_U. Nontechnical means, such as user contracts and legal accountability, can also discourage users from sharing credentials.

Because credentials can be presented long after being issued, identity management systems that employ credentials must address the issue of stale credentials. Several approaches have been suggested, including expiration dates and revocation lists,[13,14] but they either require users to place additional trust in another party (for example, checking a submitted credential against an identity provider–maintained stale blacklist, which could render authorization requests detectable) or require some party to incur significant delay.

Unlinkability

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

Identities Linked with Other Identities

The first question is whether users trust identity providers with knowledge about connections between different identities and, if not, how to prevent such linking. Note that providing users with unlinkable identities implicitly relies on the assumption that trusted identity providers don't collude with one another to link distinct identities. Today's identity management systems don't attempt to prevent such collusion.

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. Users who opt in must trust the identity provider and disclose all user attributes to this identity provider. Even when users choose to create multiple distinct identities (for example, personal and work identities), 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. Because 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 users choose which identity providers to trust with links between certain identities associated with them. Project Liberty supports this: identity providers with 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. Client-Side Federation is also a federated identity management system, but it's designed to ensure that local handles are known only to the user, not the identity provider, thereby guaranteeing privacy against identity providers that collude with service providers.

Decentralized. If identity providers can be trusted only with attributes that are specifically released to them and not trusted with identity linking, then the most appropriate choice is a *decentralized identity management system* in which multiple, distinct identity providers each function separately—possibly using different protocols—and might not even be aware of each other. Users can create one or more identities with any identity provider in the system. This architecture lets users not only choose which identity providers to trust with which attributes but also distribute sensitive attributes across distinct identity providers, thereby ensuring unlinkability of distinct identities. Most existing identity management systems, including Idemix, Shibboleth, Higgins, PRIME, OpenID, CardSpace, U-Prove, and P-IMS, have adopted this approach.

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. Service providers 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 beyond the scope of today's 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 opaque, globally unique identifiers. They're 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, OpenID, and Client-Side Federation. Passport credentials all contain an opaque, globally unique identifier (PUID). In Project Liberty, federated parties exchange locally unique user handles. OpenID users submit a globally unique identifier (for example, username@example.com) to the service provider to initiate authorization. Client-Side Federation uses local handles that are known only to the user and the service provider.

Anonymous authorization. If service providers aren't trusted with links between authorization requests and identities, then an identity management system can employ *anonymous authorization*. Anonymous authorization doesn't 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 ensures only that such linking can't occur unless the service provider explicitly requires such a combination of attributes.

Anonymous authorization can be implemented simply by eliminating from messages or credentials all unique identifiers that the service provider doesn't explicitly require. For example, Shibboleth supports anonymous authorization, although users can choose to reveal a persistent identifier. Project Liberty lets a service provider request an anonymous, temporary, identifier for a 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 avoiding reuse of a credential with the same service provider is sufficient to achieve anonymous authorization. Idemix and P-IMS both employ zero-knowledge proofs to let users demonstrate a credential's existence. 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.

Anonymous authorization interferes with standard notions of accountability, and many communities believe that individuals should be held accountable for unacceptable online behaviors, such as fraud or accessing illegal materials. To support both anonymous authorization and accountability, a *deanonymizing party* can be implemented. Only under certain circumstances does it 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. Identity management systems can employ cryptographic techniques to ensure that a deanonymizing party can't create conditions under which links might be reestablished, so such systems can be made resilient against deanonymizing parties that are bribed or corrupted.

In Shibboleth, identity providers observe and log all user actions, so they can act as the deanonymizing party. In P-IMS, authorization is granted to a particular persona. Identity providers that issue a credential or persona can determine the identity or persona to which it was issued; therefore, identity providers collectively perform the functionality of the deanonymizing party. Deanonymization is more difficult in Idemix, in which credential presentation consists of a zero-knowledge proof, and U-Prove, in which the issuing identity providers don't learn the credential identifier. However, users in either system can choose to include identifying information 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.

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, as with pseudonymous authorization), then by transitivity, it can link all the authorization requests issued using the same identity. If the system instead employs anonymous authorization, the service provider's ability to link multiple authorization requests to each other depends on the implementation. Existing interactive, anonymous authorization protocols (for example, Shibboleth) provide no persistent identifier across authentication assertions, so actions can't be linked to each other. To show an Idemix credential, users employ a zero-knowledge proof, so two authorization requests can't be linked, even when issued using the same credential. However, U-Prove credentials have a unique identifier, so requests containing the same credential can be linked, although requests issued with different credentials can't. And P-IMS authorization protocols require users to reveal the claimed persona, so requests issued with the same persona can be linked, although, again, requests issued with different personae can't.

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 can't 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. Because the same persona is used during the issuing protocol and the use protocol, identity providers and service providers can also establish this link in P-IMS. However, both U-Prove and Idemix implement untraceability.

Confidentiality in Identity Management

Identity management systems should let users 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 an identity provider acting on U's behalf,
- *attribute forwarding*—P acquires the attributes from another party P', and
- *attribute inference*—P deduces the attributes from other known or observed information.

Design choices here focus on how users control which attributes to release, when, and to which parties over intentional channels—those instigated by users. Users don't control attribute forwarding and attribute inference, so design choices concern whether and how to control or prevent attribute release over these channels.

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 to both direct release of attributes and dissemination of attributes by an identity provider acting on a user's behalf. Of course, the user must trust the service provider with all disseminated attributes.

Instance-based attribute release. Under this approach, users explicitly specify parties to which each attribute might be released. Instance-based attribute release is extremely flexible. However, the approach can be inconvenient because it requires users to make many decisions and because determining which service providers to trust can be difficult.

Nonetheless, many existing identity management systems provide instance-based attribute release to

control attribute dissemination to service providers. CardSpace and Higgins service providers send users a policy (described using HTML or WS-SecurityPolicy) specifying required attributes and authentication assertion types. Users must then interactively decide whether to share the requested attributes with the corresponding service providers and, if so, which available identity to use.

Depending on the identity management system's other design choices, the choice of identity could determine what information various identity providers can observe and which attributes the service providers receive. Facebook Single Sign-On presents users with permissions requested by the service providers, which users must accept or deny. Mechanisms for controlling attribute release are beyond the OpenID specification's scope, but the OpenID Attribute Exchange protocol specification does permit responses that indicate an attribute isn't available (using *attribute.count = 0*). The Google OpenID identity provider leverages this capability and gives users an option to permit or deny the release of each attribute required by a service provider, with an option to remember selected permissions for future interactions with the same service provider. Existing credential-based identity management systems, such as Idemix, U-Prove, and P-IMS, leave the approach to attribute release unspecified; some of these systems let users share either an entire credential or derived credentials that convey subsets of the attributes and coarser attribute values.

Policy-based attribute release. With this approach, users define a policy, and an attribute is released to service providers 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, the software (for example, the Web browser) will automatically determine which parties should receive specific attributes on the basis of user policies. Although policy-based attribute release is typically less expressive than instance-based release (depending on the language in which policies are written), policy-based release simplifies the user experience by automating attribute dissemination. Passport, Shibboleth, and PRIME employ policy-based attribute release.

Passport employs a very restrictive language for expressing user policies. Users tag each attribute with a label, "public" or "private." Attributes are then automatically released to other parties on the basis of these tags. Specifically, Passport sends attributes tagged "public" to any service provider with which users choose 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, which determines whether the policy rule is binding for a particular interaction, and zero or more attribute rules, which specify the attributes governed by the policy rule and whether 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.

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

Release by Attribute Forwarding

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

Deniable attributes. *Deniable attributes* can't be validated without identity provider or user cooperation. This embodies the philosophy that the significant concern about attribute release is whether parties can prove to others that 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 validated interactively only with explicit user permission (Project Liberty and Shibboleth). Systems that employ credential-based authentication can implement deniable attributes by requiring the presenting party to know a particular secret for the credential to be validated (Idemix, U-Prove, and P-IMS).

Obligations. Another approach to control attribute forwarding involves tagging released attributes with *obligations* that describe if and how the attribute can be used. For example, an obligation could describe circumstances under which the attribute may be forwarded

to a third party. To be an effective control, the tagged attributes' recipients must obey these obligations. Thus, users must trust those recipients; such trust can be developed on the basis of previous experience or reputation, assertions generated by trusted hardware, or (if deviations are detectable) 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 on 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. Legal accountability can be invoked if correct assertions aren't received.

Attribute Inference

Attribute inference occurs when a party *P* can deduce an attribute's value 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.

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[15] for database queries—the goal of which is to prevent adversaries from inferring attributes that couldn't be deduced from previously available information.

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 *P* can detect or observe as well as limiting the attributes that *P* can learn over other channels. If a party has prior access to information about a particular user action or attribute, then it's 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. Therefore, a second standard defense is to prevent linking using one of the techniques we discussed.

The Framework Applied to Public Policy

This taxonomy of design choices is not only descriptive; it has analytic value in the design of identity management systems intended to serve public policy goals. We illustrate this value using the current National Strategy for Trusted Identities in Cyberspace (NSTIC), initiated by the US federal government in April 2011.

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 document (www.whitehouse.gov/sites/default/files/rss_viewer/NSTICstrategy_041511.pdf) highlights the need for new, secure, and private solutions to enable validation, authentication, authorization, and accountability. It identifies four principles to guide identity ecosystem design; these systems should be privacy enhancing, secure and resilient, interoperable, and easy to use. We show how the design choices we described can inform the technical aspects of the design of such systems.

Noting that people currently use drivers' licenses as a real-world assertion without the Department of Motor Vehicles or the state government being aware of their actions, NSTIC recommends implementing a system in which user actions are undetectable by identity providers. Undetectable actions are also consistent with NSTIC's articulated goal of enhancing user privacy by minimizing the release of user information. As we noted, credential-based systems implement undetectable actions, but no other design choice does. Credentials are employed by all existing identity management systems that focus on anonymity—a feature that NSTIC recommends identity management systems support. Thus, we shouldn't be surprised to find that an NSTIC-compliant ecosystem explicitly embraces credential-based authentication on technical grounds.

NSTIC advocating for a credential-based system is consistent with other goals espoused by the strategy document. It stipulates 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 would best be materialized with a system that uses policy-based attribute release. Because it's possible to implement credentials that permit selective release of attributes and support arbitrary claims about attributes (for example, 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 privacy enhancement and release minimization goals imply that these techniques 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. A credential-based system can support these goals as well.

However, the NSTIC documentation asserts that an identity ecosystem should be resilient to credential loss, compromise, and theft. This goal is difficult to achieve in a credential-based system because it requires a 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 requires contact between service providers and identity providers every time a credential is validated, which is inefficient, at best. Moreover, current implementations either allow identity providers to learn information about user actions that could violate system privacy goals or rely on computationally expensive cryptographic techniques to enforce private information retrieval, further violating the efficiency requirement. 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 or requires significant computational overhead. If neither of these approaches is adopted, then the inability to revoke credentials requires users to contact all service providers to ensure that lost or stolen credentials aren't fraudulently used. Such a task is infeasible and certainly violates the ease-of-use goal. Given current capabilities, implementing a credential-based system that's efficient, easy to use, and resilient to the loss of user credentials isn't possible.

Moreover, to support flexible credentials that enable partial release of identities, users must perform some computations, such as selective release, locally. Given current Web browsers' limitations, this implies downloading and installing specialized software, a technical requirement that conflicts with NSTIC's nontechnical goals of facilitating deployment and ensuring ease of use.

However, if system designers are resigned to deploying a system that makes certain decisions locally, as well as the difficulties that accompany this choice, then they can abandon the credential-based approach in favor of an active-client system. Although this makes user actions detectable to identity providers, active-client systems can ensure that the context and details of those actions are still unobservable, a standard that appears consistent with NSTIC's stated goals. 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.

The goals NSTIC outlined 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. ■

Acknowledgments

We're grateful to Donna Dodson, Jeff Friedberg, Susan Landau, Naomi Lefkovitz, and Deirdre Mulligan for many helpful comments on an earlier version of this article. The final version of the article was also much improved from comments provided by editor Shari Lawrence Pfleeger and the reviewers. 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 N00014-09-10652, and grants from Microsoft.

References

1. E. Birrell and F.B. Schneider, *Federated Identity Management Systems: A Privacy-Based Characterization*, tech. report 30471, Computing and Information Science, Cornell University, Oct. 2012.
2. S. Canard, E. Malville, and J. Traoré, "Identity Federation and Privacy: One Step Beyond," *Proc. ACM Workshop Digital Identity Management*, ACM, 2008, pp. 25–32.
3. J. Camenisch and A. Lysyanskaya, "An Efficient System for Non-Transferable Anonymous Credentials with Optional Anonymity Revocation," *Advances in Cryptology* (EUROCRYPT 01), LNCS 2045, Springer, 2001, pp. 93–118.
4. J. Camenisch and E. Van Herreweghen, "Design and Implementation of the Idemix Anonymous Credential System," *Proc. 9th ACM Conf. Computer and Communications Security* (CCS 02), ACM, 2002, pp. 21–30.
5. M. Hussain and D. Skillicorn, "Persona-Based Identity Management: A Novel Approach to Privacy Protection," *Proc. 13th Nordic Workshop Secure IT Systems* (WWSecIT 08), Technical Univ. of Denmark, 2008, pp. 201–212.
6. D. Skillicorn and M, Hussain, *Personas: Beyond Identity Protection by Information Control*, A Report to the Privacy Commissioner of Canada, Mar. 2009.
7. A. Pfitzmann and M. Hansen, *A Terminology for Talking about Privacy by Data Minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management*, tech. report, TU Dresden, Apr. 2010.
8. M. Deng et al, "A Privacy Threat Analysis Framework: Supporting the Elicitation and Fulfillment of Privacy Requirements," *Requirements Eng.*, vol. 16, no. 1, 2011, pp. 3–32.

9. Committee on Authentication Technologies and Their Privacy Implications, *Who Goes There?: Authentication through the Lens of Privacy*, S.T. Kent and L.I. Millett, eds., Nat'l Academies Press, 2003.
10. L.M. Kornfelder, "Towards a Practical Public-Key Cryptosystem," bachelor's thesis, Dept. Electrical Engineering, Massachusetts Inst. Technology, 1978.
11. C. Adams and S. Farrell, *Internet X.509 PKI Certificate Management Protocols*, RFC 2510, Internet Eng. Task Force, Mar. 1999.
12. D. Bauer, D. Bloguh, and D. Cash, "Minimum Information Disclosure with Efficiently Verifiable Credentials," *Proc. 4th ACM Workshop Digital Identity Management*, ACM, 2008.
13. S. Brands, L. Demuynck, and B. De Decker, "A Practical System for Globally Revoking the Unlinkable Pseudonyms of Unknown Users," *Proc. 12th Australasian Conf. Information Security and Privacy* (ACISP 07), Springer, 2007, pp. 400–415.
14. J. Lapon et al., "Analysis of Revocation Strategies for Anonymous Idemix Credentials," *Proc. 12th Int'l Conf. Communications and Multimedia Security*, Springer-Verlag, 2011, pp. 3–17.
15. C. Dwork, "Differential Privacy," *Automata, Languages and Programming*, LNCS 4052, Springer, 2006, pp. 1–12.

Eleanor Birrell is a PhD student in computer science at Cornell University. Her research interests include principles of systems security and theoretical cryptography. Birrell received a BA in computer science and mathematics from Harvard University. Contact her at eleanor@cs.cornell.edu.

Fred B. Schneider is the Samuel B. Eckert Professor of Computer Science at Cornell University. His research interests include trustworthy computing systems. Schneider received a Doctor of Science honoris causa from the Newcastle University upon Tyne. He's a member of the US National Academy of Engineering and a foreign member of its Norwegian counterpart (NTKV). Schneider is a Fellow of AAAS, ACM, and IEEE. Contact him at fbs@cs.cornell.edu.