A Proposed Architecture for the NSTIC Ecosystem

Francisco Corella, PhD
fcorella@pomcor.com

Karen Lewison, MD
kplewison@pomcor.com

July 17, 2011

Abstract

The National Strategy for Trusted Identities in Cyberspace (NSTIC) aims at creating an Identity Ecosystem by deploying alternatives to passwords that will improve security while at the same time increasing privacy and user choice. The last decade has seen the deployment of third-party login solutions, which reduce the use of passwords but also reduce privacy. The last decade has also seen the development of privacy-enhancing credentials, but they have yet to be successfully deployed.

To maximize the chances of success of NSTIC, we propose an architecture where privacy-enhancing technologies are built into the fabric of the Web, so that they can be used with little or no effort by users, relying parties, identity providers, and social sites. Specifically, we propose extensions to the two core protocols of the Web, TLS and HTTP, that will facilitate the use as well as the automated issuance of credentials, including privacy-enhanced credentials, PKI certificates, and bare-key credentials for session tracking. We describe a range of use cases to illustrate how the proposed architecture will meet the goals of NSTIC.

1 Introduction

The National Strategy for Trusted Identities in Cyberspace (NSTIC) [1] aims at creating an Identity Ecosystem by deploying alternatives to passwords that will improve security while at the same time increasing privacy and user choice [2, 3, 4].

The last decade has seen the deployment of several Internet identity[1] solutions, including Microsoft Passport [5] (now Windows Live ID), SAML Web Browser SSO Profile [6], Shibboleth [7], OpenID [8] and OAuth [9]; OAuth was originally intended for authorization rather than authentication, but is used by Facebook and other social sites for social login. These solutions reduce the use of passwords, but they take a step back in terms of privacy, because they rely

[1]We use the phrase "Internet identity" to mean "Web identity", as is usually done.

on online third-party identity providers that learn how the user uses the credentials they provide. These solutions also reduce security by facilitating phishing attacks [5, 10] and, sometimes, by allowing credentials to be sent in the clear.[2]

Third-party identity solutions where the identity provider is offline provide more privacy because the identity provider is not informed of how its credentials are used. This is true in particular of PKI certificates. But use of a certificate can be linked to its issuance if the issuer shares information with the relying party, and multiple uses of the same certificate can be linked if the relying parties share information among themselves.

Privacy-enhanced (PE) credentials, such as IBM's Idemix anonymous credentials [12] and Microsoft's U-Prove tokens [13, 14] provide additional privacy by preventing linkage between issuance and use of a credential, and by allowing selective disclosure of information from a credential. Idemix anonymous credentials go a step further by also preventing linkage between multiple uses of the same credential.[3]

PE credentials have never been successfully deployed. The Idemix system exists only in the form of a software library. The U-Prove system was deployed within the Information Cards framework, but Information Cards were then discontinued [15]. One reason why Information Cards failed is that they required users and relying parties to install new software and learn how to use it [16, 17].

To maximize the chances of success of NSTIC, we propose an architecture where privacy-enhancing technologies are built into the fabric of the Web, so that they can be used with little or no effort by end-users and by developers of relying parties, identity providers and social sites.

Specifically, we propose extensions to the core protocols of the Web, TLS [18] and HTTP [19]. We describe these extensions in Section 2. In Section 3 we explain how user accounts and user sessions are created and maintained in the proposed architecture. We briefly describe a variety of use cases in Section 4, and we recapitulate in Section 5. Section 4 can be read first if desired.

Before explaining the proposed extensions to Web protocols, we must explain how we use some vocabulary and describe the implementation of *trustmarks* in the proposed architecture.

1.1 Vocabulary

We use the term "identity provider" in a broad sense to refer to any credential issuer, whether or not the credentials it provides uniquely identify the user in a particular context. Others might use instead the term "attribute provider" in this sense.

Similarly, we use the term "identity data" to refer to a collection of attributes provided by an identity provider, whether or not they uniquely identify the user.

[2]For example, the Facebook developer documentation [11] uses the *http* URL scheme rather than *https* in examples of OAuth callback URLs. Use of *http* means that the OAuth authorization code is sent in the clear. In social login, the authorization code functions as a credential, and sending it in the clear allows user impersonation.

[3]U-Prove tokens do not provide this feature, as explained in [13, Section 4.2].

1.2 Trustmarks

The NSTIC strategy document [3] envisions trustmarks that are verified and displayed by the Web sites of identity providers and relying parties, but also mentions trustmarked credentials.

Therefore trustmarks play two roles:

1. They provide trust for Web sites of relying parties and identity providers, and

2. They provide trust for credentials issued by identity providers.

The first role can be played by a traditional TLS server certificate that binds a public key to the server's domain name, the identity of the organization that owns the server, and NSTIC-specific data such as a level of assurance. The certificate will be signed by a NSTIC accreditation authority, playing in this case the role of a certificate authority (CA), and backed by a certificate issued to the accreditation authority by a root certificate authority (CA) set up by NSTIC ("the NSTIC Root CA"). Browsers will recognize the NSTIC-specific data in the certificate and the NSTIC root CA, and will display a visual trustmark in the "browser chrome", i.e. in a trusted area of the browser window, not modifiable by the server. This is equivalent to how browsers indicate today that a server certificate is an "Extended Validation" certificate.

The second role can be played by a credential-issuance certificate that can be used to back credentials issued by the identity provider. The credential-issuance certificate will bind the public key component of a credential-issuance key pair to NSTIC-specific data, and will be signed by an NSTIC accreditation authority. Notice, however, that the credentials issued by the identity provider may be PE credentials, in which case the credential-issuance key pair will not be a traditional key pair, and the credential-issuance certificate will not be a traditional PKI certificate, because it will contain a non-traditional public key.

An identity provider may thus need two different trustmarks, one for each role. In the special case where the credentials issued by the identity provider are ordinary PKI certificates, a single dual-use PKI certificate could play both roles, but even in this case it is probably easier to use a different certificate for each role.

2 Proposed Extensions to Web Protocols

2.1 Extensions to TLS

TLS is a client-server protocol universally used to implement secure connections between Web browsers and Web servers playing the roles of TLS clients and TLS servers respectively.[4]

[4]Use of TLS is triggered by the *https* URL scheme. There is no security on the Web without TLS, so in this paper we are only concerned with connections between Web clients and Web servers that are protected by TLS.

TLS begins with the "TLS handshake", where client and server perform a key exchange to set up a shared master secret. After the handshake, an underlying "TLS record layer" uses symmetric keys derived from the master secret to secure the connection, encrypting the traffic and providing integrity protection.

The record layer carries different "content types", including handshake messages and application data. We propose to extend it so that it carries two additional content types, intended for a proposed optional message exchange initiated by the client, which can be used for client authentication, and a proposed optional message exchange initiated by the server, which can be used to issue credentials to the client.

The proposed exchanges take place after the handshake and are thus encrypted and integrity protected. By authenticating during the client-initiated exchange, the client's credentials will be sent encrypted, whereas today, if the client authenticates with a client certificate, the certificate is sent in the clear.[5]

The proposed client-initiated exchange may take place immediately before an HTTP Request carried by the TLS connection, or it may be interleaved with the HTTP Request data. The proposed server-initiated exchange may take place immediately before an HTTP Response carried by the TLS connection, or may be interleaved with the HTTP Response data. Interleaving of different content-types is supported by the TLS record layer and improves performance for exchanges that include roundtrips between client and server.[6]

2.1.1 Client-Initiated Exchange

The client can use this exchange to:

1. Authenticate with a PKI certificate, as is done today but with the certificate being sent encrypted.

2. Authenticate with a PE credential, such as an Idemix anonymous credential or a U-Prove token, disclosing only necessary information.

3. Authenticate with an uncertified key pair used as a session credential as explained below in Section 3.2.

4. Authenticate with multiple credentials, possibly of different kinds, by sequentially executing a credential presentation protocol for each credential within the exchange.

The client can also use the exchange to:

5. Issue a delegated authorization credential to the server.

[5]The current authentication mechanism is to be preserved for backwards compatibility, and to continue supporting rare certificates containing Diffie-Hellman public keys that are used in the key exchange and must therefore be sent to the server before the key exchange.

[6]Rountrips take place in the server-initiated exchange, but usually not in the client-initiated exchange. No roundtrips are required for user authentication with a PKI certificate, an Idemix anonymous credential, or a U-Prove token.

Using the same exchange for authentication and delegated authorization is motivated by the fact that user authentication is a by-product of delegated authorization in social login, as explained in Section 4.8.

2.1.2 Server-Initiated Exchange

The server can use the server-initiated exchange to issue one or more credentials to the browser,[7,8] by executing protocols such as:

1. A protocol for issuing an Idemix anonymous credential [12, Section 6.1.1].
2. A protocol for issuing a U-Prove token [14, Figure 6].
3. A protocol for issuing a PKI certificate for user authentication.[9]
4. A protocol for issuing a delegatable credential such as:
   (a) A delegatable anonymous credential [20], or
   (b) A PKI certificate for delegating authorization.[10]

An issuance protocol can be used to replace a credential present in the browser with a credential having the same *name*. This requires a credential nomenclature where credential names refer to specific credential issuers. Such a nomenclature can be achieved by using URLs as names.

An issuance protocol can also be used to remove a credential from the browser, by the convention of issuing to the browser a zero-length credential with the same name.

The ability to issue credentials within TLS can be used to automate cumbersome enrollment processes, as illustrated below in Section 4.4.

[7]When it issues a credential, the server sends to the browser an entire credential chain, ending in a root credential.

[8]Reasons for issuing multiple credentials simultaneously include: issuing alternative credentials based on different technologies for the same identity data; and issuing separate PKI certificates for separate attributes as an alternative to issuing a PE credential for all the attributes.

[9]The certificate is issued as follows: the browser generates a key pair and sends the public key to the server, together with a signature computed with the private key on a random string derived from the TLS Master Secret; the server verifies the browser's signature, creates and signs a certificate for the public key, and sends the certificate to the browser.

[10]The key type of a certificate for delegating authorization must support digital signature, and the certificate must have the keyCertSign extension [21]. The certificate is issued as described in the previous footnote. The browser later uses the private key to sign PKI certificates that bind a public key supplied by a delegate party to a delegated authorization to access an account or other resource owned by the user. The public key supplied by the server can be the one contained in the delegate party TLS server certificate.

2.2 Extensions to HTTP

2.2.1 Extension for Requesting Credential Submission and Delegated Authorization

HTTP is the protocol that specifies the messages exchanged between a Web browser and a Web server. It includes a rudimentary authentication mechanism [22] that allows the server to ask for credentials; in response to which the browser prompts the user for a username and a password, uses them to authenticate to the server, and stores them for future use. The facility does not include any session management features, and it is rarely used. However it embodies sound principles: the browser, as trusted user agent, should be responsible for storing credentials and submitting them to relying parties; and the mechanism by which relying parties request credentials should be built into HTTP to ensure interoperability.

We propose an HTTP extension, built on these sound principles, that allows the server to ask the browser to submit one or more credentials of diverse types, inclucing PKI certificates and PE credentials such as Idemix anonymous credentials and U-Prove tokens. As in the existing authentication mechanism [22], the requests for credentials are triggered by an HTTP Request Message (sent by the browser to the server), and are specified in an HTTP Response Message (sent from the server to the browser). Each request is encoded in a header of the HTTP Response Message, in a manner to be determined.

Requested credentials may include any of the following:

- A particular credential from a particular identity provider, e.g. a social security number from the Social Security Administration; the credential is identified by a URL used as a credential name, as we saw in Section 2.1.2. An important special case is a credential issued by a relying party when the user registers, and later submitted to that same relying party when the user logs in.

- A particular attribute from an accredited provider of that attribute, e.g. the fact the user is 21 years old from an identity provider accredited to issue proof of age; the attribute is identified by an attribute name, with an attribute nomenclature to be determined.[11] In that case, the browser must present a credential chain consisting of the attribute credential, a trustmark issued to the attribute provider by an NSTIC accreditation authority and a certificate issued to the accreditation authority by the NSTIC Root CA.

- A self-asserted attribute provided by a *personal data site* as described below in Section 4.7.

- An uncertified key pair used as a session credential as explained below in Section 3.2.

[11] An example of a very simple attribute nomenclature can be found in [23].

In addition to, or instead of, requesting credentials to be submitted by the browser to the server, the server can ask for:

- Issuance of a delegated authorization credential by the browser to the server. Examples of this can be found in the use cases of Section 4.5 and Section 4.8.

The browser complies with requests for credentials and/or delegated authorization by resending the HTTP Request Message that triggered them over a new TLS connection. The requested credentials and/or delegated authorization are presented by the browser in the client-initiated exchange of the new connection.

The browser is responsible for asking the user's permission before sending identity data to the server or delegating authorization, when appropriate.

2.2.2 Extension for Session Initiation and Termination

We also propose an HTTP extension for initiating and terminating browser-side sessions. The extension defines a session-initiation HTTP header, which specifies the duration of the session and whether it should survive a browser restart.[12] Like the HTTP header used today to set cookies, the session-initiation header also specifies the range of URLs that participate in the session, the range being specified by a DNS domain suffix and a path prefix.

Browser-side sessions are implemented using uncertified key pairs as session credentials as described below in Section 3.2.

The extension also defines a session-termination header, which instructs the browser to terminate the session whose credential can be found in the current TLS connection.

3 Account Login and Session Tracking

The proposed architecture includes two special credentials: *login certificates* used for account login, and *session credentials* used for session tracking.

3.1 Account Login

To authenticate a user who logs in to an existing account, the relying party may use a PKI certificate, which we call a *login certificate*, issued by the relying party itself. The login certificate binds the public key component of a browser-generated key pair to a unique identifier of the account internal to the relying party; a cryptographic hash of the public key can serve as the identifier. Because the login certificate is submitted to its issuer, no certificate chain and no revocation service are needed.

[12]For sessions that do not survive restart the duration can be omitted, in which case, by convention, the session lasts until the browser exits.

The relying party can choose to place sensitive data in the certificate instead of storing it in a database record; sensitive data will thus be out of reach of an attacker who succeeds in breaking into the database.

Since the login certificate is issued by the relying party and contains an identifier internal to the relying part, it raises no privacy issues, and there would be no privacy advantage in using a PE credential instead of a PKI certificate.

The relying party issues the login certificate to the browser when the user registers and the account is created, at which time the user may present third-party credentials. The relying party may choose to request at login time some of the credentials presented at registration time, in addition to the login certificate, if the attributes supplied by those credentials may have changed. We will see in Section 4.7, and again in Section 4.8, how the login certificate may be useful even when presented in addition to third party credentials that uniquely identify the user.

In the proposed architecture the user has many login certificates, one for each account, but does not need to be aware of them, because they are issued and presented automatically. The browser does not have to ask the user for permission before presenting a login certificate to a relying party, because the login certificate is presented to the same party that issued it. The user logs in to an existing account with a single click on a login button.

3.2 Session Tracking

Because HTTP is a stateless protocol, every HTTP request pertaining to a session has to be authenticated. In the traditional Web architecture, this is achieved with authentication cookies. In the proposed architecture it is achieved more securely and, arguably, more efficiently, using what we call *session credentials.*

A session credential is a key pair generated by the browser whose public key component is presented bare rather than embedded in a certificate; we refer to such a credential as an *uncertified key pair* or a *bare-key credential.* The relying party uses a cryptographic hash of the public key as a reference to a session record.

A *registered-user session* is created when a registered user logs in to her account. A *guest-user session* is created when a guest user, who does not have an account, initiates a series of related transactions; shopping cart software, for example, often creates guest-user sessions.

To create a registered-user session, the server asks the browser to present a session credential together with a login certificate and/or third-party credentials. To create a guest-user session, the server asks for a session credential and possibly third-party credentials, but no login certificate. In either case, the browser establishes a new TLS connection to the server, and submits the session credential in the client-initiated exchange of the new connection, sending the public key, and proving ownership of the credential by sending a signature computed with the private key on a pseudo-random string derived from the

TLS Master Secret.[13] The browser submits only one session credential per TLS connection.

After receiving and verifying the session credential, the server computes a cryptographic hash of the bare public key and creates a session record indexed by the hash.[14] Then it instructs the browser to create a browser-side session, using the session-initiation HTTP header described in Section 2.2.2.

While the session is in effect, the browser submits the session credential each time it opens a new TLS connection that targets a URL that participates in the session, without being asked by the server. To authenticate the browser, the server validates the session credential, computes a cryptographic hash of the bare public key, and verifies that there is a session record indexed by the hash.

Notice that multiple HTTP requests can be sent over the same TLS connection. A session credential will only be sent to a participating URL when a new connection is established to that URL, whereas today an authentication cookie must be sent with every HTTP request carried by the connection.

4 Use Cases

In all the use cases, user credentials are stored in a credential repository that is part of the browser or in a device accessible to the browser. This is no different from how TLS client certificates are stored today. Examples of devices accessible to the browser are CAC cards [24] and PIV [25] cards. Browser and device vendors will be able to compete on ways of protecting the credentials, backing them up, and syncing credential repositories between browser instances on different machines.

4.1 Internet Shopping

The NSTIC strategy document [3, "Envision It" no. 1] suggests using digital cash for Internet shopping. Digital cash is an idea that should be pursued, but it is arguably out of scope for NSTIC, since cash decouples payment from identity. Furthermore, payment systems and protocols are difficult to set up: it took many decades for the credit card concept to mature; and in the nineties, the Secure Electronic Transaction (SET) payment protocol [26], which promised increased security for Internet payments, failed despite strong industry support.

We propose an incremental approach to securing Internet purchases by credit card, fully within the scope of NSTIC, that does not require any modification of the existing credit card payment system, but nevertheless has the potential to reduce or eliminate credit card fraud over the Internet. The approach could be used in the short term while a digital cash system and/or some other alternative

[13]The user must be required to prove ownership of the session credential, at least in the case of a registered-user session. Otherwise an attacker might be able to submit a victim's public key, causing the victim to unwittingly access the attacker's account, where the victim may enter sensitive data that becomes available to the attacker.

[14]When creating a registered-user session, the server places a reference to the user's account in the session record.

to the current credit card payment system are developed. It would arguably yield the biggest payoff for NSTIC in terms of fraud reduction.

In the proposed approach the user has an ordinary credit card, supplemented by a PE credential, issued to the user's browser by the credit card issuing bank, whose attributes are credit card data items.[15] At the checkout page, the user clicks on a button to pay by credit card, without manually entering any credit card data. The merchant's site asks for credit card data attributes, with the restriction that the card-type attribute must be one of those accepted by the merchant (e.g. either Visa or MasterCard).[16] The browser uses the PE credential to supply (only) the requested attributes. Then the merchant's shopping cart software processes the payment as it would if the user had manually entered the credit card data into a Web form.

Usage of the credit card credential provides two important benefits.

First, stolen credit card data cannot be used at the merchant site, unless the thief has also obtained the private portion of the credit card credential, which is stored in the user's browser. The merchant is therefore much less exposed to chargebacks. If all online merchants require the use of credit card credentials, stolen credit card data can no longer be used by itself on the Internet.

Second, the user does not have to manually enter the credit card data. Therefore there is less reason for the merchant to store the data, which reduces the danger of the data being stolen if the merchant's site is broken into.

4.2 Shopping for Wine with Proof of Age

The user wants to purchase wine from an online merchant. The merchant needs proof that the user is at least 21 years old.

At the checkout page, when the user clicks a button to make the purchase, the merchant's site simultaneously requests a credit card credential as in Section 4.1 and an attribute asserting that the user is at least 21 years old. To satisfy the latter request, the browser uses a PE driver's license credential, disclosing only the requested fact that the user is 21 years old, derived from the date of birth attribute in the credential. This functionality is supported by Idemix anonymous credentials [12, Section 6.2.5].[17]

The driver's license credential is issued by the department of motor vehicles of the state where the user resides (which may be different from the merchant's state). The browser backs the driver's license credential with a trustmark issued by a NSTIC accreditation authority asserting that the department of motor vehicles of the user's state is accredited to issue proofs of age, itself backed with a PKI certificate for the accreditation authority signed by the NSTIC Root CA.

[15]The privacy gained in this particular case from a PE credential is limited because the credit card payment system, which we do not want to modify, requires the parties involved to have substantial information about each other. Therefore a PKI certificate could be used instead of a PE credential without much relative loss of privacy.

[16]To support this, the HTTP extension of Section 2.2.1 must allow the server to specify a logical disjunction of acceptable attributes.

[17]U-Prove tokens allow selective disclosure of attributes but no inequality proofs [13, Section 4.3].

4.3 Opening a Bank Account with Government-Issued Credentials

Banks have the obligation to know their customers; they have therefore strict identity proofing requirements. When a customer opens an account, she has to supply a social security number; a driver's license which the bank photocopies and from which it obtains the user's name, address, birth date and other data; and a self-asserted email address.

The proposed architecture makes it possible to conduct the proofing online using government-issued credentials. The user fills out a form requesting creation of the account and providing the self-asserted email address. When the user submits the form, the bank's site requests two specific credentials: a social security credential asserting the user's name and social security number, issued by the Social Security Administration; and a driver's license credential, issued by the department of motor vehicles of the state where the user is opening the account. The bank's site asks for all data in both credentials. (If either credential is a PE credential, the browser discloses all the attributes asserted by the credential.)

Upon successful verification of the credentials and the email address, the bank issues a login certificate to the browser as described above in Section 3.1. Later, when the user clicks on the login button at the bank's site, the site asks for the login credential, but does not ask again for the social security number and driver's license credentials. If the bank requires two factor authentication, the login certificate and associated private key may be stored in a smart card that requires a PIN for activation; or the user may manually enter a one-time password.[18]

Notice that the bank relies on third parties for authentication at registration time, but not at login time; and the only third parties it relies on are government identity providers. This should alleviate or dispel the liability concerns that banks may have when they consider whether to participate in the NSTIC Ecosystem.

4.4 Government Agency as Relying Party

The US Patent and Trademark Office (USPTO) currently issues PKI certificates that allow patent lawyers, patent agents and *pro se* inventors to file patent applications online. To obtain a certificate, the user has to fill out and print a form stating the user's name, address, telephone number, email address, and one or more USPTO customer numbers that the user wants to associate with the certificate.[19] The user must have the form notarized and mail it to the USPTO. The USPTO returns two "access codes": an "authorization code" by

[18]The one-time password may be supplied by a synchronized device, or may sent by the bank to the user's cell phone in a text message. However the latter does not constitute a second factor if the use is using that same cell phone to access her bank account.

[19]The same person may conduct business with the USPTO under multiple customer numbers.

email and a "reference number" by US mail or by telephone. The user uses the authorization code and reference number to request the certificate at the USPTO Web site. The USPTO generates a key pair on behalf of the user,[20] issues a certificate for the public key, and creates a file containing the certificate and the private key. The file is encrypted under a password and downloaded to the user's machine. To use the certificate the user has to use a Java applet downloaded from the USPTO Web site, overriding a browser warning about the certificate used to sign the applet.

With the proposed architecture, this cumbersome process is avoided as follows.

The user fills out the certificate request form online, but may leave any of the fields blank, except that she must enter at least one customer number. When she clicks on the form submission button, the USPTO site simultaneously requests the following credentials:

1. A credential asserting an "official name" attribute. The browser may present a PE driver's license credential issued by a state, disclosing only the name; as in the previous use case the driver's license is backed by an accreditation chain consisting of a trustmark issued by a NSTIC accreditation authority asserting that the department of motor vehicles is accredited to issue proofs of age and a PKI certificate for the accreditation authority signed by the NSTIC Root CA. Or the browser may present a PE passport credential issued by the State Department, in a similar fashion.

2. If the user did not enter an address, a credential for a self-asserted "address" attribute.

3. If the user did not enter an email address, a credential for a self-asserted "email address" attribute.

To supply the address and/or email address attributes the browser uses a PE credential supplied by a personal data site as described below in Section 4.7, disclosing only the needed attributes.

Upon successful submission of the credentials the USPTO issues the PKI certificate to the browser automatically. The PKI certificate is a login certificate, issued and used as described above in Section 3.1. When the user logs in, the certificate is presented to the USPTO site by the browser rather than by a Java applet.

4.5 Doctor Gains Access to Patient's Electronic Medical Record

During a patient visit, the doctor may need to access the patient's medical records at a hospital with which the doctor is not affiliated, with permission

[20]It is not a best practice for a certificate issuer to generate a key pair on behalf of a party that applies for a certificate. The applicant should generate the key pair and have the public key certified by the issuer.

from the patient. Today, the process of obtaining permission and retrieving records is a paper process that takes substantial time. The patient may have to schedule a subsequent visit to allow time for the doctor to obtain the records.

With the proposed architecture, the patient grants permission and the doctor obtains the records electronically without delay, during the initial patient visit. This is accomplished as follows.

The doctor has a medical license credential issued by the state's medical licensing agency. The patient has a delegatable credential issued by the hospital, stored in the credential repository of her smart phone's browser.

During the visit, the patient uses her smart phone to access the doctor's computer system. The system requests a delegated authorization credential to access the patient's medical records. The browser in the smart phone uses the delegatable credential to issue the delegated credential to the doctor's system.

The doctor can then access the hospital's site and request access to the patient's medical records, authenticating with two credentials: the doctor's medical license credential, plus the delegated credential obtained from the patient.

4.6 User Authentication without Third-Party Involvement

Most Web sites still ask users to register by entering self-asserted identity data into a registration form and choosing a username and a password that are used to authenticate subsequent logins; no third party is involved. This is the most frequent use of passwords on the Web. Providing a password-free alternative authentication method for such sites would arguably yield the biggest payoff for NSTIC in terms of reduction of password use.

The proposed architecture provides such an alternative.

The user visits the registration page of the site and enters self-asserted data in the registration form, including a name or pseudonym, but no username and no password. When the user submits the form, the site creates an account for the user and issues a login certificate to the browser, which contains an account identifier internal to the site, as explained above in Section 3.1.

Later the user logs in simply by clicking on a login button, without having to enter username and password. When the user clicks on the button, the site requests the login certificate issued by the site itself. The browser submits the certificate. If the certificate refers to a valid account, the site logs the user in and creates a session as described above in Section 3.2.

As in the traditional password-based solution, no third-party is involved and use of the login certificate raises no privacy issues.

4.7 Using a Personal Data Site

In the use case of Section 4.6 the user has to provide self-asserted identity data to each site, and must remember to keep that data up to date in each site. Several *personal data sites* have recently emerged that allow users to store and maintain personal identity data for various purposes [27]. It would be natural for some of those sites to serve as identity providers for self-asserted identity data.

In the proposed architecture, this is accomplished as follows.

The user creates an account at the personal data site without third-party involvement as explained above in Section 4.6, and stores self-asserted identity data at the site. The personal data site uses the TLS server-initiated exchange described above in Section 2.1.2 to issue a PE credential to the user's browser containing an attribute for each self-asserted data item. There may be attributes such as name, nickname, email address, postal address, date of birth, URL pointing to an avatar, etc. Later, when using the PE credential, the browser discloses only those attributes or derived information requested by the relying party. When the user modifies the self-asserted data at the personal data site, the site automatically issues an updated PE credential to the browser as explained in Section 2.1.2.

It is possible to use different personae for different purposes, as advocated in [28], either by using multiple personal data sites, or by setting up multiple profiles in one personal data site, if this is supported by the site. Browsers should allow the user to easily switch from one persona to another by changing a default PE credential for self-asserted identity.

The PE credential may be used differently by different relying parties. Some relying parties may create a user account and issue a login certificate to be used in conjunction with the PE credential. Some may not create an account, but may create guest-user sessions tracked by session credentials. We describe each case separately.

4.7.1 Relying Party Creates User Account

The relying party may create an account if, for example, it provides a service to the user that involves storage of user data such as documents or media.

Such a relying party provides a button for registering with self-asserted identity data provided by a personal data site. When the user clicks on that button, the relying party asks the browser for specific self-asserted attributes. The browser uses the PE credential provided by the personal data site to supply the attributes requested by the relying party.

Then the relying party creates an account for the user, populates it with the identity data in the supplied attributes, and issues a login certificate to the browser, as explained in Section 3.1. Later, when the user logs in by clicking on a login button, the relying party requests the login certificate, as well as the same attributes that it requested for registration. The login certificate is used to uniquely identify the user's account. The attributes are used to update the identity data in the user's account with fresh data from the personal data site.[21] Notice however that the personal data site does not need to be online, since the data comes from the PE credential kept by the browser in its credential

[21]Alternatively, the relying party may choose not to store the identity data, counting on the fact that it will be supplied each time the user logs in.

repository. The data is fresh because a fresh credential is issued to the browser whenever the user modifies the data at the personal data site.

Upon successful login, the relying party creates a registered-user session and instructs the browser to create a browser-side session, as described above in Section 3.2.

It would be a mistake for the relying party to identify the user's account on the basis of a unique identifier provided by the personal data site instead of issuing its own login certificate for that purpose. Omitting the login certificate would make the relying party dependent on the personal data site. In particular, if the personal data site went out of business, users would lose access to their relying party accounts as credentials expired and could not be renewed. By contrast, use of the login certificate for account identification makes it possible for the user to switch to a different personal data site and for the relying party to seamlessly obtain the same attributes from the different site.

4.7.2 Session but no Account: the Shopping Cart Use Case

A party that relies on self-asserted identity data provided by a personal data site may not need to create a user account, but may still need to create a guest-user session to link together a sequence of related transactions. As an example, consider the case where the relying party is a shopping site that uses shopping cart software.

The shopping cart is activated when the user starts shopping. At that time, the relying party asks for a fresh session credential as described above in Section 3.2. Later the user proceeds to the checkout page. At that time the relying party needs personal data. It asks for the attributes it needs, and the browser uses the PE credential obtained from the personal data site to provide only those attributes.[22] The relying party stores the personal data in the session record that it created earlier.

4.8 Social Login

Social login is the process, pioneered by Facebook Connect, whereby a relying party delegates user authentication to a social site, and obtains authorization to access the user's account at the site. Authentication is actually a by-product of authorization: the relying party obtains the user's identity by accessing the user's account.

Social login is usually implemented using the OAuth protocol. Like other identity solutions deployed today, OAuth uses a double redirection mechanism: the user is redirected by the relying party to the social site, which authenticates the user and redirects her back to the relying party. The social site is thereby

[22]As noted above in Section 2.2.1, the browser is always responsible for asking the user for permission, when appropriate, before providing identity data to the relying party. In this case, the browser will ask for permission unless the user has previously visited the relying party and given the browser longstanding permission to provide the data that is now being requested to the relying party.

informed of the identity of the relying party and can trivially track the user. Furthermore, contrary to identity solutions such as OpenID, OAuth requires the relying party to register with the social site. This reduces user choice, by limiting her social login options to those sites that the relying party has registered with; and it makes the relying party dependent on the social site, which may, at its sole discretion, revoke the relying party's registration.

Our proposed architecture makes it possible to implement social login without these drawbacks.

When the user registers with the social site, the site automatically issues a delegatable credential to the user's browser.[23]

Broadly speaking, the browser uses this credential to issue a delegated authorization credential to the relying party, and the relying party uses the delegated authorization credential to access the user's account at the social site, where it obtains user identity data.[24] The relying party does not have to register with the social site, and the social site is not informed of the identity of the relying party. The social site may try to identify the relying party through the source IP address of the incoming TCP connections, or by parsing any "updates" that the user may publish concerning her activities at the relying party. However that would be an egregious privacy violation, which could be ruled out, if necessary, by regulation. By contrast, today the social site *must* know the identity of the relying party because it is responsible for asking the user for permission to provide the relying party with access to the user's account.

As when using a personal data site as identity provider, the details of the social login procedure depend on whether the relying party creates a user account and registered-user sessions, or only creates guest sessions.

4.8.1 Relying Party Creates User Account

If it creates a user account, the relying party features a button for the user to register with her identity at her preferred social site. When the user clicks on the button, the relying party asks the browser for a delegated authorization credential. The browser looks for a delegatable credential from a social site in its credential repository, and uses it to issue the requested delegated credential during the TLS client-initiated exchange described above in Section 2.1.2. The relying party uses the delegated credential to obtain identity data from the user's account at the social site. Then it creates a user account which it populates with the identity data, and issues to the browser a login certificate that uniquely identifies the account, as explained above in Section 3.1.

Subsequently the user logs in by clicking on a login button. The relying party then requests the login certificate as well as the delegated authorization credential. It uses the login certificate to locate the user's account, and the

[23]As mentioned in Section 2.1.2, the delegatable credential can be a PKI certificate, whose associated private key is used to sign another PKI certificate that grants access to the user's account at the social site, or a delegatable anonymous credential [20].

[24]In addition to obtaining identity data, the relying party may use the user's account, e.g., to issue "updates" on behalf of the user. This is what is making social login so successful.

delegated authorization credential to obtain fresh identity data from the social site, which it uses to update the user's account as needed.[25] Then the relying party creates a registered-user session as described in Section 3.2.

Use of the login certificate issued by the relying party itself to identify the user's account at the relying party breaks the dependencies on the social site created by the today's social login solutions. First, the relying party does not have to register with the social site. Furthermore, at any time the user can switch to a new social site without losing access to her account at the relying party.[26]

4.8.2 Session but No Account: Shopping Cart Obtains Data from Social Site

A shopping site that uses shopping cart software, or any relying party that create guest-user sessions, can rely on a social site for identity data just as it can rely on a personal data site. The shopping site creates a guest-user session by asking the browser to submit a fresh session credential, and later asks the browser to provide it with a delegated authorization credential, which the browser uses to obtain identity data from the social site. As discussed above, the social site is not informed of the identity of the relying party, let alone of the purchase made by the user.

5 Conclusion

We have proposed an architecture for the NSTIC ecosystem based on the assumption that privacy-enhancing technologies should be built into the fabric of the Web, so that they can be used with little or no effort by end-users and by developers of relying parties, identity providers and social sites. Specifically, we have proposed extensions to the two core protocols of the Web, TLS and HTTP, that will facilitate the use as well as the automated issuance of credentials, including privacy-enhanced credentials such as Idemix anonymous credentials and U-Prove tokens, PKI certificates, and bare-key credentials for session tracking. We have described a variety of use cases to illustrate how the proposed architecture will meet the goals of NSTIC.

The proposed extensions remain to be specified in detail, and will have to be standardized by the IETF; and they will not be trivial to implement for developers of TLS libraries, browsers, and Web middleware software. But these challenges are commensurate with the ambitious privacy and security goals of NSTIC. Once the extensions are standardized and implemented, it will be easy

[25] As in the use case of Section 4.7, the relying party may choose not to store the identity data, counting on the fact that it will be supplied by the social site each time the user logs in.

[26] After the switch, when the user logs in to the relying party, the browser submits the same login certificate to the relying party, while it uses a new delegatable credential from the new site to issue a delegated authorization credential to the relying party. The relying party uses the latter credential to obtain identity data from the new social site.

for relying parties, identity providers and social sites to participate in the identity ecosystem, and for users to enjoy its benefits.

References

[1] NSTIC National Program Office. Web site of the National Strategy for Trusted Identities in Cyberspace. At http://www.nist.gov/nstic/.

[2] Jeremy Grant. National Strategy for Trusted Identities in Cyberspace, April 6, 2011. Presentation at NIST IDtrust 2011, available at http://middleware.internet2.edu/idtrust/2011/slides/03-national-strategy-trusted-identities-cyberspace-nstic-grant.pptx.

[3] The White House. National Strategy for Trusted Identities in Cyberspace, April 2011.

[4] Howard A. Schmidt. The National Strategy for Trusted Identities in Cyberspace and Your Privacy, April 26, 2011. White House blog post, available at http://www.whitehouse.gov/blog/2011/04/26/national-strategy-trusted-identities-cyberspace-and-your-privacy.

[5] David P. Kormann and Aviel D. Rubin. Risks of the Passport Single Signon Protocol. *Computer Networks*, 33:51–58, 2000. Available at http://avirubin.com/passport.html.

[6] J. Hughes et al. Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0. OASIS Standard, March 2005. Available at http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf.

[7] Tom Scavo and Scott Cantor. Shibboleth Architecture Technical Overview, Working Draft 02, June 2005. Available at http://shibboleth.internet2.edu/docs/draft-mace-shibboleth-tech-overview-latest.pdf.

[8] OpenID Foundation. OpenID Authentication 2.0 Final, December 5, 2007. At http://openid.net/specs/openid-authentication-2_0.html.

[9] OAuth Working Group. Web Page of the OAuth Working Group of the IETF. At http://datatracker.ietf.org/wg/oauth/charter/.

[10] Ben Laurie. OpenID: Phishing Heaven, January 19, 2007. At http://www.links.org/?p=187.

[11] Facebook. Authentication. Retrieved on July 7, 2011 from http://developers.facebook.com/docs/authentication/.

[12] Jan Camenisch et al. Specification of the Identity Mixer Cryptographic Library, Version 2.3.1, December 7, 2010. Available at www.zurich.ibm.com/~pbi/identityMixer_gettingStarted/ProtocolSpecification_2-3-2.pdf.

[13] Christian Paquin. U-Prove Technology Overview V1.1 Draft Revision 1, February 2011. There is no http URL for this document, but it can be downloaded by following links from http://www.microsoft.com/u-prove.

[14] Christian Paquin. U-Prove Cryptographic Specification V1.1 Draft Revision 1, February 2011. There is no http URL for this document, but it can be downloaded by following links from http://www.microsoft.com/u-prove.

[15] Microsoft Identity and Access Team. Beyond Windows CardSpace, February 15, 2011. Available at http://blogs.msdn.com/b/card/archive/2011/02/15/beyond-windows-cardspace.aspx.

[16] Mike Jones. Personal reflections on the CardSpace journey, February 15, 2011. Available at http://self-issued.info/?cat=3.

[17] Ashish Jain. Open letter to the CardSpace team, October 2, 2007. Available at http://itickr.com/?p=82.

[18] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2, August 2008. RFC 5246. Available at http://tools.ietf.org/html/rfc5246.

[19] R. Fieldig et al. Hypertext Transfer Protocol – HTTP/1.1, June 1999. RFC 2616. Available at http://tools.ietf.org/html/rfc2616.

[20] M. Belenkiy et al. Randomizable proofs and delegatable anonymous credentials, September 16, 2009. Available at cseweb.ucsd.edu/~hovav/dist/delcred.pdf.

[21] D. Cooper et al. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, May 2008. RFC 5280. Available at http://tools.ietf.org/html/rfc5280.

[22] J. Franks et al. HTTP Authentication: Basic and Digest Access Authentication, June 1999. RFC 2617. Available at http://tools.ietf.org/html/rfc2617.

[23] J. Hoyt et al. OpenID Simple Registration Extension 1.0, June 2006. Available at http://openid.net/specs/openid-simple-registration-extension-1_0.html.

[24] Department of Defense. CAC: Common Access Card. At http://www.cac.mil/CardInfoIdentification.html.

[25] NIST. About Personal Identity Verification (PIV) of Federal Employees and Contractors. At http://csrc.nist.gov/groups/SNS/piv/index.html.

[26] Wikipedia. Secure Electronic Transaction. At http://en.wikipedia.org/wiki/Secure_Electronic_Transaction.

[27] Kaliya Hamlin. Personal Data Ecosystem Consortium. At http://personaldataecosystem.org/.

[28] Kaliya Hamlin and Mary Hodder. Empowering individuals with tools to manage their personal data for the identity in the browser, May 2011. Presented at the W3C workshop on Identity in the Browser. Available at http://www.w3.org/2011/identity-ws/papers/idbrowser2011_submission_54.pdf.