Improving Digital Credential Management in Browsers & Servers

David L. Wasley, Editor[1]

September 2002

It is widely understood that use of the web for trustworthy business applications requires use of strong digital credentials[2] supporting the identity[3] and privileges[4] of the individual transactor. Such credentials can be created and validated by means of a well designed and deployed Public Key Infrastructure (PKI). PKI can provide great flexibility, trustworthy interoperability, and a very convenient user experience. This is the strategy chosen by both higher education and the Federal government for new business applications, both internally and with external partners.

Most recent distributed applications use a standard browser as the user (client) tool of choice. Two predominant web browsers - Netscape Communicator and Microsoft Internet Explorer - will be used by most transactors. Each browser has some degree of support for PKI credentials but neither has sufficient flexibility and ease of use to be truly viable, especially for less technically adept individuals. Vendors of these browsers and developers of other newer web clients must incorporate far better support for sophisticated and user-friendly use of these digital credentials.

Furthermore, management of and access to PKI credentials should be independent of the application making use of them, e.g. the browser. The browser should not have to be concerned about where the certificates are stored, path discovery and certificate validation, etc. These common services must be available to applications in all platform operating environments.

To that end, the higher education community through its professional IT activities, offers suggestions below for improving existing browser and server functionality. These are intended to be starting points for discussion, not specific requirements. They are intended to lead towards desktop clients and more sophisticated servers that can support "user friendly" and secure, robust electronic commerce environments.

Assumptions

A number of practical considerations underlie the recommendations to follow. These include:

1) *Most users of the browsers will be non-technical or at least technology-challenged.* The typical user must not be confronted with technical jargon or complex choices to make with respect to security issues. The user security interface in browsers today is a serious impediment to wide spread acceptance and use of PKI credentials.

2) *The PKI credential will contain very little of the holder's actual identity attributes.* The primary personal digital credential is intended to be a long lived binding of a digital token to a known individual. If it were to contain very much about an individual, it might become invalid relatively quickly. Furthermore, the attributes of "identity" that are useful will vary from application to application. "Common name" for example indicates nothing of a person's roles or authority. However, roles and authority change often enough that they should not be included in a long lived credential. Therefore, the vast majority of identity attributes will be held in associated secure databases and/or directories.

3) *Most individuals will require or desire a number of different digital credentials.* Just as we have different forms of identity and authorization in traditional commerce (authorization lists, drivers' licenses, credit cards, passports, checks, employee ID, ...) we will have multiple digital credentials. For example, one might have a personal credential, one or more functional or role based credentials, short lived temporary credentials, group membership credentials, and so on.

   Specific industries might well have need for unique credentials. For example Banking might have different requirements than Health Care. Libraries will want credentials that protect intellectual freedom by not mapping (directly) to a known individual. Other on-line resource providers may want credentials that assert eligibility of the holder under terms of a specific contract. For various reasons, a community of interest may insist on issuing credentials to its constituents instead of relying on externally issued credentials.

   Equally important, individuals may not be willing to tolerate the potential invasion of privacy that a single credential would constitute if it were to be used for identification in all network-based activities in which the holder engaged. Even though some individuals may prefer fewer credentials and be willing to aggregate functionality at the cost of privacy, the need for the more general scenario must be recognized.

4) *Access to resources will be based on algorithmic business rules.* In a complex environment, a user may need to gain access to a wide variety of resources, each with its own access management rules. These should be automated so that the user need never be bothered with excessive login prompts, choices, verifications, etc. The PKI credential environment will include secure directories that hold the individual attributes necessary to support the algorithmic automation of access rules. The digital credential must have strong binding to the appropriate record(s) in such directories.

5) *Browsers will not be the only client application using PKI certificates.* Secure email, for example, or SSH applications or even multiple browsers should be able to use the same set of PKI credentials. These credentials represent a collection of important authentication and other information that is not specific to an application but to the individual to whom they apply. PKI credentials should be cached at the system level and be available to all applications through a standard API.

6) *Users will need to invoke the same digital credential on more than one computer, and some of these will be multi-user or public computers.* Students using public computer labs or special purpose systems will need to be able to use their PKI digital credentials without the risk of leaving traces of them on the platform after they depart. Individuals who use multiple computers routinely, for example telecommuters, should be able to make use of the same set of credentials with all their platforms.

7) *Trust structures will evolve and become far more complex.* Simple two level, single root trust hierarchies simply do not reflect how trust works in the "real world". Instead there will be many smaller, multilevel trust hierarchies with bilateral agreements both with bridge CAs and directly with other hierarchies. Policy reconciliation at the bridges is a difficult problem but must be accommodated, ultimately.

8) *Networks will continue to become more robust, ubiquitous, and of higher capacity.* Today 100 Mb/s to the desktop is taken for granted and inter-institutional data capacities range from 45 to 1,000 Mb/s and continue to grow. Critical networks are made robust through redundant connectivity and engineering to avoid single points of failure. Therefore, distributed systems, using resources at several different network nodes, can be relied upon for mission critical applications.

Finally, we assume that not all conventions, protocols, and hardware exist today to achieve the level of convenience and flexibility that we believe is required. We are anxious to participate in development of these added standards. However they are developed, it is essential that they be freely available to the industry and broadly implemented across all platforms.

Recommendations

The following set of recommendations is neither exhaustive nor diagnostic. Additional functionality may be required once more experience has been gained with this technology. Additional misfeatures may be found as well.

The challenge for browser developers is to provide ways for the technology-challenged user to manage a fairly complex set of security and authentication features without undermining their control or the flexibility of these features.

These recommendations are not in any particular order.

1. **Thoroughly document the browser's handling of PKI credentials**

   Documentation, if any, is sketchy and often difficult for non-technical users to understand. The common answer to "How does the browser handle e.g. multiple personal credentials?" is "Try it and figure it out." This should not be necessary, Furthermore, the security and trust models and related assumptions must be clearly documented. There should be both easy to understand, basic descriptions and detailed documentation for technical support staff and adept users.

2. **Clarify dialog boxes to avoid confusion**

   Some browsers or browser dialogues are sufficiently ambiguous that novice users can become confused. Passwords have been known to be entered in "userID" boxes, for example, because at a quick glance the user was not certain what was being asked for. Dialogue boxes that ask the user to make decisions about complex security issues will simply discourage use of security technologies. While some of this confusion is produced by poor server code, a lot of it is in the browser itself.

3. **Obtaining and securing a PKI certificate**

   Requesting and downloading a certificate is fairly straight forward but some browsers allow the resulting certificate to be stored unprotected. It **must** be possible for the certificate issuer to require a user specified password or pass phrase to securely store the certificate's private key. If the issuer so desires, this **must not** be at the user's option.

   The password should be associated with a "group" of certificates so that different users of the same platform can protect their caches individually. The minimum "group" would be 1 certificate but, since a given user might have more than one, it should be possible to have a common password protect and unlock them all.

   If the keypair associated with the issuance request was generated on a smartchip device, the certificate cache manager (browser or system services) must verify that the resulting certificate is being stored onto the appropriate smartchip device by verifying that it holds the companion

private key. This action might take place long (e.g. days) after the request was issued. Again, it must be possible to require a passphrase or PIN to protect the private key on the smartchip device.

4. SSL negotiation and user privacy

During the SSL handshake, the server may be configured to send a certificate request to the client. This is only a request; the SSL connection might still be established if the client has no certificates. However, if the client does have one or more certificates, one may be sent automatically or the user is asked to select one. Often the user is not given the option of declining to send any. In order to allow the user to defend her or his privacy, there should be a "none of the above" option which tells the browser to send no certificate even if one is requested. In the same vein, there should be a way for a user to specify specifically which of their certificates may be used for SSL purposes (see also paragraphs 5 and 10 below).

5. Requesting a user's certificate from another platform

Currently, the only way for a server to request a certificate over HTTP is to open an SSL connection to the client. This is far too rigid and limiting. The certificate to be used for SSL might be entirely different than the certificate required for access to the server's services. There must be a standard protocol for requesting a PKI certificate at the application level, including support for automatic selection of the appropriate certificate by the client side.

6. Accepting Server certificates

When a browser doesn't already "trust" a server's CA, it asks the user to approve use of the server's certificate. (Some browsers simply don't allow unrecognized certificates to be accepted at all.) This dialogue could be avoided if the browser was able to discover a trust path or chain between the CA that issued the certificate and at least one of the CAs in the user's personal certificates. The credential is merely an assertion of authenticity vouched for by a trusted authority. Acceptance of a server's credential does not imply trust of the server itself but merely of it's identity. The user has already indicated her or his desire to gain access to the server by selecting its URL.

If no trusted chain can be found, the user should see a simple explanation of the choice she or he must make and be required to see no more than one or two screens in order to accept or deny the server's certificate.

In the meantime, it must be made easier for common root CA authority certificates to be configured into browsers. For example, the Corporation for Research and Education Networking (CREN) is supporting a root CA for the higher education community in the US. It should be reasonably straight forward to add the CREN root CA to the set of trusted CAs that comes with the standard browser distributions.

In addition, campuses or corporations might have CAs that should be trusted by their communities. It would be inappropriate to place those root CA authority certificates in standard browser distributions. Instead, organizations that wish to do so must be able to add a server's root CA's certificate to the browser's cache, e.g. through a special "import" function or as an external configuration file. Alternatively, support for a trusted "path discovery, validation, and certificate verification" server should be developed.

7. Trust path discovery, including bridge CA's

Many trust hierarchies will be deeper than 2 levels and there will likely be branches through multiple bridge CAs and ad hoc bilateral agreements. Both servers and clients will need to verify PKI certificates using the X.509 open standards policy processing rules through moderately long trust chains, which may have bridge CAs or other policy-based restrictions along the way. Local CAs might mediate trust chain discovery for their clients, or special servers which focus on trust chain creation and validation might be developed (see above), but if neither is done, ultimately the client browser or web server may have to discover the chain independently.

8. Dual certificates: signing vs. encryption

Both digital signatures and data encryption are essential for a robust business environment. By the same token, the decryption (private) key that must be used to recover an encrypted document often must be escrowed in order to ensure business continuity, while the digital signature (private) key of a transactor must never leave control of the individual holder or else any digitally signed transactions making use of that key become repudiable. Therefore, browsers, email clients, and signing applications must support dual certificates and respect the appropriate use of signing versus encryption key pairs. Again, this requirement is in conformance with the X.509 standard processing rules.

9. Signing and encryption

Browsers must make the user interface for signing and/or encrypting information simple and intuitive. For example, browsers used for financial applications or e-commerce should have "buttons" for signing and/or encrypting the information input by the user or the complete screen as seen by the user.[5] Encryption at the transport layer (SSL) may not be adequate to secure sensitive information once it is received by the server. Auditability may require verification of the identity of the specific transactor either at the time the data is received or later in the archive, or both.

It might be a requirement of some highly sensitive applications that the user be required to re-verify their ownership of a private key at the time a document is signed. Clearly this would require that a directive from the server cause the browser (or other signing application) to ask the user to "unlock" their credential cache again.

10. Managing multiple PKI credentials (private keys and certificates)

One of the most difficult problems to be solved is making the use of multiple PKI credentials held by the user as transparent as possible to that user. It is absolutely inevitable that users will have more than one PKI credential (possibly a great many more) and it is critical to minimize the user overhead in deciding which to use in a given situation.

A new set of standards is needed whereby a server can ask for a certificate of a certain type or with certain attributes, but only for certificates which are allowable by some overall aegis. For example, an on-line publisher's server should be able to ask for a certificate that asserts the holder's eligibility to gain access to their holdings but should not be able to ask for a certificate that asserts eligibility to gain access to a different publisher's content.

This could be accomplished with a combination of globally defined certificate extensions and/or external information in the user's certificate caches so that the appropriate credential can be chosen automatically by the browser. In the near term, users might be asked to define preferences at the time a certificate is first used, e.g. "Do you want to use this certificate once, always for this site, or never for this site?", and that preference recorded. The ultimate goal is to minimize the user's involvement in every access control dialogue while retaining appropriate levels of privacy, security, and trust decisions.

11. Certificates on demand (e.g. anonymous)

Many servers and applications need only associate a credential with a class of individuals eligible to make use of the service. For example, access to on-line library materials may be granted to any member of a campus community and the specific identity of the individual may be inappropriate to reveal during the process of granting access. Therefore, users may have (or be able to get) "anonymous" or other special purpose certificates "on demand" from their primary CA (the CA that issued their personal identity credential).

This requirement is strongly linked to the issue of managing multiple certificates. In a scenario where a server asks for a certificate type that the user does not have, the browser should automatically ask the user's primary CA or perhaps a special purpose authorization server for such a certificate and add it to the user's cache.

For example, a McGraw-Hill server might ask for a certificate asserting eligibility under the terms of the holder's campus contract with McGraw-Hill. If the user's cache does not contain such a certificate, the browser should ask the campus' CA or perhaps a campus library authorization server for one on behalf of the user, using the user's primary identity credential to validate the request. The CA or server would check the user's attributes and either issue the appropriate eligibility credential or deny the request.

The proxy authorization certificate being developed in the Grid computing community is another example of this. This concept might become valuable in a more general, commodity context.

Electronic voting is a special case of "certificates on demand." What the user would see is a web page with a description of the election issues, etc., and a button to select when the user is ready to vote. Then, behind the scene, the user's browser asks the ballot server for a copy of the ballot and a certificate asserting eligibility to "vote". The user is presented with the ballot and radio buttons to select their preferences. When complete, the browser uses the voting certificate to submit the completed and encrypted ballot to the "ballot box server". The ballot box only allows one instance of that certificate to be used and has no way of (directly) associating the user with the ballot. What the user sees is simple and intuitive; what the browser does is fairly sophisticated.

12. Using private keys and certificates for multiple applications

The browser will not be the only application using private keys and PKI certificates. For example, secure email clients and secure "telnet" require the use of public/private key pairs and/or certificates. It can not be assumed that a single client package will or should provide the complete user interface to managed services.

Therefore, PKI credential caches (which may include both private keys and PKI certificates) should be managed at the operating system level with a standard API for all cache functions. In preparation for this, browsers should be designed to be able to easily take advantage of such system services when they exist.

For the near term, import and export of PKI credentials between the browser and a local encrypted cache should be simple and reliable. Unfortunately not all browsers even support this functionality.

13. Portability/Roaming: Smartcards, USB dongles, PKCS11+

People use more than one computer and in some cases use shared computers. The PKI credentials that they need are associated with the individual, not with the particular computer they happen to be using. Therefore, there must be a convenient and secure way for people to make their PKI credentials available on any platform they use.

Smartchip technology (smartcards, DataKeys, etc.) seems appropriate for this but there are several impediments to be overcome. First is the hardware interface; second is the capacity and functionality of the smartchip; third is compatibility, and fourth is cost. It is reasonable to assume that these can be overcome in the not-too-distant future.

Related to the need for standard operating system support for PKI credentials, the cache managed by the operating system should be that which is stored on the user's portable smartchip. All common platforms should be delivered preconfigured to be able to do this.

The smartchip devices should be recognized automatically when they are inserted or connected. Some browsers and/or operating systems require a "registration step" before a new smartchip device certificate can be recognized. Some go so far as to build a table of all registered smartchip certificates which the user then must scroll through to find their own. These 2 characteristics present a very awkward and unfriendly user

interface! It should be possible to plug in a smartchip device and have it available to applications without any further action by the user.[6]

Until smartchip support is widely implemented in operating systems, browsers should do the same: look for and utilize a smartchip-like object if it is connected to the user's platform. In no case may a browser ever copy a user's private key from the smartchip into a local cache! This has the benefit of minimizing the risk that a usable certificate will be left on a public platform accidentally.

Until smartchip technology is widely available, browsers may have to support secure PKI credential servers. Such servers would hold the user's private key, doubly encrypted, and the associated PKI certificate. When needed, the certificate and private key might be transferred to the user's platform but must be done in a way that self destructs when the user leaves the platform. It also might be possible to create a "virtual smartchip" server that would provide all of the same functionality but would be a secure server instead. Only identity credentials, possibly to be used also for encryption, may be served in this way; digital signing credentials may not be served because to do so would compromise the veracity of the signing (private) key. This approach allows roaming without smartchip devices, but is obviously not as secure as smartchip devices since the crypto functions are done in memory or another server rather than on the chip.

14. Servers must recognize Attribute Certificates

Finally, servers must be smarter as well in their use of PKI digital credentials. Digital credentials will not include all possible identity attributes. Instead, they will refer to directories or databases where such attributes can be found. There should be standards for commonly needed attribute object classes and configurable rules based processes for using these attributes in access management decisions. Careful adherence to such standards will help minimize the unnecessary proliferation of the number of certificates a user must hold.

Clearly there is overlap among many of the issues raised above. Clearly there may be multiple approaches to addressing them, e.g. roaming servers versus smartchip devices. The intent of the above is to highlight the sort of functionality that we believe is required and encourage browser and server developers to take seriously their current shortcomings.

Getting it Done

We realize that the requirements above are non-trivial to implement (although some may already be in the pipeline). The higher education community is both ready to help with this development and a natural proving ground for the results. We look forward to working closely with industry to achieve this important next step towards a robust and easy to use e-business environment.

---

[1] Contributors to this document included
Judith Boettcher, CREN
Michael Gettes, Georgetown University
Clair Goldsmith, University of Alabama and Chair, EDUCAUSE Net@EDU PKI Work Group
Richard Guida, Department of the Treasury
Ken Klingenstein, University of Colorado and Chair, Internet2 Middleware Work Group
Clifford Lynch, CNI
Eric Norman, University of Wisconsin
David Wasley, University of California, Office of the President

[2] "Digital credential" refers to the combination of PKI certificate and private key. The credential is something that can be validated independently by a receiver. The content of a credential certificate may include a digital token that is unique to the holder of the credential. It is this token that allows retrieval of additional attributes relevant to the holder. This form of credential is therefore a personal or identity credential. Other forms of digital credentials may assert specific eligibility for services or membership in a class of non-specific individuals.

[3] "Identity" is the set of attributes pertaining to an individual entity. These include such things as common name, addresses, roles, status within a context, authority, etc.

[4] "Authorization" is a process followed by a service provider to determine whether a requestor is to be allowed to complete a transaction or gain access to a resource. Authorization may be based on some set of the requestor's attributes, location of the access node, time of day, availability of resource capacity, or any combination of these and other things.

[5] It must always be possible for the user to see exactly what has been signed, and in no case should the signed object contain a dynamically followed link, URL, etc.

[6] Consider a "public kiosk" application where access to restricted information or services is based on a user's smartchip device. Any user should be able to walk up, plug in, and get service.