

Credential Wallets:

A Classification of Credential Repositories Highlighting MyProxy

Jim Basney William Yurcik Rafael Bonilla Adam Slagell

National Center for Supercomputing Applications (NCSA)

University of Illinois at Urbana-Champaign

{jbasney,byurcik,bonilla,slagell}@ncsa.uiuc.edu

Abstract

In recent years, individuals have experienced an increased requirement for digitally identifying themselves, in both their work and personal lives. People typically have multiple credentials for digitally identifying themselves to different online software or services in different roles (for example, different credentials for work and home). We are looking for technology that allows users to manage their credentials conveniently and securely. This paper presents a classification of existing credential management software, highlighting the MyProxy online credential repository, an open source system for managing credentials in the Grid Security Infrastructure.

1. Introduction

There is a plethora of interfaces to network applications and many of these interfaces require a unique user name and password or similar technique for authentication. As a result individuals typically possess many digital identities, which are often short-lived. For example, digital identities are created for new employees, modified due to promotions and changes in responsibilities, and revoked when those employees leave the organization. Identity management is the ability to control the full life cycle of digital identities from creation to termination.

Secure identity management is a serious and important challenge. Identity theft topped the list of complaints to the U.S. Federal Trade Commission in 2002 for the third consecutive year, accounting for 43% of all complaints with a large gap between the next highest complaint (Internet auction fraud is second at 13%). An Associated Press article in January estimated that up to 700,000 people in the U.S. will be victims of identity theft in 2003, a figure that doubled from 2001 to 2002, with an average of \$1,000 in expenses per victim to cope with the damage to

their accounts and reputation (do the math for the astounding total cost).¹ The 2002 joint CSI/FBI study states that 38% of respondents suffered some type of identity misuse on their website (with fraudulent or stolen user names and passwords).² The empirical data clearly shows that identity theft is a real problem that is growing significantly in both frequency and impact.

There are strong forces converging to find a solution. Ecommerce is dependent upon accurate, authenticated user preference data to support price discrimination. The legal community requires a secure method of authenticating user identities that will hold up in court. Technical operations staff is eager to prevent intrusions that can be traced to breeches of user authentication. Lastly, end users are eager for a solution that will improve usability of identity management mechanisms that otherwise are being avoided and circumvented.

To this point, most work has focused on identity management solutions in which a single user identity can be used across multiple websites and electronic resources. In this paper we also highlight a different solution gaining momentum in which multiple different credentials can be stored in a repository that we call a "credential wallet". A credential wallet service can provide a consolidated view of a user's credentials, allowing users to easily manage their own credentials.

The unique contributions of this paper are two-fold: (1) to organize single sign-on solutions and credential repositories into a classification structure for comparison and (2) to present our novel implementation of the open source MyProxy credential repository for identity management between users and networked resources within the Grid Security Infrastructure.

The remainder of this paper is organized as follows: Section 2 presents the Grid Security Infrastructure PKI, which has provoked significant research activity on authentication for distributed systems. Section 3 presents two alternatives to traditional PKI key management being explored in the Grid security community: online certificate authorities and credential repositories. Section 4 presents the open source MyProxy online credential repository. Section 5 reviews other currently available single sign-on solutions and credential repositories. Section 6 concludes the paper.

2. Grid Security Infrastructure

The Grid is a vision of a ubiquitous middleware infrastructure supporting collaborative, distributed computing across organizational boundaries that has prompted large deployment projects around the world using the Globus Toolkit™ [Foster99]. Multi-organizational Grids typically have multiple supercomputers distributed across a WAN, each with its own scheduler. While users can run jobs remotely across the Grid on these networked supercomputers (to create a virtual supercomputer), the security environment and resources of each supercomputer are

¹ estimates from Associated Press article ("*Identity Theft Cases Have Doubled*," The Wall Street Journal, Jan. 23, 2003, p. D2); check our math: 700,000 victims times \$1000/victim = \$700M; includes only individual expenses and not corporate costs that have been estimated at \$3.4B for 2002 (A. Borrus, "*To Catch an Identity Theft*," Business Week, Mar. 31, 2003, p. 91); Ty Sagalow CIO/AIG eBusiness Risk Solutions also reports each victim spends an average of 14 months/175 hours clearing a stolen identity (*IRMI Update*, Mar. 18, 2003).

² see <<http://www.gocsi.com/press/20020407.html>>; while this survey is often criticized as being inappropriate for making generalizations due to procedural flaws such as self-selection, to our knowledge a more comprehensive or statistically-valid computer crime survey does not exist.

typically locally controlled. To manage these distributed issues, the Grid security architecture relies heavily on strong authentication.

Almost all of these Grid-building efforts have adopted the Grid Security Infrastructure (GSI) [Foster98, Butler], a Public Key Infrastructure (PKI) based on X.509 certificates [X.509], the SSL/TSL protocol [Dierks], and the GSSAPI [Linn] standard. For many organizations, adopting GSI included the creation of a Certificate Authority (CA) to provide the organization's Grid users and administrators with authentication credentials.

Users can run the Globus Toolkit™ *grid-cert-request* program to generate an RSA key pair and an X.509 certificate request to be submitted to a CA for signing. The key pair and signed certificate are stored in files in a subdirectory of the user's home directory, with the private key encrypted with a user-chosen passphrase.

2.1. A Traditional PKI

In a typical Grid PKI, users must generate and manage their own keys, and they must authenticate to a Registration Authority (RA), typically with a photo ID, to obtain a signed certificate. Grid CAs are typically offline, meaning that a human operator must verify and approve each certificate request and insert a hardware device containing the CA's private key into the CA computer to sign certificates.

The operation of a traditional Grid CA is relatively expensive. Certificate requests must be manually approved and certificates must be manually revoked by issuing new Certificate Revocation Lists (CRLs) when compromise is suspected or keys are lost. Multiple CA operators are sometimes required, each with distinct roles, to provide reasonable resilience to insider attack.

Users in a traditional Grid PKI find credential management, including the responsibility of initially obtaining credentials (i.e., PKI enrollment), renewing credentials, and keeping their credentials secure, to be cumbersome. Enrollment and renewal are error-prone, and users must manually copy their credentials to the systems from which they perform PKI authentication operations. There are also serious security concerns with end-user key management: people do not always choose good passwords or follow good practices when securing their private key files.

3. Alternatives to Traditional PKI Key Management

While user identity management concerns can be partially addressed by better software and documentation, a traditional PKI can exert only limited technical control over client-side operations. The CA has no knowledge of how the user generated his or her private key, cannot technically enforce any rules for choosing good passwords, and cannot technically control how the user protects his or her private key. The best the PKI can do is to specify policies that users are required to follow, and hope that users are technically able to do so.

We believe that centralizing key distribution and key management in a PKI has the potential to improve usability, manageability, and security. A centralized key distribution center is a well-accepted practice in the Kerberos authentication system and we believe this same approach can be applied effectively to a PKI. For an outline comparison between PKI and Kerberos centralized key distribution systems see Table 1. The two primary key management alternatives

that appear to have the most benefits are: (1) an online CA and (2) an online credential repository. We discuss both of these two key management alternatives in more depth in the following sections and an outline comparison in Table 2.

3.1. Online Certificate Authority

An online CA can simplify PKI enrollment and key management by bootstrapping from existing online authentication mechanisms. Users authenticate to the online CA and issue a certificate request. If a user authenticated identity matches the identity in the certificate request, the CA signs and returns the certificate to the requester. The identity match may be determined by a simple transformation, i.e., by transferring identity information directly from the authentication mechanism to the certificate, or the online CA can maintain an identity-mapping database. The online CA must have a CP like any other CA. The CP will state what authentication mechanisms the CA will accept and how identities are mapped from the source authentication mechanism to the PKI.

A significant benefit of the online CA approach is manageability. The online CA must be professionally managed so it is reliable and secure but otherwise does not require traditional CA operations such as manually approving requests and physically managing an offline CA key.

The CA may issue long-term (years) or short-term (hours) credentials. If the CA issues long-term credentials, then as with a traditional CA, users must continue to manage their credentials, i.e., storing them securely, transferring them between machines, etc. However, if the CA issues short-term credentials, users can retrieve credentials on demand. Additionally, issuing short-term credentials provides for straightforward revocation by removing a user's authorization to retrieve credentials in the future.

A compromise of an online CA would allow the attacker to sign certificates for any identity in the PKI. If the CA key is directly accessible (for example, stored unencrypted in a file on disk), the attacker could steal the key and sign bad certificates from another location at a later time. Alternatively, if the CA key were hardware protected, the attacker would need to fool the hardware device into signing bad certificates during the attack. The attacker could also manipulate the authentication database or identity-mapping configuration on the CA to obtain invalid credentials. In either case, the compromised CA would need to be abandoned, and a new CA would need to be created in its place, requiring notification of all relying parties that the old CA certificate should no longer be trusted and a new CA is now in place.

The National Partnership for Advanced Computational Infrastructure (NPACI) has developed an online CA called CACL that issues long-term credentials. NPACI Grid users run a single command from any NPACI production computer, enter their password to authenticate, and immediately obtain credentials.

KCA/Kx509 [Kornievskiaia] provides an online CA tied to a Kerberos realm that issues X.509 credentials with a lifetime equal to the lifetime of the Kerberos ticket used for authentication. Identity mapping from Kerberos to X.509 can be implemented via an LDAP database or by simply using the Kerberos ID (username@REALM) with which the requester authenticated as the CommonName. Development is underway to add KCA support for LDAP-based password authentication in addition to Kerberos authentication. KCA/Kx509 has been deployed at Fermi National Accelerator Laboratory and the University of Southern California.

3.2. Online Credential Repository

An online credential repository can provide an alternative to user-managed long-term credentials by providing a well-secured credential storage service that allows credentials to be retrieved easily over the network with appropriate authentication.

One of the benefits of a credential repository is that it need not be tied to a CA, although that is one option. Instead, a credential repository could be deployed to manage the credentials of a single user, a small group within a single organization, or a large collaborative group that spans organizations. The repository could hold credentials from one CA or could be used to manage credentials from multiple CAs, allowing users with multiple credentials issued from different CAs to access all the credentials via a single interface.

A compromise of a credential repository would give the attacker access to all credentials in the repository. Encrypting the credentials in the repository requires the attacker to perform an additional offline attack on each credential. Sufficiently strong encryption may make the offline attack prohibitively expensive. However, if the credentials are compromised, either all the credentials must be revoked, resulting in a very large CRL, or the underlying CA must be abandoned.

Credential repositories can be categorized as either mechanism-aware or mechanism-neutral. A mechanism-aware repository can support mechanism-specific protocols for credential retrieval, which allows the repository to implement policies on the credentials that clients can retrieve. For example, a repository can hold long-term user keys that never leave the repository but are instead used to sign short-term credentials, so credential revocation can be implemented by simply removing the keys from the repository. However, allowing the repository server direct access to the keys weakens non-repudiation claims. In contrast, a mechanism-neutral credential repository can store many types of credentials, encrypted/decrypted by the client, so the repository itself never has access to the unencrypted credentials.

In [Sandhu], Sandhu et al. draw a related distinction between virtual smartcards and virtual soft tokens. In their terminology, virtual smartcards are systems where users retrieve private keys from a credential server and then use the keys directly without further interaction with the server, whereas virtual smart tokens are systems where the users don't have direct access to their private keys but instead must interact with the server to perform signing operations.

The IETF Securely Available Credentials (SACRED) working group is developing a protocol for mechanism-neutral credential repositories. The working group's requirements state that the credential must be opaque to the protocol and must not force credentials to be present in cleartext at the server [Arsenault], which precludes a mechanism-aware solution. In contrast, the MyProxy online credential repository, described in more detail below, provides a mechanism-aware service for storing GSI credentials [Novotny]. Many commercial PKI credential repositories are also available, reviewed below and in [Sarbari].

To provide further transparency, a user's credentials could be generated and loaded into a repository by an organization's security personnel on the user's behalf. PKI credential creation could be a natural addition to standard account creation for a new user. For example, when a new user account is created, the account management group sets an initial site-wide password for the user and sends it to the user in a letter sent via postal mail. The user's pre-generated PKI credentials could be loaded into the credential repository under his or her username and initial

password. When the user receives the letter, he or she can immediately login and retrieve his or her PKI credentials from the repository.

Like the online CA, the credential repository must be professionally managed for reliability and security. However, a CA may still be needed for the organization, so both a CA and repository may need to be managed. If the repository is tied to the CA and credentials are loaded on behalf of users by the site's security staff, then CA operations may be simplified. Rather than interacting directly with PKI users, authorizing certificate requests and signing certificates on demand, the CA could sign new certificates as part of other account creation processes, which could be performed in batches.

3.3. The Credential Wallet

Many current Grid deployment efforts are dedicating significant resources to defining and publishing CA policies and vetting and cross-certifying CAs. This effort is primarily motivated by the desire to allow Grid users to access the resources of different organizations using a single credential. As many in the Kerberos community discovered when attempting cross-realm authentication, establishing trust in the authentication systems of external organizations is a tedious and expensive process.

As the need for online authentication continues to grow, with many more types of devices and services coming online, we do not believe that a single credential per user will be sufficient (or even desirable from a privacy perspective³). Every application or service has the potential to require a new set of identity data. Just consider ubiquitous access (Internet, Grid, intranet, extranet) to a limitless variety of resources, from different user devices (desktops, kiosks, PDAs, and cell phones), each with evolving interfaces, different credentials for varied mechanisms (X.509, Kerberos, .NET), for different trust policies, issued by different authorities (CAs, KDCs), for different purposes (authentication, signing, encryption). Users may have credentials for different roles such as project-based accounting or managing different security levels. Authorization credentials will be issued to provide scalable authorization services. Additionally, multiple trusted credentials (CA certificates, public keys) will be required for establishing the roots of trust. Given this reality, we see a practical need to develop good mechanisms for managing multiple credentials.

A "credential wallet" can provide identity management based on the physical analogy to "out-of-wallet" information. Specifically we envision a credential wallet service that provides a consolidated view of a user's credentials, allowing users to easily access and manage their credentials. A single sign-on unlocks the wallet, allowing the user to transparently access the credentials he or she needs during an online session. Protocols that support automatic negotiation of required credentials will allow applications to retrieve the needed credentials from the wallet or notify the user that a needed credential is unavailable.

The credential wallet need not hold all the user's credentials; it could include pointers to online credential services (like a SAML server or X.509 Attribute Authority) from which the user can

³ Despite the convenience of a single user identity across websites and electronic resources, this also brings a concentrated privacy risk from the enhanced ability to aggregate audit trails from all online navigations. The collected preferences and purchasing history tied to a single identity could result in electronic profiling and unauthorized surveillance that is much more difficult to accomplish with multiple secure identities.

retrieve credentials on demand. Maintaining the pointers to these services in the wallet makes finding the services much easier. An application can query the wallet for a specific credential, and the wallet will know where to go to retrieve it.

Rather than hoping for agreement on a single authentication token for all online services, the credential wallet approach addresses the reality (and desirability) of heterogeneity. Many authentication and authorization services can co-exist online without resulting in poor usability. Empowering the user's own devices to manage multiple credentials of different types and negotiate with services for the credentials required gives the user the flexibility to conveniently use services that use different authentication methods without relying on a third-party service somewhere on the network to hold all the user's keys.

4. MyProxy Online Credential Repository

MyProxy [Novotny] is a credential repository for the Grid. Storing Grid credentials in a MyProxy repository allows users to retrieve a proxy credential whenever and wherever they need one, without worrying about managing private key and certificate files.

Proxy credentials [Butler] are short-term credentials generated from long-term credentials to facilitate single sign-on. Rather than exposing long-term credentials for the duration of a user's session, the user generates proxy credentials from long-term credentials at the start of the session and uses the proxy credentials for that session. The short lifetime of the proxy credentials limits their exposure.

Rather than storing long-term credentials on his or her workstation, a user can store his or her long-term credentials in a MyProxy repository that is securely managed by professional staff of the user's home organization. Users can then securely retrieve proxy credentials from MyProxy from any computer where MyProxy client software is installed without manually transferring credential files between computers.

MyProxy provides a set of flexible authorization mechanisms for controlling access to the repository. Server-wide policies allow the MyProxy administrator to control how the repository may be used. Per-credential policies allow users to specify how each credential may be accessed. Pass-phrase and/or certificate-based authentication is required to retrieve credentials from MyProxy. If a credential is stored with a pass-phrase, the private key is encrypted with the pass-phrase in the MyProxy repository for increased security.

In addition to providing direct access to a user's credentials, MyProxy is used to indirectly delegate credentials to services when native protocols do not support credential delegation. For example, Grid portals (web servers that provide access to Grid services via standard web browsers) require credentials to act on the user's behalf but the user cannot delegate credentials directly from the browser to the portal because HTTP does not support it. Instead, the user can establish a MyProxy account for the portal and send a username and pass-phrase to the portal (via HTTPS) under which it can retrieve credentials for the user from the MyProxy repository.

MyProxy supports credential renewal by allowing authorized services to renew a user's delegated credentials when needed before they expire. For example, a user can submit a long-running task to a job broker like Condor-G [Frey] and authorize the broker to use MyProxy to renew credentials. If the job queues or runs longer than the lifetime of the credentials initially delegated to the broker, then the broker can authenticate to MyProxy and retrieve new

credentials with an extended lifetime for the job, so the job can continue accessing Grid resources during its run. MyProxy's credential renewal service provides a trusted point of control and audit for the user's credentials, in contrast to delegating long-lived credentials directly across the Grid.

5. Comparison of Credential Repositories

We now compare several currently available Single-Sign-On (SSO) and credential repository systems. The SSO systems include Microsoft .NET Passport, MIT Kerberos V, and the Liberty Alliance Identity Federation Framework. The set of credential repositories we examine include Entrust TruePass, RSA Web Passport, VeriSign Roaming Service and Baltimore UniCERT. The results are summarized in Table 3.

5.1. Explanation of Table Fields

We use six different traits to compare the different SSOs and credential repositories. In this section we thoroughly describe the meanings of each trait.

5.1.1. Number of IDs

This field can assume two values: Single or Multiple. Single refers to systems that store or use only one ID. Microsoft's .Net Passport is an example of such a system. Here a user gets a unique Passport User ID (PUID), which is a 64 bit pseudo-random number. A user will be represented by this ID to all the .NET Passport enabled application servers. Of course, a user can always get more PUIDs, but this requires the user to reregister with passport.com, and these two user IDs are not linked in any way. This is why .NET Passport is considered a single ID system.

Multiple ID systems present the user with a single mechanism that allows her to access credentials for multiple IDs. Entrust's TruePass is a credential repository that works in this way. A user authenticates to the credential repository and is presented with a choice of IDs and corresponding credentials to retrieve. It does not require different or separate authentications to retrieve different credentials.

5.1.2. Credential Type

The "Credential Type" field refers not to how one authenticates with the credential repository or SSO but rather what type of credentials the repository returns to the user. These come in two basic varieties: tokens and keys. Tokens are short-lived statements that a user has authenticated to a particular server and can include more information such as the type of authentication used and the lifetime of the token. A user will typically get a token by authenticating to a mutually trusted server. Then he or she can present this token to application servers as a proof of identity since the application servers trust this server to authenticate users for them.

Keys are typically public/private key pairs. A user can go to a credential repository and retrieve a certificate and the corresponding private key. In this manner the user can submit the certificate to an application server along with proof that the user holds the private key. This could be through a challenge and response protocol or simply by signing the request with the private key.

5.1.3. Platform

By “Platform”, we are referring to client’s platform and NOT that of the credential repository or application servers. Many of the SSO solutions just require a web browser and hence are platform independent. For these we report “Any” in the field. Others download applets into the browser that are not platform independent. This could be because they are zero-footprint applications or because of their interactions with the Windows Digital ID store. Other clients are tricky to classify. While Kerberos is not bound to a particular OS, it requires “Kerberized” applications. The client applications must understand the Kerberos protocol. For example, any old e-mail client will not work. It may have to use KPOP instead of POP to retrieve e-mail. While this doesn’t restrict the OS used, it does inhibit portability from the client’s perspective.

5.1.4. Key Generator

In systems that use keys instead of tokens as credentials, it is of interest who generates the keys. The question is of course not applicable to token based systems. There are four possible answers to who generates the keys: repository, client, both or either. The first two answers are self-explanatory. By “both” we mean that both the client and the repository cooperatively generate the keys. This could mean that each only has a part of the key. By “either” we mean that some keys may be user generated and others may be generated by the repository. The choice is often up to the user.

5.1.5. Application Server Aware

The application server, which could be a web server simply protecting a URL, may or may not be aware of the credential repository’s involvement. In the case of token based systems the application server almost certainly is aware since it must know how to interpret the tokens, and it must have established a trust relationship with the token generator. If a web server is performing some standard method of authentication, such as using SSL based public key authentication, it may not be aware that a credential repository was ever involved. The credential repository could simply retrieve the certificates and keys for the browser to use.

5.1.6. Secure Channel

Here we are referring to the channel in which the credentials are transferred. Most systems require SSL or some similar system to protect credentials whether they are tokens or private keys. However, .NET Passport does not require such protection for its tokens. SSL is optional, and as such .NET Passport is vulnerable to replay attacks. Kerberos does not protect the token at all, but it is not an issue since there is a session key inside the encrypted token without which the token is useless. The session key is sent encrypted and hence protected. Additionally, timestamps are used that give replay attacks only a limited window in which they can be mounted. Any system that sends private keys must take care to protect them. This is usually done by securing the channel with encryption.

5.2. Single-Sign-On and Credential Repository Details

In this section, we present the details for each SSO and credential repository system we have reviewed.

5.2.1. .NET Passport

Microsoft's .NET Passport [MSPass] is a Single-Sign-On (SSO) service. By requiring Hotmail, MSN and Windows Messenger users to get a .NET Passport, it has become one of the most prevalent SSOs. Like other SSOs, it uses token based credentials. While the token format is proprietary, it bears similarities to Kerberos [Kaufman] tickets. .Net Passport uses a single ID that is recognized by all Passport enabled sites. This ID is a pseudo-random 64 bit number. Passport is one of the most portable systems that we examined. All that is required is that the client has a cookie capable web browser. SSL support is not even required, though this may not be a benefit as we discuss shortly. While the client is completely portable, the application or web servers using .NET Passport are not. Special software, called the Passport Manager, is required on the server end. Without the Passport Manager, the web server would not be able to interpret or handle the tokens created by Passport.com.

One of our most serious concerns is that .NET Passport has some security vulnerabilities and a troubled past. By not requiring the use of SSL when passing tokens or writing cookies, Passport is vulnerable to replay attacks. Sniffing tokens would be very simple in situations like a café with wireless Internet access. Sniffing is just as simple on shared bus Ethernet and only slightly more difficult on switched networks. Additionally, Passport has had flaws in its registration and administration processes. As recently as May 7, 2003 it had a critical flaw that allowed an attacker to reset any Passport user's password with only minimal effort [BadPass]. In fact the process could be scripted to attack many accounts. Due to a past history of serious vulnerabilities and the un-enforced use of SSL, .NET Passport may struggle to gain further adoption.

5.2.2. Liberty Alliance Identity Federation Framework

As the name indicates, the Liberty Alliance solution is a framework and not an implementation [LibID-FF]. Also, it is more than just an SSO framework: it sets standards for sharing attribute information that is stored in a decentralized manner across entities in a Circle of Trust (COT). Their definition of identity includes attributes about a user in addition to login IDs. The Liberty Alliance solution has two types of entities within a COT: Identity Providers and Service Providers. Identity providers provide the federated authentication. A user will already have an ID at each service provider they use. By linking the IDs at the service providers to an ID at an identity provider, the user can login to the identity provider instead of directly into the service provider. Now a user needs only sign in once at the identity provider to access all of the service providers to which it has linked that ID. And when a user goes to a service provider, not only does he or she not have to login again, but the session appears as if he or she has actually logged in using the local service provider ID. By having multiple identity providers within a COT and linking those IDs, a user could login to any service provider with one of multiple IDs. So there is no single globally unique ID as in .Net Passport, but a multitude of linked IDs. In fact, a service provider does not even know the ID used with the identity provider but rather just a pairwise unique ID number created at the time of the linking between the identity and service provider IDs.

While Liberty Alliance does have those fundamental differences with .Net Passport, it has many similarities [LibInter]. Both are SSOs that use tokens, and both are platform independent. Clients simply need a web browser. Both require application servers to be aware of the system, not only to understand the tokens, but also because a trust relationship must already be

established between entities. For example, service providers must trust identity providers to authenticate users. There is one more major difference between the two systems; the Liberty Alliance framework requires secure channels for transmitting messages and tokens. This protects against replay attacks to which .NET Passport is vulnerable.

5.2.3. Kerberos

Kerberos [Kaufman] is an SSO that is commonly used as a replacement for Network Information Services (NIS) in UNIX environments and in conjunction with Andrews File System (AFS). However, Kerberos can be used as an SSO for almost any application; the applications simply have to be rewritten on the client and server ends to handle Kerberos authentication. This is not always very difficult to do if the applications already support use of Pluggable Authentication Modules (PAM). Because there is not any specific OS restriction on Kerberos, we say it is platform independent in our chart. However, one should take the comments above into consideration before stating it is portable.

Kerberos uses a single, globally unique ID like .NET Passport and unlike the Liberty Alliance framework. Like both of the other SSOs, the application servers are Kerberos aware. Unlike the Liberty Alliance framework, Kerberos does not require secure channels of communication. However, it is still not vulnerable to replay attacks like .NET Passport. The reason is that the token (called a “ticket” in the case of Kerberos) is useless without the corresponding symmetric key that is within the encrypted token. This key is sent to the client encrypted with a key derived from her password so someone who sniffs the token must either crack the encrypted key or the encryption of the token. The best bet would be to attack the key which is encrypted by a function of the password since it allows an offline dictionary attack. To add protection against such an attack, timestamps are also used to prevent replays. Thus the window of opportunity for a replay attack is limited even if the encryption is broken.

5.2.4. Entrust TruePass

By using the Entrust TruePass Portfolio [Entrust] with a Roaming Entrust Profile handled by the Entrust Authority Roaming Server, we have a credential repository system for web servers and application servers that make use of Entrust Digital IDs. When a user goes to a TruePass enabled website, he or she automatically downloads a small Java applet that performs cryptographic operations and certificate validation which communicates with the TruePass modules on the web servers and possible backend application servers. It is these parts that make up TruePass; without an Entrust Authority Roaming Server the TruePass applet can just access IDs locally from smart cards, a Desktop Entrust Profile or the Windows Digital ID store. TruePass normally just handles Entrust Digital IDs, but on the Windows platform it can use the Windows security framework to handle any type of credential stored in the Windows Digital ID store.

Unlike the services described so far, TruePass is a credential repository. The client application retrieves keys and certificates (as part of the Entrust Digital ID) to authenticate to web/application servers via a challenge response mechanism. Entrust claims Windows and Mac compatibility but makes no mention of any other client platform. While it is a Java applet, it is most likely not completely portable. There is actually a separate applet that is half again as large for Windows users because it must be able to interoperate with the Windows Digital ID Store. Also, the fact that it is zero-footprint, meaning data is never written to disk, could inhibit

portability between different operating systems as different operating systems prevent paging to disk in different ways. Key generation can be done either locally or by the credential repository. As noted, the application server must be aware as there are web and application modules that send the client applet to the user and communicate with the applet.

As is necessary with any credential repository sending private data, the communication channel over which credentials are sent is secured with encryption. In fact, if backend application servers are interfaced through a web server, the encryption is persistent beyond the SSL channel with the web server and all the way to the application server with which the applet communicates. The applet also supports PKCS #7 signatures as well as encryption allowing transactions to be signed. There are mechanisms incorporated into the system that use the digital signature support to present text to the client to sign and hence provide non-repudiation and audit trails of transactions.

5.2.5. VeriSign Roaming Service

VeriSign's Roaming Service [VerVideo] shares many similarities to Entrust's TruePass. Both are credential repositories that can store multiple credentials for multiple IDs. Both use private keys and certificates as the credential type. Both download small zero-footprint web applets from web servers. In VeriSign's system this applet is called the Personal Trust Agent (PTA). The PTA handles all of the credential management. Both systems require the use of secure communication channels. A small difference is that the PTA must generate keys locally. In fact, the private key is never even seen by the credential repositories. This is a direct consequence of the unique architecture of VeriSign's credential repository described below [VerWhite].

VeriSign wished to address a problem it saw with most other credential repositories. The typical solution has all credentials stored on a central server protected only by encryption by a key derived from a simple password. This makes these servers a central point of attack from the outside world as well as very vulnerable to insiders who could mount an offline dictionary attack on credential information. They solve this problem by creating stronger secret information than a simple password contains and distributing it across multiple domains. So now when a user goes to a web server that is VeriSign Roaming Service enabled, it first downloads the PTA from the web server and information about where the enterprise roaming server is located. The PTA contacts the enterprise roaming server(s) and authenticates with a hash of the password. The enterprise roaming server returns a salt value that is combined with the password to form part of the larger secret. The PTA does this with all of the enterprise roaming servers to gather pieces of the larger secret. Then it contacts a roaming server controlled by VeriSign to get the rest of the secret in the same manner. Once the PTA has all the parts of the secret, it authenticates to an enterprise roaming server via the password hash to retrieve an encrypted version of the credential. The roaming servers do not know all the parts, and hence cannot decrypt the credential which is encrypted by a stronger key that is derived from the strong secret known only to the PTA. At this point the PTA can authenticate to the web server via a challenge/response protocol.

The PTA is only compatible with Windows. However, a very new product called Personal Trust Service (PTS) is platform independent and can be used in place of the PTA [VerPhone]. Being that this product is so new, there is very little information on it and in our experience not all of the VeriSign technical support personnel are aware of it yet.

5.2.6. NSD Security Practical PKI™

The *Secure Identity Appliance* (SIA) is at the heart of the *NSD Security Practical PKI™* [NSD] that offers solution for users' credential and information storage, along with single sign-on. The appliance is a secure specialized hardware device connected to an organizational network wanting to implement a PKI. Practical PKI™ implements the 3-key RSA algorithm for splitting every user's private key into two components. One component is stored at the SIA. The second component is stored at the user as a password; actually, the second component is derived from the user's password. Thus passwords become the new users' private keys. When users want to use their credentials they must first authenticate to the appliance using a strong password-based method provided by a browser plug-in that users need to download to their computers. The plug-in is compatible with Internet Explorer 4.0 or higher and Netscape 4.7 or higher, and runs on any PC with Windows 95, 98, XP, 2000 and NT as its OS.

Once they have been successfully authenticated the signing process works like this: the user submits, by encrypted means, the information to be signed to the SIA. The SIA uses the user's private key component that it holds to partially sign that information. Then, it is transmitted back to the user who completes the signing process using her private key component. After that, the signature verification is identical to the traditional RSA signing. Single Sign-On can be implemented by two methods: Ticket passing and Client Side SSL. With ticket passing, after a user logs into the server, a ticket (similar to a token) is generated and digitally signed. Then, the ticket is passed to any other server requiring single sign-on. Notice that under this method, web servers and applications must be aware of the technology to understand the ticket. Moreover, they must verify the signature and the "valid until" timestamp present in the ticket. Ticket passing is similar to Microsoft .Net Passport. The second method, Client Side SSL, requires servers be C-SSL compatible and trusting certificates issued by a list of CAs. The CAs can be part of the organization or commercial CAs (like VeriSign or Entrust). The plug-in mediates all operations at the client side helping the user to authenticate and download his or her certificates. Then the certificates can be used by any application supporting the Microsoft *Cryptographic API* (CAPI) or PKCS #11.

One final observation for Practical PKI™ is that private keys must be generated by the SIA to comply with the 3-key RSA algorithm. This also restricts the use of multiple IDs even though Practical PKI™ allows the use of multiple credentials per user.

5.2.7. RSA Keon Web PassPort

Keon Web PassPort [RSA] is the solution from RSA to credential management. Like the solutions discussed above, Web PassPort needs users to download and install a small digitally signed plug-in to the computer they happen to use. As Entrust TruePast, this plug-in implements a zero-footprint virtual (smart) card repository to store user's keys and credentials. Hence, the credentials and keys are available to any application (e.g. web browsers, e-mail clients, etc.) running on Windows OS and complying with CAPI or PKCS #11.

The other main component is the Web PassPort Server that resides on the web server to protect web resources. Users must authenticate to the server to access those resources. After the user successfully authenticates to the web site, the Web PassPort server creates a PKI credential for the user, or retrieves it from an LDAP directory if the credential already exists, and sends it in the form of a virtual card to be stored by the user's plug-in so it can be used by other

applications. These virtual cards can hold up to two certificates with the corresponding private keys. The plug-in handles several virtual cards simultaneously, allowing the user to authenticate to multiple sites or use multiple identities. Additionally, Web PassPort uses cookies to remember the authentication status.

Web Passport Server can be involved in the user key generation process or the PKI credential can be issued by any other commercial CA. In either case, the credentials are stored at the server side, so if the credential is issued by another CA, the plug-in talks with the server to automatically update such credential. The same happens for credential or key updates. The main drawback with this approach is that the server is now a central point of attack by insiders as well as outsiders. A compromise of a Web PassPort server at a particular web site, though, does not imply a compromise of other Web PassPort servers at different web sites.

5.2.8. Baltimore UniCERT Roaming

Baltimore has added a roaming component to its UniCERT PKI [UniCERT] product to allow users to digitally sign messages from almost any Internet browser without using hardware tokens, such as smartcards, or any other software at the client side. Thus, UniCERT Roaming [UCRoaming] is independent of hardware, software, and operating system. Even more, Baltimore assures that its roaming solution can be easily added to any new or existing PKI deployment.

Baltimore achieves this functionality by centrally managing credentials. This approach, which provides flexibility to end users, requires especial secure hardware and software infrastructure to prevent the central server from being a single point of attack. For that, the server side comprises three major components. The *Credential server* controls the operation of the other roaming components. User authentication is a function of the *Protection Encryption Key (PEK) server*. Finally, both servers are hidden from the external internet by a *Proxy server* protecting them from network-based attacks. Baltimore avoids the single point of attack by using two security servers, the Credential and PEK server. As part of the Credential server there is a LDAP directory that stores users' credentials. Someone can consider the directory as a fourth component, but it is really part of the Credential server.

Even though Baltimore claims no software is required at the client side, users need to download to applets to their browsers: the Signing applet and the Change Passphrase applet. The signing applet allows users to connect with the server to retrieve their credentials and sign web data. These credentials can be issued by the Baltimore UniCERT CA or any other industry leading CA, and can refer to different identities. The proxy first filters all connections redirecting users to the PEK server for authentication. Once authenticated, users can have access to the Credential server to download their credentials from an LDAP directory to their computers. The credentials are double encrypted with a key derived from a user's passphrase. Additionally, internal sequence numbers are used to protect credentials from brute force or replay attack. Finally, end users use their credentials to create digital signatures. The Change Passphrase applet is used to change the passphrase that protects the user's signing key.

5.2.9. Microsoft Active Directory Roaming Profiles

Active Directory (AD) is a concept that was first introduced by Novell. An Active Directory is a data structure containing information about objects (e.g. computers, printers, users) within a

network. For users, the information is stored using a special data structure called the *Profile*. With Windows 2000, Microsoft directly supported AD [MSActDir] for the first time and allowed profiles to store PKI credentials. Active Directories have several characteristics that make PKI easy to implement. For example, ADs allow automatic replication and update of profiles that can help in publishing the latest certificates and CRLs. Besides that, the distributed design of ADs allows users to access their profiles from almost anywhere as long as an internet connection and a computer running Windows 2000 or XP are present.

Users' profiles contain PKI credentials issued by a Microsoft CA or any other commercial CA. The credentials give users flexibility in managing and using different identities. The user profile is protected using the *Microsoft Data Protection API* (DPAPI). Under DPAPI, a profile is encrypted with a master key created per user the first time he or she logs in. The master key is encrypted with a 160-bit RC4 key derived from the user's password and store in the profile. Another copy of the master key is also put in the profile but this time encrypted with a derivation of the Domain Controller's master key. From now on, the profile is sent encrypted to the user's computer. The master key must be unlocked using the user's password. Once the master key has been retrieved, it is used to decrypt the key used to protect the user's keys used for authentication and signing purposes. Since DPAPI security depends on the secrecy of the user's password, it makes AD Roaming Profiles vulnerable to a password guess attack.

5.2.10. MyProxy Online Credential Repository

The MyProxy online credential repository was described in detail in section 4 above but here we describe how it compares to other systems according to the taxonomy presented in section 5.1. MyProxy supports multiple credentials (certificates and keys) per user: users can query the repository for the credentials they have stored and retrieve specific credentials by name. Multiple open source client implementations are available, including a portable Java implementation [JavaCoG]. To retrieve credentials from the repository, the client generates proxy keys and sends a proxy certificate request to the MyProxy server to be signed by a private key in the repository, so we can say that keys are generated both at the client and the server. Application servers support standard GSI authentication and are not aware that the proxy credentials were obtained from MyProxy. Finally, all communication between MyProxy clients and servers is encrypted via the SSL protocol.

6. Conclusions

Given the fact that the number of identities a user must manage will increase (this is a common experience for all users), signs indicate that this burden may become even more challenging before it gets any easier. As more functions shift online, E-commerce and E-government transactions are accelerating simultaneously with demands for secure identity management due to terrorism and fraud cost control. For example, the largest web site certifier, VeriSign, issued about 500,000 certificates in 2002, a rise of about 15% from 2001 despite the slow economy.[AP]. Until online identity management systems are uniform, users will have to deal with incompatible and often indecipherable credential requirements with isolated pockets of interoperability.

We propose that the intuitive solution of a "credential wallet" can greatly improve the ease-of-use of security mechanisms that will otherwise be avoided or circumvented by users. The current

identity management alternatives we compare in this paper have clear tradeoffs that make a consensus solution unlikely in the short-term. However long a transition to a common standard make take, a credential wallet catch-all for holding or pointing to different types of credentials for different identities will be an essential accessory – you will want to have it handy wherever you go online just in case you need to prove one of your virtual identities!

References

- [AP] Associated Press. “*VeriSign Seeks to Shake Off Domain Name*”, Jan. 27, 2003.
- [Arsenault] A. Arsenault and S. Ferrell, “*Securely Available Credentials – Requirements*”, IETF RFC 3157, 2001.
- [BadPass] Yen-Ming Chen, “*Untrustworthy Passport*”, 2003. Security Focus Guest Feature. Available at <http://www.securityfocus.com/guest/20225>
- [Butler] R. Butler, D. Engert, I. Foster, C. Kesselman, S. Tuecke, J. Volmer, V. Welch, “*A National-Scale Authentication Infrastructure*”, IEEE Computer, 33(12), pp. 60-66, 2000.
- [Czajkowski] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, S. Tuecke, “*A Resource Management Architecture for Metacomputing Systems*”, Proc. IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing, pp. 62-82, 1998.
- [Dierks] T. Dierks and C. Allen. “*The TLS Protocol Version 1.0*”. IETF RFC 2246, 1999.
- [Entrust] “*Entrust TruePass™ Portfolio*”, 2003. Available at <http://www.entrust.com/truepass/specs.htm>
- [Foster98] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke, “*A Security Architecture for Computational Grids*”, Proceedings of the 5th ACM Conference on Computer and Communications Security, 1998.
- [Foster99] I. Foster and C. Kesselman (Eds), “*The Grid: Blueprint for a New Computing Infrastructure*”, Morgan Kaufmann, 1999.
- [Frey] J. Frey, T. Tannenbaum, I. Foster, M. Livny, and S. Tuecke, “*Condor-G: A Computation Management Agent for Multi-Institutional Grids*”, Proceedings of the Tenth International Symposium on High Performance Distributed Computing (HPDC-10), IEEE Press, 2001.
- [JavaCoG] Gregor von Laszewski, Ian Foster, Jarek Gawor, and Peter Lane, “A Java Commodity Grid Kit,” Concurrency and Computation: Practice and Experience, vol. 13, no. 8-9, pp. 643-662, 2001, <http://www.cogkits.org/>.
- [Kaufman] C. Kaufman, R. Perlman and M. Speciner. “*Network security: Private Communications in a Public World*”. Prentice Hall, Second Edition, 2002.
- [Kornievskaja] O. Kornievskaja, P. Honeyman, B. Doster, and K. Coffman, “*Kerberized Credential Translation: A Solution to Web Access Control*”, USENIX Security Symposium, 2001.
- [LibID-FF] “*Liberty Alliance Project: Introduction to the Liberty Alliance Identity Architecture*”, 2003. Available at <http://www.projectliberty.org/resources/whitepapers/LAP%20Identity%20Architecture%20Whitepaper%20Final.pdf>

- [**LibInter**] “*Liberty Alliance Project: Identity Systems and Liberty Specification Version 1.1 Interoperability*”, 2003. Available at <http://www.projectliberty.org/resources/whitepapers/Liberty%20and%203rd%20Party%20Identity%20Systems%20White%20Paper.pdf>
- [**Linn**] J. Linn, “*Generic Security Service Application Program Interface Version 2*”, Update 1, IETF RFC 2743, 2000.
- [**MSActDir**] “*Microsoft Windows Server 2003 Active Directory Technical Overview*”, 2002. Available at <http://www.microsoft.com/windowsserver2003/docs/ADtechover.doc>
- [**MSPass**] “*Microsoft .NET Passport Review Guide*”, 2003. Available at http://www.microsoft.com/net/downloads/passport_reviewguide.doc
- [**NCSACP**] “*National Computational Science Alliance Certificate Policy*”, Version 0.9.1, June 30, 1999.
- [**Novotny**] J. Novotny, S. Tuecke, and V. Welch, “*An Online Credential Repository for the Grid: MyProxy*”, Proceedings of the Tenth International Symposium on High Performance Distributed Computing (HPDC-10), IEEE Press, 2001.
- [**NSD**] “*NSD Security Solutions Whitepaper*”, 2003. Available at www.NSDSecurity.com
- [**RSA**] “*RSA Keon[®] Web PassPort Technical Overview*”, 2002. Available at http://www.rsasecurity.com/products/keon/whitepapers/passport/KWEBP_WP_0702.pdf
- [**Sandhu**] R. Sandhu, M. Ballere and R. Ganesan, “*Password-Enabled PKI: Virtual Smartcards versus Virtual Soft Tokens*”, Proceedings of the 1st Annual PKI Research Workshop, 2002.
- [**Sarbari**] G. Sarbari. “*Security Characteristics of Cryptographic Mobility Solutions*”, Proceedings of the 1st Annual PKI Research Workshop, 2002
- [**UniCERT**] “*Baltimore UniCERTTM Product Overview*”, 2003. Available at <http://www.baltimore.com/unicert/overview/index.asp>
- [**UCRoaming**] “*Baltimore UniCERTTM Roaming*”, 2003. Available at <http://download.baltimore.com/download/pdf/BaltimoreUniCERTExtendedRoaming.pdf>
- [**VerWhite**] “*New-Generation Roaming Technology from VeriSign*”. Available at <http://www.verisign.com/rsc/wp/roaming/introduction.html>
- [**VerVideo**] *VeriSign Roaming Demo*. <http://www.verisign.com/media/roaming/demo/01/intro.html>
- [**VerPhone**] VeriSign, Inc. Telephone Customer Representative, August 8, 2003.
- [**X.509**] *ITU-T Recommendation X.509 (1997 E): Information Technology - Open Systems Interconnection - The Directory: Authentication Framework*, 1997.

Traditional PKI Certificate Authority	Kerberos Key Distribution Center
key and certificate	password or v5srvtab
user, host, services	user (principal), host (principal), services
off-line, responsiveness measured in days	always online, real-time availability required
proxy delegation, certificate renewal and postdating	ticket forwarding, renewal, and postdating
user handles long-term certificate (usability and security vulnerability issues)	short-term tickets stored in local cache (security vulnerability issue)
key generation off-line per long-term certificate	key generation of long-term shared secret at time new user joins organization and per session via short-term tickets
human operators must verify and approve each certificate request	human operators only needed at time a new user is initially added to the Kerberos database.
bridge CA or hierarchical tree trust relationships	cross-realm ticket forwarding between KDCs
revoked certificates may continue to work unless proactively checked by servers via CRLs	revoked short-term tickets expire within hours
worst-case compromise: all issued certificates revoked, CA abandoned, new certificates issued	worst case compromise: short-term tickets expire, KDC with long-term shared secrets abandoned and replaced

Table 1: A Comparison of Two Established Centralized Key Distribution Approaches for Identity Management

Online Credential Repository	Online CA
leverages existing authentication mechanisms (e.g. password, Kerberos, or LDAP identity databases)	leverages existing authentication mechanisms (e.g. password, Kerberos, or LDAP identity databases)
may handle short-term or long-term credentials, short-term preferred	may handle short-term or long-term credentials, short-term preferred
manages multiple types of credentials (certificates, proxies, Kerberos, AFS, etc.)	provides certificates
key management independent of key generation if keys pre-generated and pre-loaded	key management tied to key generation
trust model varies, from a single user, small groups, virtual organization or CA	CA trust model
usability: GUI dependent upon implementation, users can retrieve credentials whenever and wherever needed via a pass phrase, no burden from enrollment, key generation or key management	usability: GUI dependent upon implementation, users can retrieve credentials whenever and wherever needed via a pass phrase, no enrollment burden but burden on user for key generation and key management
manageability: central point for control and auditing, requires monitoring plus the added cost of managing an off-line CA via aggregated requests in batches per day/week/month	manageability: central point for control and auditing, requires monitoring but does not require operator intervention
renewal services	renewal services
may be populated with only requested credentials; user can store credentials only when needed	credentials issued by request
worst case compromise: requires a separate off-line attack on each credential but if successful all credentials in repository must be revoked and associated CAs abandoned and replaced. However, a repository can be rebuilt with new credentials pre-loaded transparently to the user.	worst case compromise: if compromised, must revoke CA certification, all long-term certificates revoked, short term credentials do not need to be revoked, change CA signing key, new certificates issued.

Table 2: A Comparison Between Two Centralized Key Management Alternatives For Identity Management

Product	Identity	Credential Type	Platform	Key Generator	Application/Web Server Aware	Secure Channel
MS Passport	Single	Token	Any	n/a	Yes	Optional
Liberty Alliance	Multiple	Token	Any	n/a	Yes	Yes
Kerberos	Single	Token	Any	n/a	Yes	Encrypted tokens
Entrust TruePass	Multiple	Keys	Windows/Mac	Either	Yes	Yes
VeriSign Roaming	Multiple	Keys	Any	Client	Yes	Yes
Practical PKI™	Single	Token and Keys	Windows	Both	Optional	Yes
RSA Web PassPort	Multiple	Keys	Windows	Either	No	Yes
Baltimore UniCERT	Multiple	Keys	Any	Either	No	Encrypted credentials
Microsoft Active Directory	Multiple	Keys	Windows	Either	No	Encrypted profiles
MyProxy	Multiple	Keys	Any	Both	No	Yes

Table 3: Comparison of Online Credential Repositories