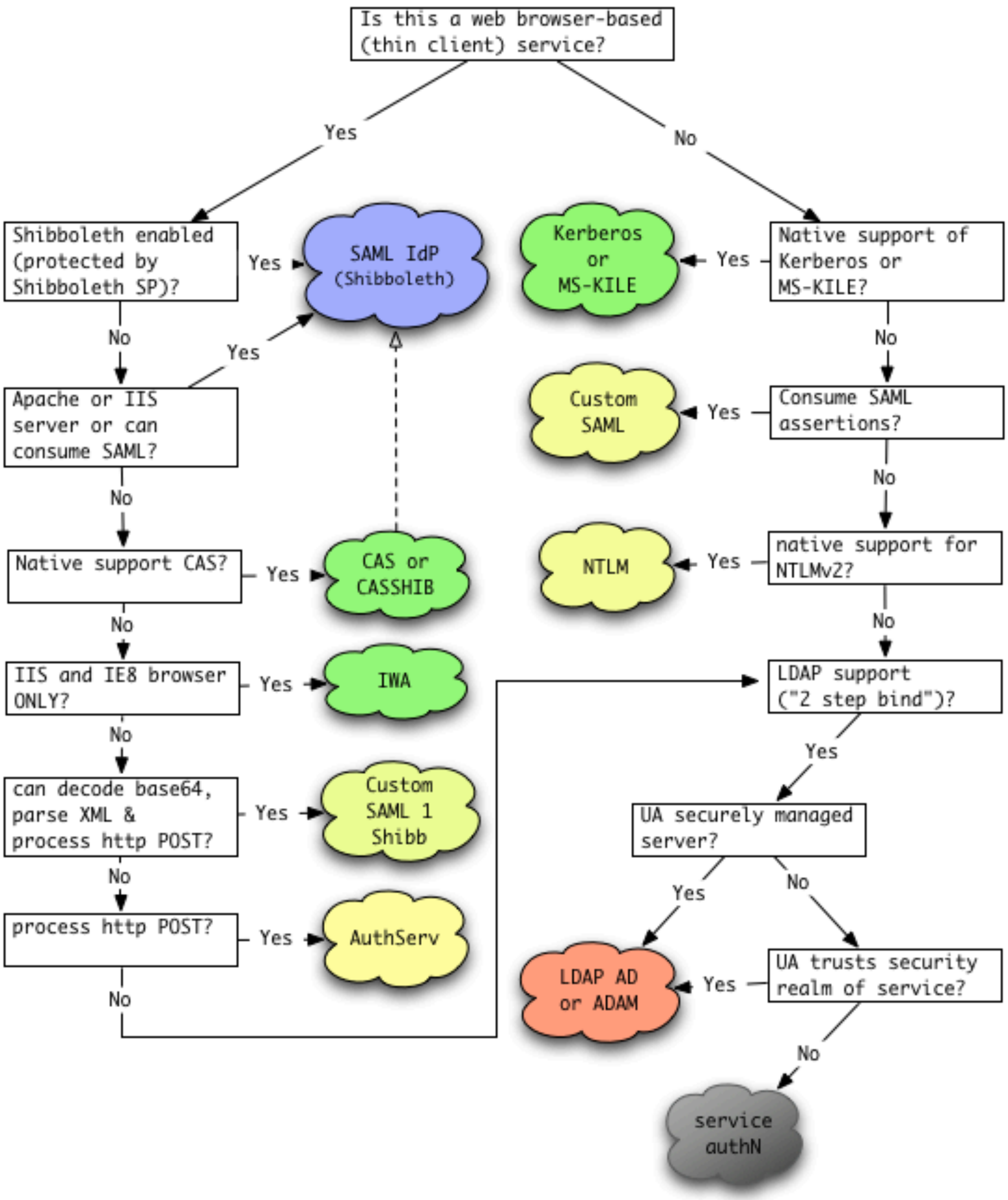
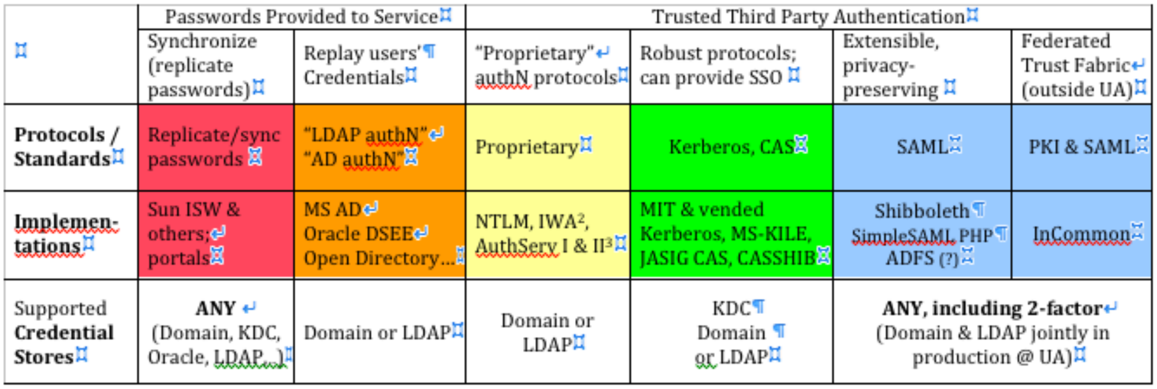
Simplified decision tree for recommended UA central authentication of a service


Is this a web browser-based (thin client) service?
Yes
No
Shibboleth enabled (protected by Shibboleth SP)?
Yes
SAML IdP (Shibboleth)
Kerberos or MS-KILE
Yes
Native support of Kerberos or MS-KILE?
No
No
Yes
Apache or IIS server or can consume SAML?
Custom SAML
Yes
Consume SAML assertions?
No
No
Native support CAS?
Yes
CAS or CASSHIB
NTLM
Yes
native support for NTLMv2?
No
No
IIS and IE8 browser ONLY?
Yes
IWA
LDAP support ("2 step bind")?
No
Yes
can decode base64, parse XML & process http POST?
Yes
Custom SAML 1 Shibb
UA securely managed server?
No
Yes
No
process http POST?
Yes
AuthServ
LDAP AD or ADAM
Yes
UA trusts security realm of service?
No
No
service authN



	Passwords Provided to Service		Trusted Third Party Authentication			
	Synchronize (replicate passwords)	Replay users' Credentials	"Proprietary" authN protocols	Robust protocols; can provide SSO	Extensible, privacy-preserving	Federated Trust Fabric (outside UA)
Protocols / Standards	Replicate/sync passwords	"LDAP authN" "AD authN"	Proprietary	Kerberos, CAS	SAML	PKI & SAML
Implemen-tations	Sun ISW & others; portals	MS AD Oracle DSEE Open Directory…	NTLM, IWA[2], AuthServ I & II[3]	MIT & vended Kerberos, MS-KILE, JASIG CAS, CASSHIB	Shibboleth SimpleSAML PHP ADFS (?)	InCommon
Supported **Credential Stores**	**ANY** (Domain, KDC, Oracle, LDAP…)	Domain or LDAP	Domain or LDAP	KDC Domain or LDAP	**ANY, including 2-factor** (Domain & LDAP jointly in production @ UA)	

Color key:

🟥 Requires repository of all users' passwords at service

🟧 Requires sending users' clear-text passwords to service

🟨 Secure 3rd party authN protocol using superseded or in-house custom interfaces

🟩 Secure & robust 3rd party authN for services / clients supporting protocol

🟪 Secure assertion-based authN & federation for many web-based services

Central Authentication

Central authentication enables multiple services (*i.e.*, applications) to authenticate users relying on credentials stored in a central password store. There is strong interest in many quarters for central authentication as an alternative to each application managing credentials internally:

- central authentication offloads provisioning and de-provisioning credentials from individual applications, reducing the effort required to deploy and maintain services
- central authentication provides a single point at which to provision and de-provision credentials, supporting timely access or denial of access to multiple services depending on the status or role of an individual user
- a single set of credentials (i.e., fewer usernames and passwords to remember) is a significant convenience sought by users
- a single set of crednetials facilitates their access to services with fewer calls for assistance
- trusted 3rd party authentication protocols increase the security of users' credentials
- some central authentication services enable "single sign on" – access to multiple services based on a single user logon event
- some central authentication services can provide additional attributes (e.g., roles) to the service useful for determining the level of service provided (authorization)
- central authentication facilitates participation in a federated trust fabrics, which enable users to authenticate locally (UA credentials provided only to UA

authentication service) for access to external services (vended or at other institutions).

UA system-wide central authentication

Currently, UA user credentials are stored in at least four stores, two of which support generic methods to extend user authentication to other services:

Banner / UA Online stores "PINs" for UA students and employees; while "PIN" suggests a numeric string, these may be alpha-numeric strings (i.e., passwords). While providing access to Banner services and data, this credential store has two limitations making it unsuitable for generic authentication service: (1) PINs are truncated internally at 8 characters – users may enter longer passwords or pass phrases, but only the firsst 8 characters are used. (2) Banner/UA Online does not expose generic authentication methods for using this password store for authenticating external services.

ELMO, the UAS-developed UA service for account claiming, password change, and password reset maintains a record of users' passwords that enables ELMO to check password history when passwords are reset (i.e., to prevent re-use of previously used passwords to increase the level of assurance of password-based authentication). ELMO does not itself provide a generic authentication method.

UA "Unified" Windows Domain provides coordinated password stores for UA students and employees. This infrastructure not only authenticates domain or workstation login, it can support authentication of users to "external" services (i.e., services or applications that operate independent of the Domain) by multiple methods summarized below. Setting passwords via ELMO synchronizes the Domain password with the Open Standards IDM passwords.

Open Standards IDM infrastructure at UA includes LDAP, Kerberos, SAML, and Shibboleth that are explicitly designed to provide central authentication service based on credentials for all students and employees in a Kerberos database using multiple methods summarized below. Setting passwords via ELMO synchronizes this password store with the Unified UA Domain passwords.

Security of central authentication protocols supported at UA based on UA Unified Domain and/or Open Standards IDM

Authentication methods provide different levels of security for users' credentials. Increasing reliance on electronic services has spawned increasingly sophisticated attempts to "crack" passwords or otherwise compromise users' digital identities, in turn triggering a clear trend to deploy or require more secure methods of authentication. While multiple authentication protocols are deployed at UA relying, ultimately, on the same passwords, the security and privacy of users' digital identities should, to the extent possible, tilt toward deploying more modern protocols that better protect those identities. The authentication protocols described below are arranged roughly in order of increasing protection of users' digital identity. Because the more modern protocols also support single-sign-on, this is also roughly in order of increasing convenience for end users.

Password synchronization: The oldest means of enabling the use of UA credentials for a service is to provide the service a copy of all users' passwords so that the service can check passwords against its internal password store. The risk to users' digital identities is clear: additional copies of the password store create more points of attack or accidental exposure through accidental or intentional compromise of security on the external service. If these external stores are regularly "refreshed" to keep them synchronized with a central store, that process itself is a weak point. With users' UA credentials now providing access to sensitive and personal information, replicating UA central password stores should be strongly deprecated.

Replay users' credentials: A simple means of relying on a general purpose central password store without having an internal copy of those passwords is to solicit credentials from users requesting access, then replay those credentials – that is, "log in" to Windows or LDAP Directory (or another service such as an email account) as the user. While this method does not entail the external service maintaining a complete copy of the UA password store, it does expose to the external service the unencrypted password of every user who successfully uses that external service. (While the password can and should be sent over the network via SSL, which encrypts the password in transit, the service must decrypt the message and recover the plain text user password to replay the user's credentials to the central password store; this second transmission – to the password store – should also be a secure encrypted transmission.) Of course, if the external service is "well behaved" it does not cache or otherwise store that unencrypted password or make it available to a third party; but the risk remains of accidental or intentional exposure of users' passwords. Replay should be prohibited for services external to UA, and deployed for internal UA services within UA's secure server zone only if more secure methods are not feasible.

- *LDAP authentication* relies on the open standard LDAP protocol. Users provide their credentials (username and password) to the application; the application then verifies or authenticates the user in two steps: The application (1) queries the LDAP directory to uniquely identify the record associated with the username provided by the user, then, (2) uses the unique record identifier found ("distinguished name" in LDAP) and the password provided by the user to log in or "bind" to the LDAP directory. If the login succeeds, the application has verified the user provided the correct credentials for a person in the LDAP directory.
- *Replay to Active Directory* has the same two-step logic as LDAP authentication, but uses the API of Microsoft's Active Directory. Here is that two-step procedure

described in the .net framework: the authentication code accepts a domain, a user name, a password, and a path to the tree in Active Directory. This code uses the LDAP directory provider. The code in the Logon.aspx page calls the LdapAuthentication.IsAuthenticated method and passes in the credentials that are collected from the user. Then, a DirectoryEntry object is created with the path to the directory tree, the user name, and the password. The user name must be in the domain\username format. The DirectoryEntry object then tries to force the AdsObject to bind by obtaining the NativeObject property. If this succeeds, the CN attribute for the user is obtained by creating a DirectorySearcher object and by filtering on the sAMAccountNameadschema.a_samaccountname….After the user is authenticated, the IsAuthenticated method returns true. < http://msdn.microsoft.com/en-us/library/ms180890(VS.80).aspx>

- *ADAM* (Active Directory Application Mode) is Microsoft's LDAP directory service in a "lightweight" user service mode specifically designed to provide directory services to applications. Microsoft describes ADAM as more scalable than the full AD for simple authentication and other LDAP directory services, and 3rd party benchmarks appear to bear this out.

Trusted Third Party authentication: The technical remedy for the limitations described of credential synchronization and credential replay is trusted third party authentication. Rather than providing credentials to the service (application), users requesting access are re-directed to a third party dedicated to authentication and trusted by both the user and the service. The user authenticates to this trusted authentication service and the application trusts an assertion from that service that the user is authenticated. Thus *the service authenticates users without the users' credentials being exposed in any way to that service*.

Web 3rd Party: UA implementations of Windows Domain and Open Standards IDM provide a method for web-based services (i.e., those accessed via a web browser) to authenticate users by referring them to an authentication service. Both have limitations that restrict their use to certain classes of web services.

- *Integrated Windows Authentication* enables services on Microsoft's' IIS (Internet Information Services) server to automatically authenticate users accessing the service with Microsoft Internet Explorer. The protocol is not readily extensible to services hosted on other web servers or to browsers other than Internet Explorer.
- *UA AuthServ* enables any service that can redirect users and process data placed in a "cookie" by AuthServ. It has been widely deployed for UA-deployed services and has proved simple to deploy and reliable, but of course requires the service to explicitly recognize the AuthServ interface and is thus not readily implemented in closed third party applications. AuthServ's limitations are (1) As an in-house UA development, third party applications are not always adaptable, and (2) maintenance and support is available only internally.

- *CAS – Central Authentication Service* is a more generic web authentication protocol developed by Yale and now deployed fairly broadly in higher education; it provides single-sign-on (like IIS and IE but not AuthServ) and supports most server-browser combinations (like AuthServ but not IIS and IE). CAS thus provides advantages over both IIS/IE and AuthServ but has not been deployed at UA. Shibboleth, described below, generally provides the services of CAS with the additional benefits of supporting policy-based release of attributes and direct support for federation, so a new deployment of CAS would be justified only if a web application could not support Shibboleth but could support CAS.

Kerberos is the original trusted third party authentication protocol in wide-spread use; Kerberos authentication does not require users to send a password but rather use the password to decrypt a message from the server; thus the password need not be transmitted. Both Windows Domain and UA's Open Standard IdM rely on implementations of Kerberos on the "back end." Some applications have the ability to use Kerberos for authentication of users natively (i.e., built in to the application), and these applications could rely on either the Kerberos KDC (the Kerberos password store or "Key Distribution Center") in the UA Unified Domain or UA Open Standards IDM. Kerberos authentication is secure, authenticating users without the password being sent on the network at all, but has other limitations that preclude widespread generic use; (1) Few applications used at UA support Kerberos natively (RADIUS and KeyServer are among the few). (2) Kerberos provides authentication service only; services relying on external authentication often want to learn more than the fact that a user is authenticated: the application may need to know their name, email address, whether they are a student or employee, etc.; protocols such as LDAP, AuthServ, and Shibboleth are able to provision such information so have greater utility to many applications.

MS-KILE (Microsoft Kerberos Protocol Extensions) "provide additional capability for authorization information including group memberships, interactive logon information, and integrity levels as well as constrained delegation and encryption supported by Kerberos principals."
http://msdn.microsoft.com/en-us/library/cc233855(PROT.10).aspx

NTLM (NT LAN Manager)

Windows NTLMv2 is a challenge response protocol for authentication that includes multiple options and supported underlying protocols to maintain some compatibility with older LM and NTLM versions. The details of the authentication protocol itself are not negotiated; they are determined by configuration on both the client and server (see note and tables below). Because it relies on the integrated Kerberos KDC and Kerberos tickets, NTLMv2 enables applications to verify users' authentication without seeing their password or having the password transit the network and can provide true single-sign-on. Because of the integration with AD, applications can also retrieve other attributes (e.g., roles and group memberships).

Windows NTLMv3 appears occasionally in blogs, but does not appear in the Microsoft Developer Network web site.

Windows NTLMSSP (NT LAN Manager Security Support Provider) is a messaging protocol to facilitate NTLM authentication and negotiate integrity and confidentiality options. NTLMSSP is deleted from Vista and Windows Server 2008 http://msdn.microsoft.com/en-us/library/aa480152.aspx

SAML (Security Assertion Markup Language) provides an OASIS standard (Organization for the Advancement of Structured Information Standards) for the exchange of authentication and authorization data between security domains.

Custom SAML coding is currently used to authenticate users to UAF Google Apps.

Shibboleth is an implementation of SAML specifically designed to enable web single-sign-on, federation, policy-based attribute release, and privacy protection developed by the Internet2 community within higher education. InCommon is a federation or clearing house that facilitates trust between federation members. Currently some 200 universities (including UA), vendors, and agencies are members of InCommon.

Potential future authentication infrastructure

- uApprove has been developed by SWITCH to work with Shibboleth to provide end users an explicit indication of which attributes about them are about to be released, and enable to the individual to choose whether to release requested attributes.
- Two factor authentication, requiring a physical token in addition to a password. Two factor authentication can work in our existing Kerberos realm to utilize existing UA credentials in combination with an additional token for those services that require additional level of assurance.
- Shibboleth-like authentication for services other than web based (i.e., for "thick client" applications).
- PKI (public key infrastructure) certificates could utilize very robust public-private key pairs and established trust fabric for authentication with federation. PKI always seems just out of reach for wide-spread adoption at the level of individuals.

From http://technet.microsoft.com/en-us/magazine/2006.08.securitywatch.aspx NTLMv2 is turned on using the LMCompatibilityLevel switch (known as some variant on "LAN Manager authentication level" in Group Policy). LMCompatibilityLevel takes six different values, from 0 to 5. The levels are shown in Figures **3** and **4**.

Server-Side LMCompatibilityLevel Impact

Level	Group Policy Name	Sends	Accepts	Prohibits Sending
4	Send NTLMv2 response only/refuse LM	NTLMv2 Session Security	NTLM, NTLMv2	LM
5	Send NTLMv2 response only/refuse LM and NTLM	NTLMv2, Session Security	NTLMv2	LM and NTLM

Client-Side LMCompatibilityLevel Impact

Level	Group Policy Name	Sends	Accepts	Prohibits Sending
0	Send LM and NTLM Responses	LM, NTLM NTLMv2 Session Security is negotiated	LM, NTLM, NTLMv2	NTLMv2 Session Security (Win2K below SRP1, NT4 & 9x
1	Send LM and NTLM—use NTLMv2 session security if negotiated	LM, NTLM NTLMv2 Session Security is negotiated	LM, NTLM, NTLMv2	NTLMv2
2	Send NTLM response only	NTLM NTLMv2 Session Security is negotiated	LM, NTLM, NTLMv2	LM and NTLMv2
3	Send NTLMv2 response only	NTLMv2 Session Security is always used	LM, NTLM, NTLMv2	LM and NTLM