Autenticazione X.509 con Shibboleth + Multiple Authentication Service

Presentation for CINECA
Bologna

Author : Giuseppe Fiameni
SuperComputing Application Innovation Department

www. cineca.it

Outline

- Objectives of our work
- Shibboleth in a nutshell
- X.509 support
- RDBMS support

Who am I

- Computer Scientist
- Leader of Middleware services group of HPC Department
- Leader of the Technology Watch activity in PRACE (www.prare-ri.eu)
- Member of the DAITF (*Data Access and Interoperability Task Force*) for the EUDAT project (www.eudat.eu)
- Leader of the Software Quality Control activity in EMI (European Middleware Initiative) (www.eu-emi.eu)
- Member of the EPOS ICT Board (www.epos-eu.org)

Objectives of our work

Extend Shibboleth system in order to:

- Provide support for X.509 authentication
  - Authentication/authorization/revocation
- Employ different authentication sources
  - RDBMS

Shibboleth in a Nutshell

“Shibboleth is ***standards-based***, ***open source middleware*** software which provides Web Single SignOn **(SSO)** across or within organizational boundaries. It allows sites to make informed ***authorization decisions*** for individual access of protected online resources in a privacy-preserving manner.”

Source: Internet2

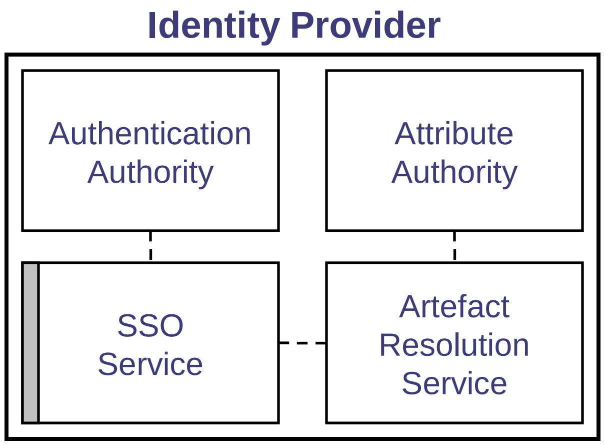
Shibboleth Entities

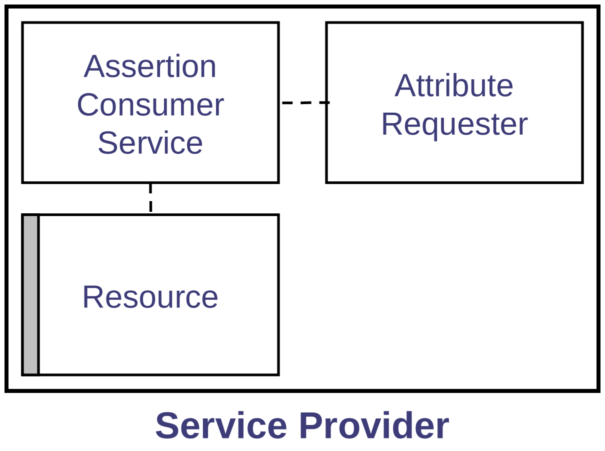
Identity Provider

- The *Identity Provider* (IdP) creates, maintains, and manages user identity
- A Shibboleth IdP produces SAML assertions

Service Provider

- The *Service Provider* (SP) controls access to services and resources
- A Shibboleth SP consumes SAML assertions





Identity Provider

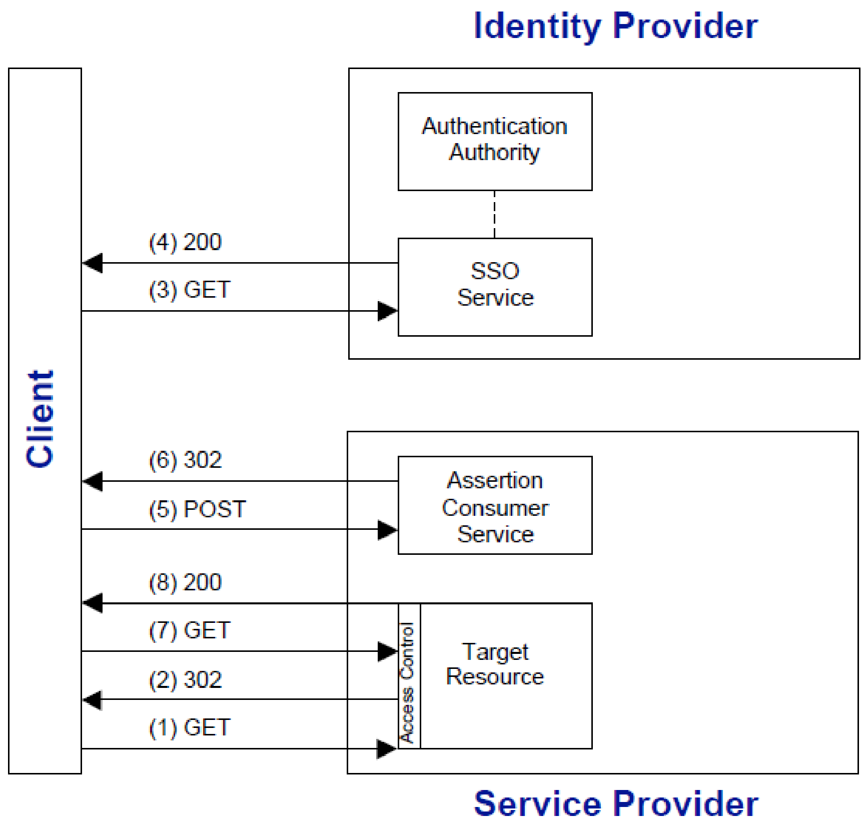
- Authentication Authority
  - Produces SAML authentication assertions
- Single Sign-On Service
  - A (SAML2) browser-facing component
  - Orchestrates SP-first browser profiles
- Artifact Resolution Service
  - Resolves SAML artifacts into assertions
- Attribute Authority
  - Produces SAML attribute assertions

Service Provider

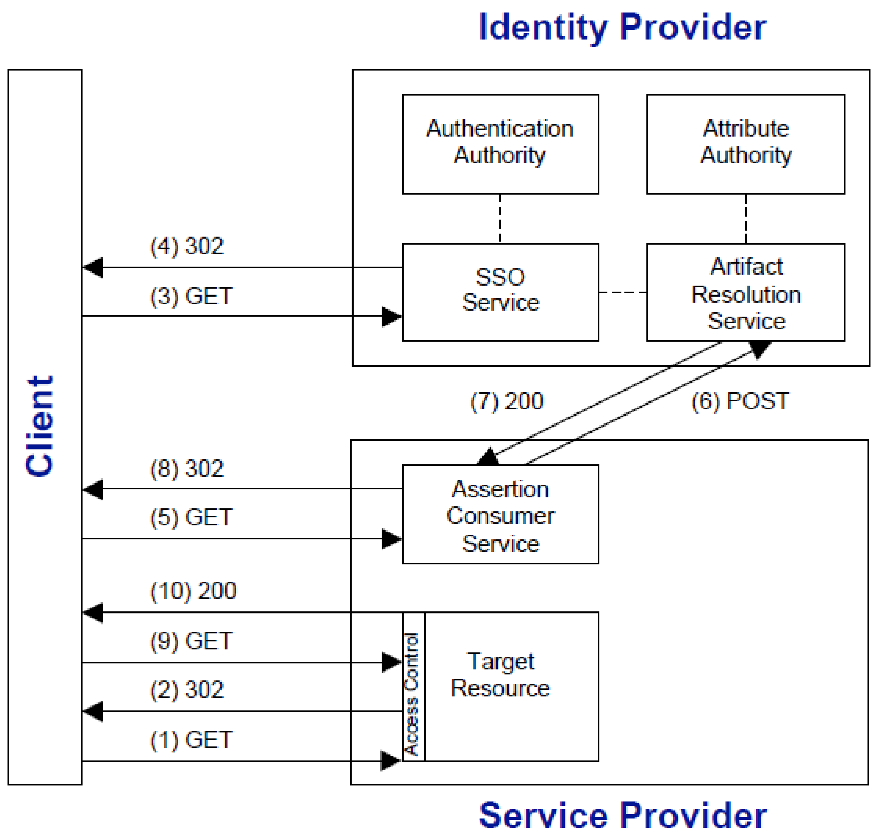
- Assertion Consumer Service
  - A browser-facing component
  - Participates in the browser profiles
  - Consumes SAML authentication assertions
- Attribute Requester
  - Consumes SAML attribute assertions
- Resource Manager
  - Protects web resources

Authentication hand-shake

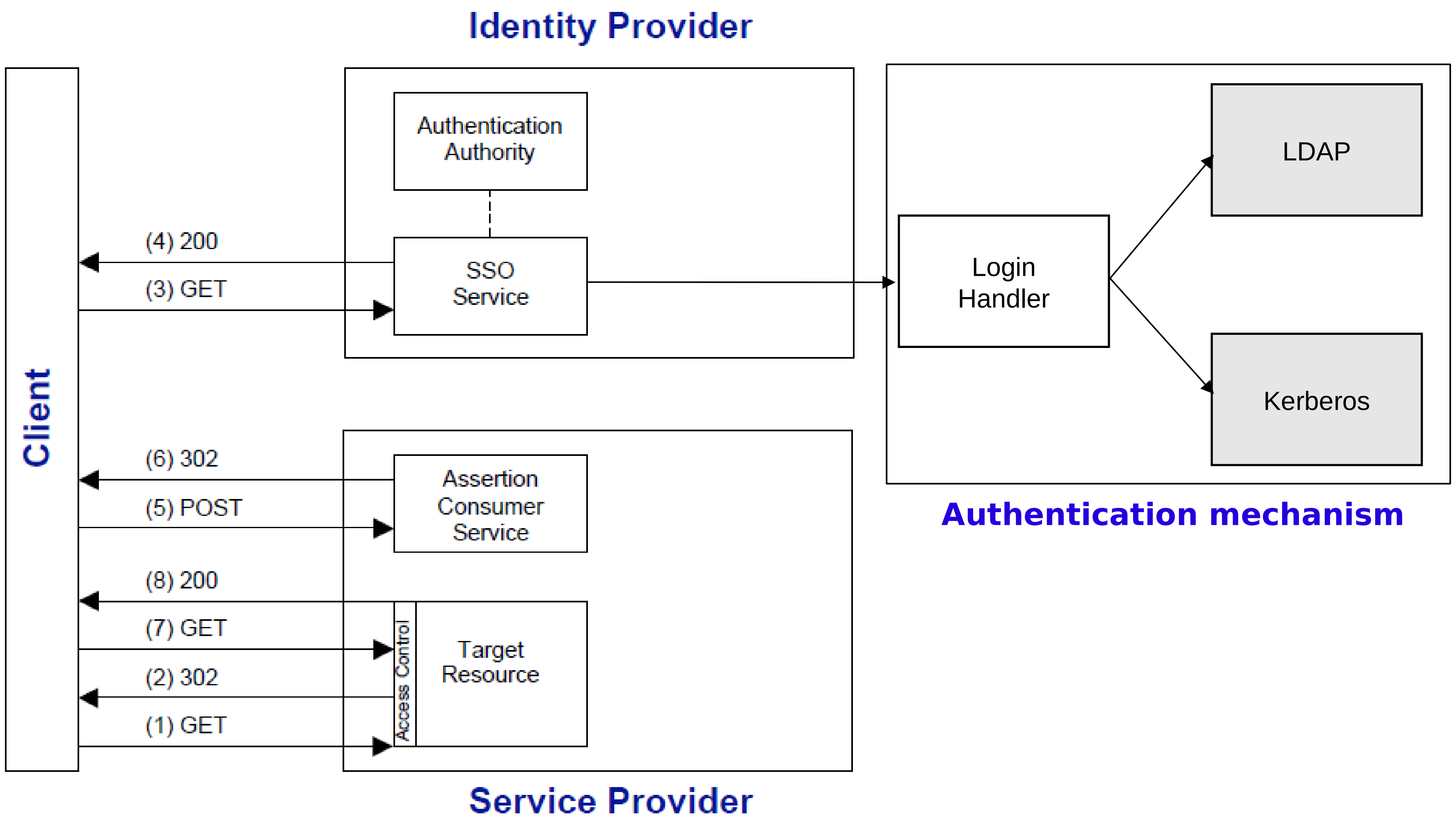
1st attempt



From 2nd on…



Authentication hand-shake



Authentication mechanism

- It is a mechanism used to authenticate a user
- Shibboleth 2 supports the following authentication mechanisms:
  - Remote User
  - Username/Password (LDAP, Kerberos)
  - IP Address

Login Handler

- An IdP component that manages authentication mechanisms
- Login handlers are defined in *handler.xml*
- Defined by <*LoginHandler*>
- Must have a type (*xsi:type*) and at least one authentication method
- Each type has its own set of configuration attributes

Login Handler: User/Pass

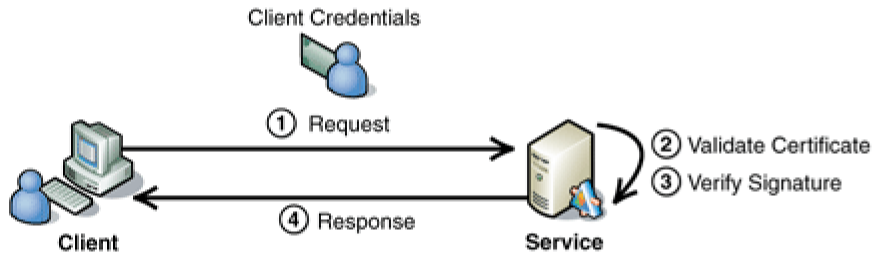
- Login handler that prompts for a username and password
- Validates against a JAAS module
- LDAP & Kerberos 5 supported
- Type:
  - *UsernamePassword*
- Configuration attributes
  - *jaasConfigurationLocation*

Login Handler: *MultiLogin*

- Login handler that prompts for a username/password or **X.509 Certificate**
- Validates credentials against a JAAS module
- LDAP & Kerberos 5 supported + **RDBMS (Various encryption algorithms are supported)**
- Type:
  - *MultiLogin*
- Configuration attributes
  - *jaasConfigurationLocation*

NAMESPACE = "urn:mace:cineca.it:shibboleth:multilogin"

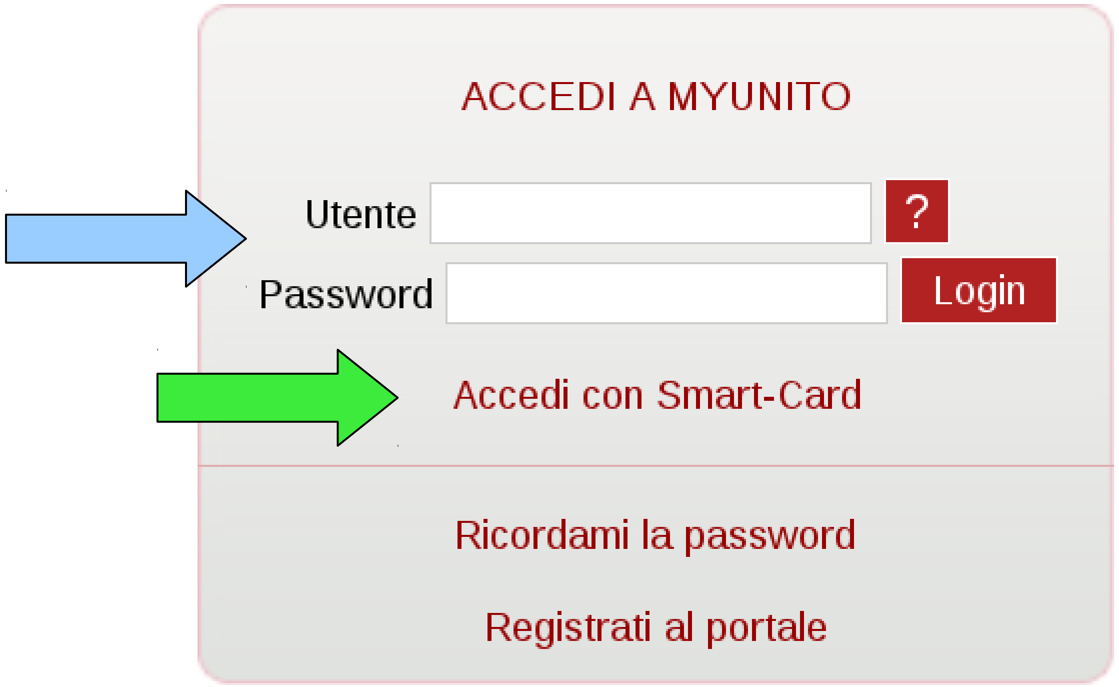
X.509 Authentication



1. The client sends a message to the service. The message includes the client's credentials, signed with the private key that is paired with the public key in the client's X.509 certificate.
2. The service validates the certificate, by performing a number of checks, including:
   - *Verifying that the certificate has not expired.*
   - *Verifying that the certificate is internally consistent.*
   - *Verifying the issuing CA of the client's X.509 certificate.*
   - *Verifying that the issuing CA has not revoked the certificate.*
   - *The service uses the public key in the client's X.509 certificate to verify the client's signature. This allows the service to authenticate the client and ensure that the signed data has not been tampered with after the message was signed.*

MultiLogin: how it works

- User can select which authentication systems to access with



ACCEDI A MYUNITO

Utente

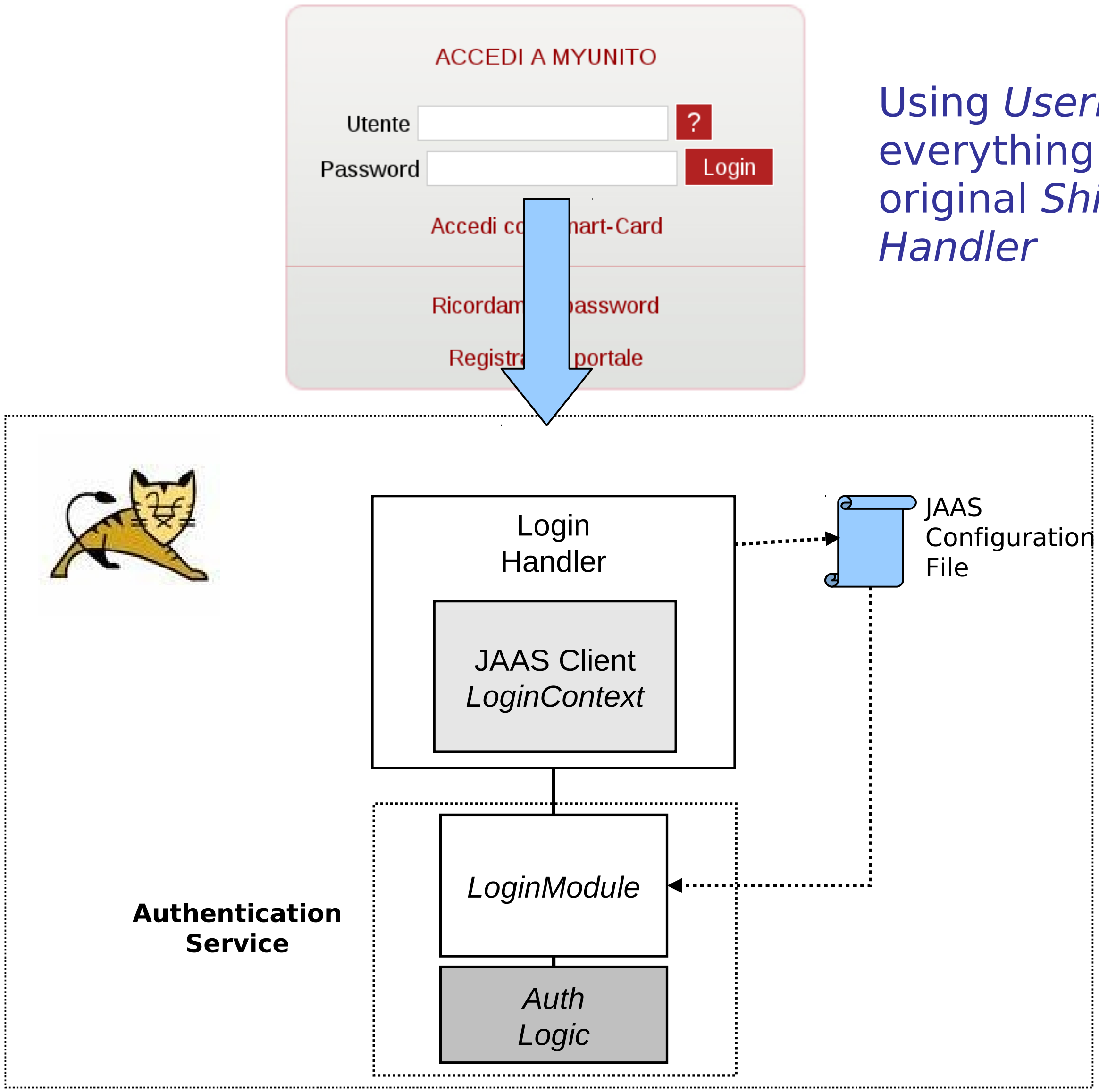
Password

Accedi con Smart-Card

Ricordami la password

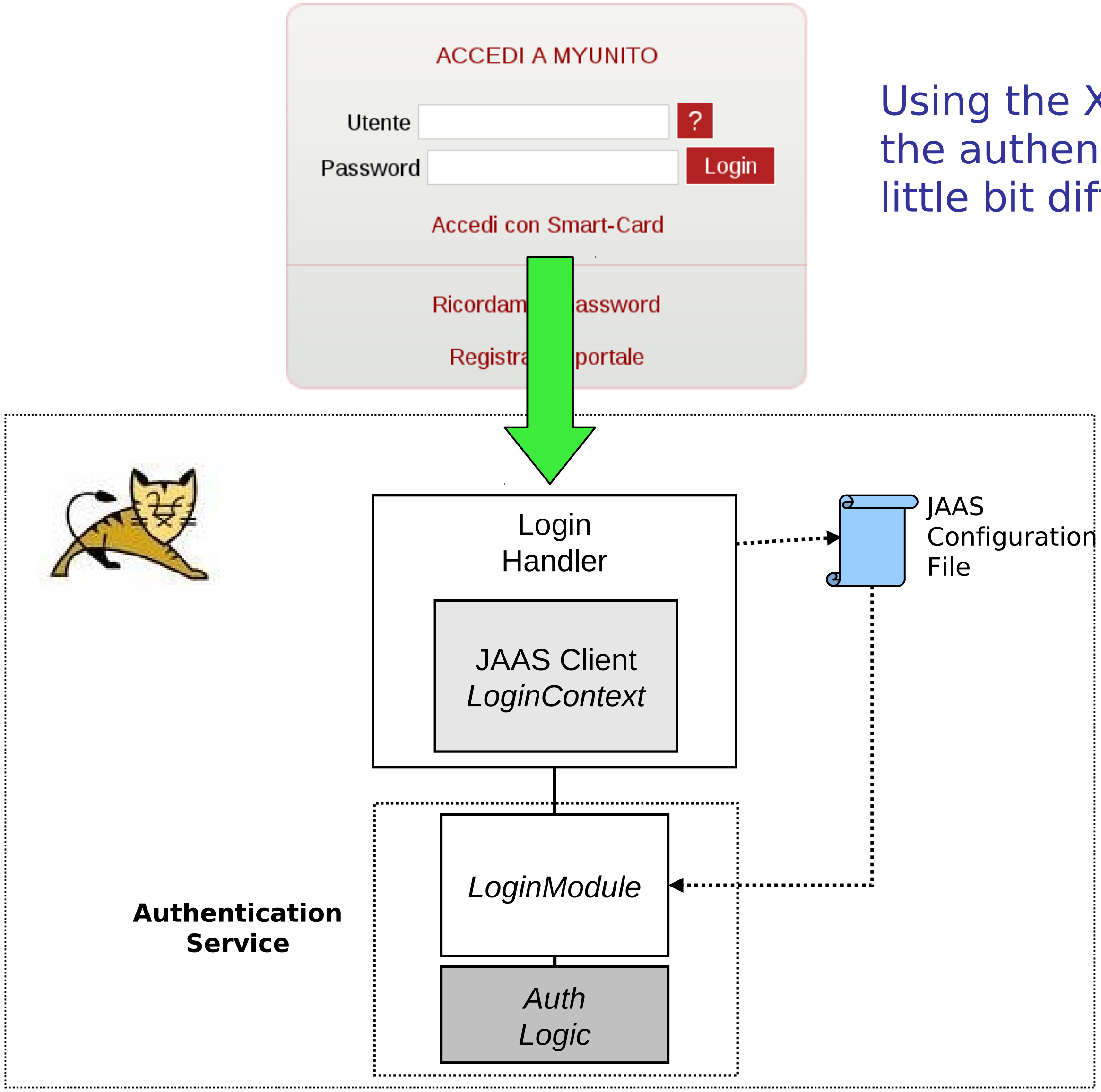
Registrati al portale

MultiLogin: how it works



Using *Username/Password* everything looks like the original *Shibboleth Login Handler*

MultiLogin: how it works



Using the X.509 Certificate the authentication flow is a little bit different...

Login Handler: *MultiLogin*

- The *Username/Password* and *X.509* login handlers cannot easily "**coexist**", neither be configured in cascade
- X.509 authentication occurs at a higher level (SSL not HTTP) implying the application container (Apache Web Server or Tomcat) to take part of the game since the IdP application would not be able to handle the SSL protocol
- Apache might be configured to **optionally** take the certificate from the browser but it will not work with most of the browsers (not standard)
- If a X.509 certificate is required, and the user does not hold anyone, he/she will never reach the user/pass form :(

Apache configuration

```
# don't use SSLv2, as it has fundamental design problems
SSLProtocol all -SSLv2
# all ciphers using strong encryption (Triple-DES)
SSLCipherSuite HIGH
SSLRequireSSL
## require a client certificate which has to be directly
## signed by our CA certificate in ca.crt
SSLVerifyClient require
SSLVerifyDepth 1
# depth of 1 means the client certificate has to be signed by a CA which is directly known
to
# the server
SSLCACertificateFile /etc/ssl/test/cacert.pem
## Location of the Webserver Certificate & private key
SSLCertificateKeyFile /etc/ssl/test/server.key
SSLCertificateFile /etc/ssl/test/server.crt
## Location of the Revocation List
SSLCARevocationFile /etc/ssl/test/test.crl
## Export environment variables for php
SSLOptions +StdEnvVars
```

SSLVerifyClient

SSLVerifyClient

Name:	**SSLVerifyClient**
Description:	Type of Client Certificate verification
Syntax:	`SSLVerifyClient` *level*
Default:	`SSLVerifyClient none`
Context:	server config, virtual host
Override:	*Not applicable*
Status:	Extension
Module:	mod_ssl
Compatibility:	mod_ssl 2.0

This directive sets the Certificate verification level for the Client Authentication. The following levels are available for *level*:

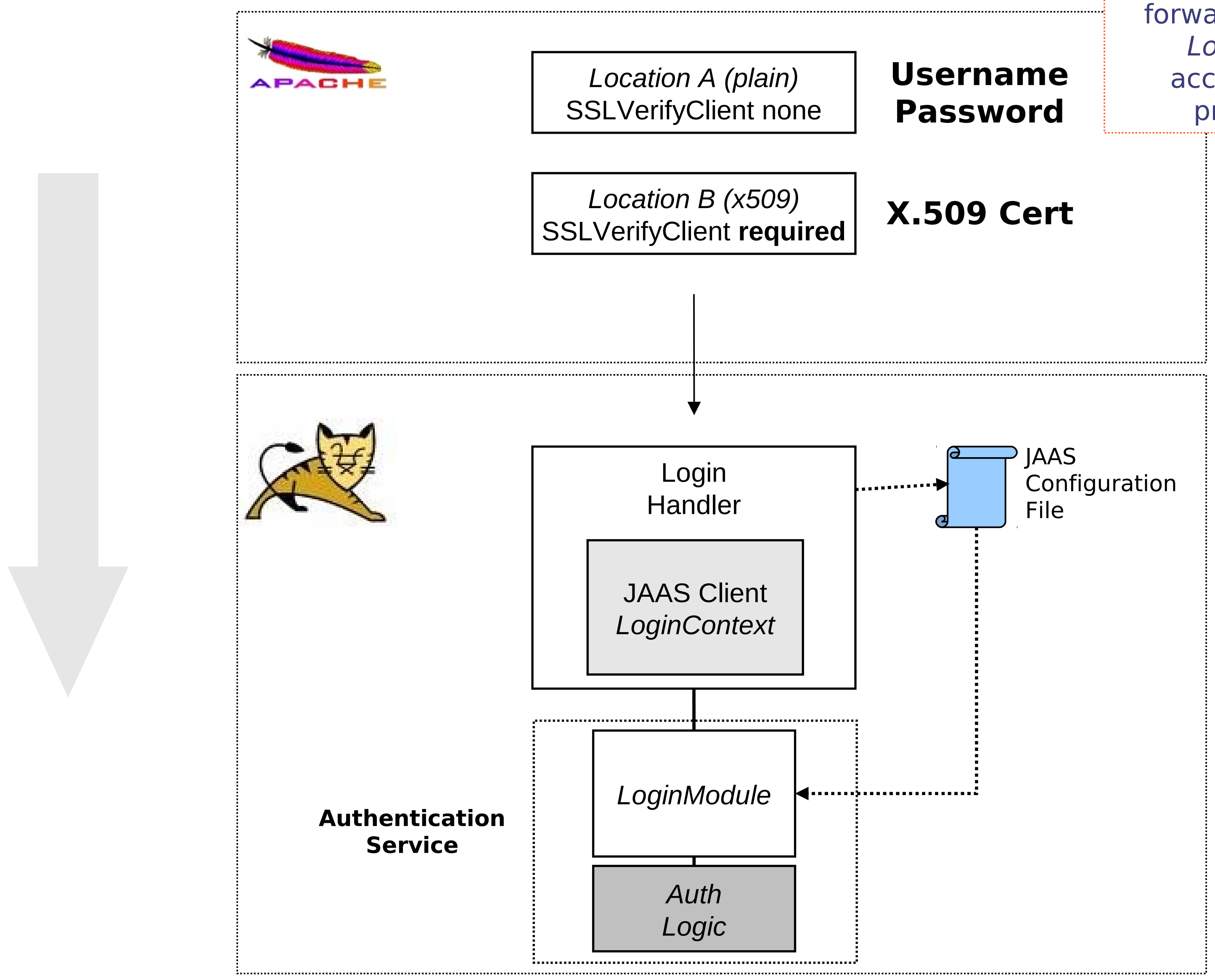
- **none**: no client Certificate is required at all
- **optional**: the client *may* present a valid Certificate
- **require**: the client *has to* present a valid Certificate
- **optional_no_ca**: the client may present a valid Certificate but has not to be (successfully) verifyable.

In practice only levels **none** and **require** are really interesting. Because level **optional** doesn't work with all browsers and level **optional_no_ca** is actually against the idea of authentication (but can be used to establish SSL test pages, etc.)

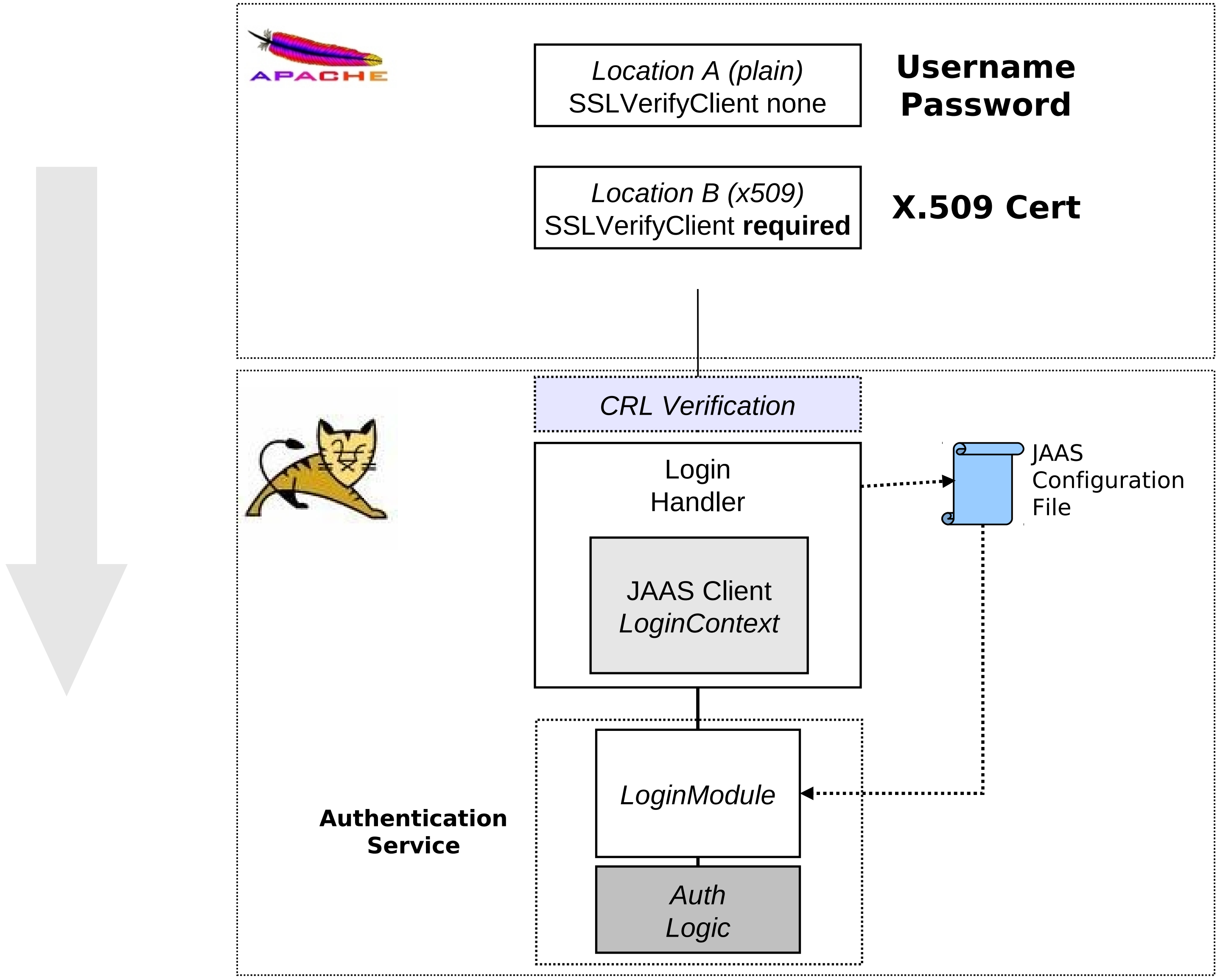
Source: www.modssl.org

Solution

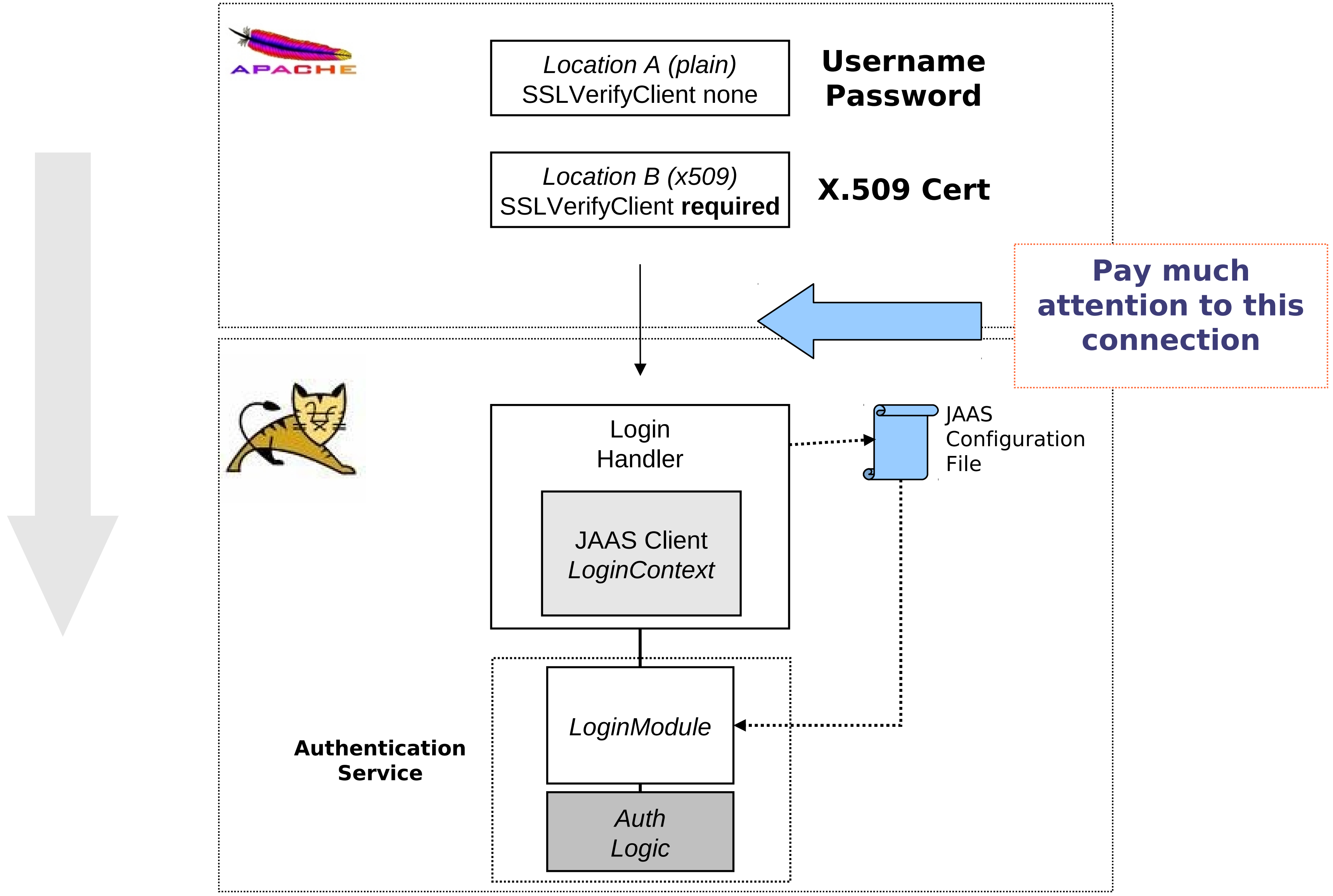
User request is forwarded to either *LocA* or *LocB* according to its preferences



Solution


Location A (plain)
SSLVerifyClient none
Username
Password
Location B (x509)
SSLVerifyClient required
X.509 Cert
CRL Verification
Login
Handler
JAAS Client
LoginContext
JAAS
Configuration
File
Authentication
Service
LoginModule
Auth
Logic

Solution

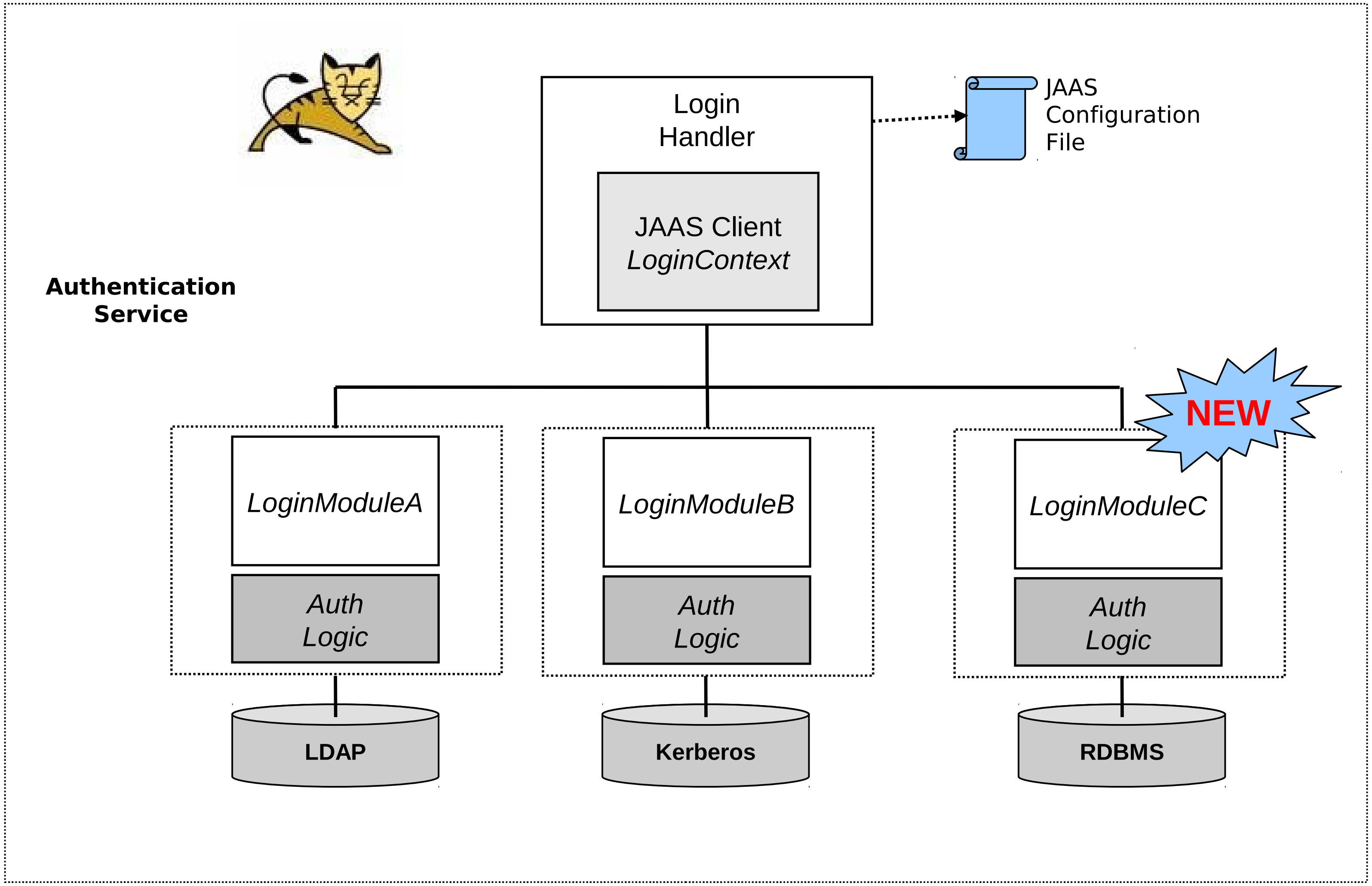

Location A (plain)
SSLVerifyClient none
Username
Password
Location B (x509)
SSLVerifyClient required
X.509 Cert
Pay much attention to this connection
Login
Handler
JAAS Client
LoginContext
JAAS
Configuration
File
LoginModule
Auth
Logic
Authentication
Service

Objective 2

Employ different authentication sources

RDBMS

RDBMS Support



RDBMS Support

- Users credentials are validated against a RDBMS
- Different hashing methods are supported for password obfuscation
- Different DB sources can be configured in cascade (JAAS native)
- The name of the driver to use for accessing the DB is part of the configuration as well as the query to extract the records

JAAS Configuration

```
ShibMultiAuth {

   com.cineca.shibboleth.jaas.DbLogin sufficient
      dbUrl="jdbc:oracle:thin:@IP:PORT:SID"
      dbDriver="oracle.jdbc.OracleDriver"
      dbUser="<username>"
      dbPass="<password>"
      userQuery="SELECT USERNAME FROM SC_CERT_USERS WHERE
SERIAL_NUMBER=?"
      roleQuery="SELECT USERNAME FROM SC_CERT_USERS WHERE USERNAME=?"
      authMethod="auth_method_client_cert";
      ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^


};
```

Shibboleth native X.509 support

- Shibboleth already provides a native *LoginHandler* to permit X.509 authentication but it :
  - *does not support different authentication sources;*
  - *relies only on Apache for the verification of the CRL (Certificate Revocation List). This means that the IdP is not able to know the reason why an authentication attempt may fail.*

https://wiki.shibboleth.net/confluence/display/SHIB2/X.509+Login+Handler

Conclusion

- Developing a new extension for Shibboleth is quite easy, like developing a Spring module
- Shibboleth architecture looks well organized and robust. However, any *LoginHandler* is responsible for the authentication of the users: *code it with much attention*!
- Why Shibboleth does not natively support DB based authentication is still a *mystery*

Q
&
A