# Extending UNICORE 5 Authentication Model by Supporting Proxy Certificate Profile Extensions

Katerina Stamou[1], Fredrik Hedman[1], and Anthony Iliopoulos[2]

[1] Center for Parallel Computers (PDC), KTH, SE-100 44 STOCKHOLM, Sweden
{kstamou,hedman}@kth.se
[2] Department of Computer Science, School of Systems Engineering, The University of Reading, Whiteknights, READING RG6 6AY, Uk
ailiop@teilam.gr

**Abstract.** Authentication interoperability between the UNICORE grid middleware system and other Grid middleware systems is addressed. An approach to extending the UNICORE authentication model to support a proxy certificate (RFC3280) profile is presented. This optional feature can then be enabled based on site policy. Furthermore, the addition capacitates further advances related to authorization. With interoperability becoming a key issue in many production environments, extending the generality of UNICORE in this way opens up the possibility of direct and general interoperability scenarios.

**Keywords:** UNICORE, proxy certificates, RFC 3280, X.509, grid, interoperability, authentication, authorization.

## 1 Introduction

This paper describes an implementation of proxy certificate profile [1] authentication support in the gateway component of the UNICORE release 5 grid middleware system. The motivation behind this effort is to enable and promote functional interoperability among the different existing grid systems currently in production [2].

The existing authentication model of UNICORE allows for end-to-end plain X.509 mutual authentication between clients and UNICORE services. The proposed extension introduces support for verifying and authenticating against proxy certificates, thus allowing a wider and more flexible range of credential management in the UNICORE architecture. Numerous kinds of setups will benefit from the appearance of the features that the proxy certificate profile offers, the most significant being a controllable restricted form of trust delegation. This enables single sign-on like functionality allowing the possibility of disconnected operations that many of the other grid middleware systems already include. Environments that have a large grid deployment base and make use of UNICORE in parallel with other grid systems can benefit from the proposed functionality enhancement.

All work conducted pertains to UNICORE release 5, and is generally directed towards the pre-WebServices era systems.

## 2 The UNICORE Authentication Model

The UNICORE 5 release [3] is the well-tested, production-ready and pre-Web Services version of the UNICORE Grid middleware [4]

The system supports end-to-end user X.509 certificates, as main tokens of authentication. The UNICORE gateway service acts as the central entrance point for task submissions, and takes care of authenticating incoming client requests based on their presented X.509 certificates. The gateway maintains a configurable list of trusted Certificate Authoritiy (CA) certificates, thus clients are considered trusted and successfully authenticated only if their X.509 certificate is signed by a trusted CA.

Following a successful authentication, the gateway forwards the user X.509-signed Abstract Job Object (AJO) task to a configured Network Job Supervisor (NJS). There takes place the authorization part. The NJS detects if the X.509 certificate that signed the supplied AJO task exists in the Unicore USer Data Base (UUDB). All entries in the UUDB are comprised by an end-user X.509 certificate and a UNIX login account name. Upon successful authorization, the requested task executes having the privileges of the local UNIX user that the UUDB entry maps to.

Overall, this process is able to operate with ordinary X.509 certificates. There is no provision for restricted privilege delegation[1] which is present in other Grid middleware efforts (e.g. gLite, Globus, Naregi) and proves to be a highly desired functionality for Grid operations. This static credential management scheme is secure but lacks some flexibility. The proxy certificate profile approach comes to cover this functionality gap and offer a finer-granularity control.

## 3 Implementation of the Proposed Enhancement

Proxy X.509 certificate extensions provide a means to achieve trust delegation and restriction of delegated privileges [1]. In general terms it forms a simple and adequately secure approach that can be easily supported by extending an existing SSL/TLS-enabled server.

In our development setup, we have used SUN's JDK release 5 and OpenSSL toolkit, release 0.9.8d. For applications written in the C/C++ programming language the OpenSSL library [5] is the most widely used; it has full support for proxy certificates. The Java API does not natively support proxy X.509 extensions. However, the Java language provides a way to override the default TrustManager methods [6] which are responsible for making the authorization decisions during the relevant SSL handshake phase. Because third-party external

[1] Except from the "Explicit Trust Delegation" scheme, which does not offer the flexibility of proxy certificate extensions.

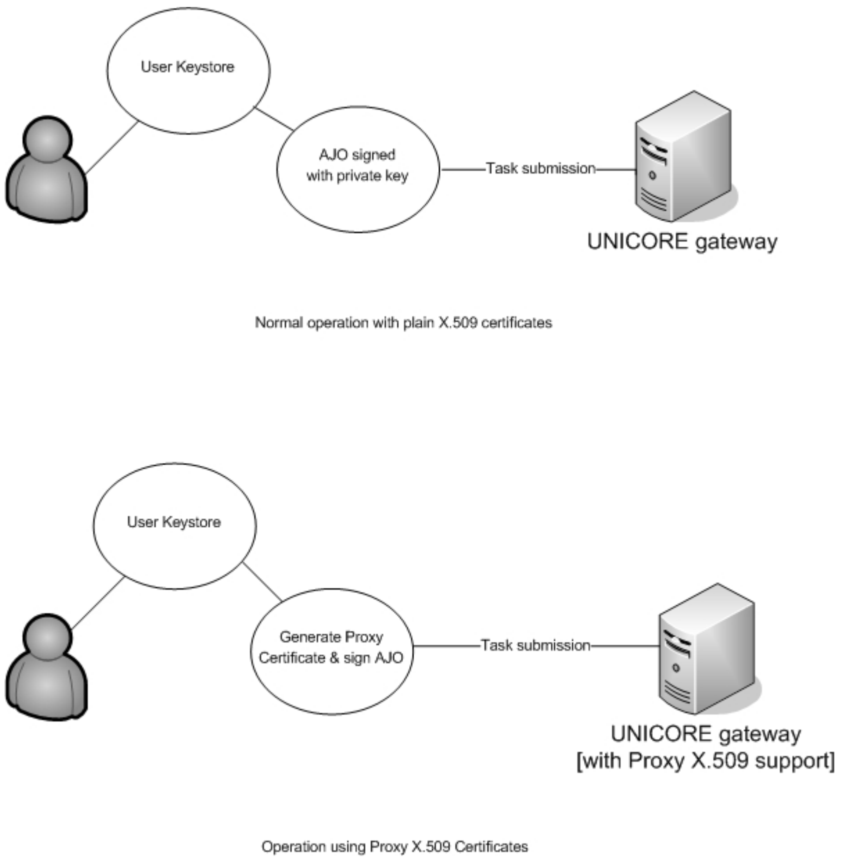
User Keystore
AJO signed with private key
Task submission
UNICORE gateway
Normal operation with plain X.509 certificates
User Keystore
Generate Proxy Certificate & sign AJO
Task submission
UNICORE gateway [with Proxy X.509 support]
Operation using Proxy X.509 Certificates

**Fig. 1.** Diagram of operation with & without proxy certificates. During normal operation the user's private key from the keystore is used to sign the AJO task to be submitted. In the proposed extension, a proxy certificate is generated and then used to sign the AJO.

libraries, such as BouncyCastle [7], have yet to include full support for proxy certificates, implementing a customized TrustManager class is thus a solution to access the functionality provided by proxy certificates from Java.

The relevant TrustManager method in the UNICORE gateway responsible for the authentication decisions is the `checkClientTrusted` method. Our customized version of this method works as follows:

- Initially, it instantiates a default TrustManager which attempts to verify the supplied client certificate chain, in order to cover authentication of connections using non-proxy X.509 certificates.
- If this fails, it attempts to verify the client certificate chain using an external proxy path certificate validation algorithm [1].
- Ultimately, if this fails too, the method raises a standard validation exception, indicating that the client certificate was not validated.

The external proxy path validation is realized by the `ProxyPathValidation` class provided by the COG-JGlobus [2] project [8]. The supplied "validate" method is able to handle rfc3820-based proxy certificates, as well as the complete range of the old-style legacy Globus proxy certificates.[3] The scheme is easily integrated to the existing gateway classes and specifically the `ListenerManager` class that handles the incoming connections, as an inner class. Several jar bundles have to be included to the gateway libraries, mainly the cog-jglobus.jar (that actually provides the `ProxyPathCertificate` class) along with some other jar files that in turn JGlobus depends on (puretls, cryptix, etc.)

Figure 1 depicts how a normal client interaction operation occurs and how an operation using proxy certificates takes place.

## 4 Validation Testing

A demo CA was created using OpenSSL. The enhanced gateway was configured to trust certificates issued and signed by this CA. A user certificate and a corresponding user key were issued by this demo CA. Subsequently, proxy certificates were issued based on that user certificate by OpenSSL.

It was verified that the enhanced gateway could handle all kinds of proxy-style certificates (RFC or not), by using the grid-proxy-init tool of both the Globus toolkit and the gLite in order to generate the various certificates.

The following comprehensive test cases were realized, in order to validate the correctness of the implementation using the OpenSSL client functionality with different kinds of proxy certificates, and connecting to the enhanced UNICORE gateway to verify the authentication decision:

- Connecting using a plain (non-proxy) user certificate signed by a trusted CA
- Connecting using a plain (non-proxy) user certificate signed by an non-trusted CA
- Connecting using a proxy user certificate signed by a user certificate which in turn was signed by a trusted CA
- Connecting using a proxy user certificate signed by a user certificate which in turn was signed by a non-trusted CA
- Repeated the above two test-cases using various types of proxy-style certificates
- Repeated the tests for expired proxy certificates

At the point where the tests proved successful, the functionality of the enhanced gateway was fully verified and it was asserted that proxy signed jobs could be executed. This was realized by using the UNICORE CLI to build a standard sample task (AJO) and submit it to the proxy-enabled gateway.

The above tests should cover most of in not all of the potential user cases, and were thoroughly successful.

[2] Specifically, release 1.4 of the COG package.

[3] The usual technique used in pre-RFC era proxy certificates format, was to prepend the subject line with a "CN=proxy" entry.

## 5 Towards Grid Middleware Interoperability

The main obstacles encountered in trying to achieve full interoperability between the various middleware software systems, are the different tokens of authentication and authorization used, as well as the specific language that each system uses to describe the task to be submitted and various restrictions and requirements specific to that. In our current efforts [2] the main interest is concentrated around the interoperability between the major middleware systems, namely UNICORE, gLite and Globus. Adding support for proxy certificate authentication in the UNICORE system with the proposed solution completes the first aspect of interoperability, as proxy certificates are the primary form of authentication as well as authorization (when used in conjunction with attribute certificate extensions [9]) tokens in the other middleware systems.

Several projects have focused on the other half of the interoperability problem, specifically the inter-mapping of the various job languages. For example, GlobusTSI [10] that made a UNICORE job submission transparently compatible with Globus NJS backends. This work is currently deprecated and not compatible with the current releases of Globus and UNICORE. Other efforts are underway to transform JSDL [11] submissions into AJO tasks and conversely [12].

From an authorization angle, the UNICORE model lacks sophisticated support, as it merely distinguishes users only by utilizing the local username mapping (Xlogin). This essentially delegates the authorization problem to the underlying operating system access control—which in the common case (UNIX, Linux) is simply the discretionary model. The authorization problem in UNICORE is double-faceted: the transport of authorization information within the authentication tokens (signed AJOs) and the actual enforcement of these pieces of information accordingly. The first part of the problem can be fairly trivially solved by enhancing the structure of the proxy certificate, adding attribute information. This scheme has been extensively used by the other middleware systems by embedding various pseudo-attribute extensions to the proxy certificate thus augmenting it to include authorization information [13]. A standardization effort is underway, attempting to formally define a structure for these extensions that have been described by an RFC [9]. Currently, not many software implementations exist, but various endeavors are on-going [14,15].

The second part of the problem, the utilization of the authorization information, could be directly solved by making use of additional features of the underlying operating system (e.g. UNIX user groups), in support of policy enforcement. More elaborate operating system setups, such as those supporting filesystem extended attributes and full-fledged access control lists [16] can be leveraged to fully utilize the supplied authorization information.

Overall, proxy profiles adopt quite well as temporary authentication tokens to convey authorization related information, as the valid life-time of the later is usually very limited, a fact that suites the commonly small validity duration period of the proxy certificates in contrast to plain X.509 that have a much longer lifetime (6-12 months).

## 6 Alternative Approaches and Solutions

### 6.1 Explicit Trust Delegation

Recognizing the need for privilege delegation due to the very dynamic nature of Grids, the UNICORE/FUJITSU team initially proposed an alternative to using proxy certificate profiles, a scheme known as ETD (Explicit Trust Delegation) [17]. This was made to avoid using the then non-standard and not very well supported proxy extensions, which had very few implementations, and while having in mind a trade-off between flexibility and security of the two different solutions.

The proposed solution was to enhance the UNICORE architectural model at its core, introducing an explicit role model to assist in stricter privilege separation. In this scheme, a new role is added, the *User role*, splitting the authority of the legacy *Endorser role* in two separate parts. The end-user retains the User role for creating jobs, while the *Endorser role* is only retained for the authorization action. The *Consignor role*, is typically held by either the end-user to transfer the signed AJO to the server, or by a UNICORE server transporting a sub-AJO to another server. Thus, the user commonly assumes all three roles (User, Endorser and Consignor) for submitting jobs. The servers have the ability to not only consign jobs, but endorse them on behalf of the user, achieving a strict form of trust delegation with the restriction that only the end-user can act in the User role and create jobs.

Although overall this scheme is efficient and secure, it currently also excludes the desired interoperability property that the proxy certificates provide.

### 6.2 The CONDOR-UNICORE Bridge

Condor [18] is a job queuing/scheduling system aiming at supporting high-throughput computing by providing a sophisticated workload management system. It supports functionality for smart matching of requested resources with available resources. As Condor attempts to be quite flexible as a system, it can be used as a component for building larger systems comprised of numerous distributed Condor installations as well as other Grid-like environments.

Condor has mostly been supporting submission of jobs to foreign Grid engines, mainly the various Globus Toolkit releases (GT2+). Recently, the Condor team has introduced a new model for easy integration of job submission "backends" requiring no modifications on the main Condor sources. A proof-of-concept implementation of this new framework was the Condor-UNICORE Bridge [19]. This approach has the advantage that it

– smoothly integrates within existing deployments. No source code or any other kind of configuration modifications are required in order for this to work with UNICORE.

The disadvantages are that it

– introduces one additional software layer. Installation, configuration, maintenance, and user training overhead.

– No source code availability (this is conditional).
– Insecurity: the user keystore passphrase is kept in file to avoid repeated user interaction in cases of failure (for recovery). A solution for this might be the usage of RFC3820 proxy X.509 certificates.

The Condor-UNICORE Bridge approach as a credential provisioning solution might be a proper one when there is already a Condor deployment which is used by users as primary means for submitting any kind of tasks to systems. The GAHP [19] provides a smooth integration interface for extending the interoperability of the Condor system with other systems. This does not offer any kind of limited and restricted authentication and authorization capabilities in contrast to the proxy certificate profile approach. The Condor-G project [20], incorporated more extensive support for job submission to Globus sites into Condor. It has also added the ability to make use of MyProxy [21] for credential management, although for proxy certificates to be used for authentication in UNICORE, the presented enhancement—or a functionally similar solution—has to be available.

## 7 Conclusion

UNICORE is a very mature grid software system, with a wide deployment base. With the ever-increasing grid site installations and different grid middleware systems, a concern of imperative importance is the interoperability between heterogeneous software environments. This paper presented a brief overview of the traditional UNICORE authentication and authorization model, as well as the main barriers that exist preventing the harmonic symbiosis and compatibility of the various grid systems. A simple solution was introduced, extending UNICORE to support authentication using a proxy certificate profile extension. The proposed solution provides a significant step towards the near-term interoperability goals, which are the main concern of the OMII-Europe project.

## Acknowledgement

This work is supported by the European Commission through the OMII-Europe project-INFSO-RI-031844. For further information please refer to [2].

## References

1. Housley, R., Polk, W., Ford, W., Solo, D.: Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile (2002), http://www.ietf.org/rfc/rfc3820.txt
2. Open Middleware Infrastructure Institute for Europe. Project no: RI031844–OMII-Europe, http://omii-europe.org
3. UNICORE: release 5, http://www.unicore.eu
4. Romberg, M.: The unicore architecture: Seamless access to distributed resources. In: Proceedings of the 8th IEEE International Symposium on High Performance Distributed Computing (HPDC-1999) (1999)

5. OpenSSL, http://www.openssl.org/
6. Java Secure Socket Extension (JSSE) Reference Guide for the Java 2 SDK, Standard Edition, v 1.4.2, http://java.sun.com/products/jsse/reference/docs
7. Castle, B.: Java crypto api, http://www.bouncycastle.org/
8. The Java Commodity Grid Kit (v1.4). http://www-unix.globus.org/cog/distribution/1.4/api/index.html
9. Farrell, S., Housley, R.: An internet attribute certificate profile for authorization, http://www.ietf.org/rfc/rfc3281.txt
10. Riedel, M., Mallmann, D.: Standardization processes of the unicore grid system. In: Proceedings of 1st Austrian Grid Symposium, pp. 191–203. Austrian Computer Society, Schloss Hagenberg, Austria (2005), http://www.fz-juelich.de/zam/vsgc/pub/riedel-2006-SPU.pdf
11. Anjomshoaa, A., Brisard, F., Drescher, M., Fellows, D., Ly, A., McGough, S., Pulsipher, D., Savva, A.: Job submission description language (jsdl) specification, version 1.0., http://www.ogf.org/documents/GFD.56.pdf
12. Russel, M., et al.: Transformation of JSDL into UNICORE AJOs. Poznan Supercomputing & Network Center. Personal Communication (March 2007)
13. Ciaschini, V.: A VOMS Attribute Certificate Profile for Authorization (2007), http://egee-jra-data.web.cern.ch/egee-jra1-data/glite-stable/stage/share/doc/voms/AC-RFC.pdf
14. Montes, J.A.M., Bernal, F.M., Sanchez, J.M.R.: The OpenPMI project: OpenSSL+AC, http://openpmi.sourceforge.net/
15. Levitte, R.: Official Support of Attribute Certificate Profiles in OpenSSL. Private communication (May 2007)
16. Authors, V.: Linux extended attributes and acls, http://acl.bestbits.at/
17. Snelling, D., van de Berge, S., Li, V.: Explicit trust delegation: Security for dynamic grids. FUJITSU Scientific and Technical Journal 40(2), 282–294 (2004)
18. Condor, High Throughput Computing Project, http://www.cs.wisc.edu/condor/
19. Nakada, H., Yamada, M., Itou, Y., Nakano, Y., Matsuoka, S., Frey, J.: Design and Implementation of Condor-UNICORE Bridge
20. The Condor-G Project, http://www.cs.wisc.edu/condor/condorg/
21. Basney, J., Humphrey, M., Welch, V.: The MyProxy Online Credential Repository. Software: Practice and Experience 35(9), 801–816 (2005)