PROCEEDINGS OF SCIENCE

Design and Implementation of Certificate Authority for High Performance Computing Infrastructure

Eisaku Sakane*, **Kento Aida and Kazutaka Motoyama**
National Institute of Informatics
E-mail: sakane@nii.ac.jp, aida@nii.ac.jp, motoyama@nii.ac.jp

The production-level operation of the High Performance Computing Infrastructure (HPCI) has been started from the end of September 2012. HPCI is a distributed supercomputing infrastructure in Japan. Currently, ten supercomputer sites, including the K computer, and two high performance shared storage sites organize HPCI. The National Institute of Informatics (NII) is in charge of operation of the authentication system, which enables single sign-on the supercomputers and the shared storages in HPCI using the grid security infrastructure (GSI). NII operates the certificate authority in the authentication system.

Recently, the Internet community strongly recommends using the SHA-2 family as hash algorithm for digital signature. In the grid community, the International Grid Trust Federation (IGTF) also has discussed SHA-2 migration, then requests the accredited certificate authorities to issue SHA-2 based certificates from the autumn of 2013. However, we do not have sufficient experiences based on practical experiments in nation-wide grid computing infrastructure such as HPCI.

In this paper, we present design and implementation of the certificate authority for HPCI (HPCI CA). HPCI CA follows the CP/CPS based on the Member Integrated X.509 Credential Services (MICS) profile, and it issues SHA-256 based certificates for CA itself and all end-entities in GSI. First, we present an overview of the authentication system in HPCI and the design of the certificate authority. Then, we discuss the issues for supporting SHA-2 based certificates in implementation of the authentication system focusing on two middleware: the software to build PKI domain (NAREGI-CA) and the GSI-enabled SSH client software (GSI-SSHTerm). NAREGI-CA is an open source software for easily building PKI domain. We mention the development of NAREGI-CA for handling the SHA-2 family. As for GSI-SSHTerm, the NCSA version that supports jGlobus 2 is adopted. Finally, we discuss the problems recognized after the start of production-level operation of HPCI. For instance, we illustrate how the problem that the cryptographic handshake between GSI-SSHTerm and GSI-enabled SSH server fails under a certain condition is resolved. We also briefly present a plan for supporting the Online Certificate Status Protocol (OCSP) responder and IPv6 capable end-points, which are used for publishing the Certificate Revocation List and the OCSP responder.

The International Symposium on Grids and Clouds (ISGC) 2013,
March 17-22, 2013
Academia Sinica, Taipei, Taiwan

*Speaker.

PoS(ISGC 2013)013

1. Introduction

Recently, with the use increase in distributed computing resources, authentication, authorization and secure communication mechanisms become more and more important. Among such mechanisms, the public-key infrastructure (PKI) is a major technology and supports secure communications in the Internet. In general, PKI basically consists of a certificate authority (CA), relying parties, and certificate users. One of the roles of CA is to issue certificates for end-entities (persons and network systems). A certificate is composed of a public key held by an end-entity, and a set of information following X.509 standard [1]. CA signs a certificate; that is, CA computes the message digest of the certificate with a hash function, e.g., MD5, SHA-1, and SHA-2, then encodes the message digest with CA's private key of a public-key cryptography like the RSA algorithm.

In X.509 certificates, a hash collision between different certificates is serious problem. The MD5 hash function is already compromised because collision attacks against MD5 exist and are feasible in realistic time. The SHA-1 hash function is a successor to MD5, however, it is thought that SHA-1 will be also compromised in the near future for the same reasons as MD5. The National Institute of Standards and Technology (NIST), US, recommended that SHA-1 would be deprecated in digital signature after 2011 in SP800-57 [2]. Namely, strong algorithms such as SHA-2 family (SHA-224, SHA-256, SHA-384 and SHA-512) should be used. In Japan, the CRYPTREC (Cryptography Research and Evaluation Committees) project [3] publicized the e-Government Recommended Ciphers List on February 2003, which is a list of ciphers that should be recommended for use in the procurement of "e-Government". In the list updated on March 2013, SHA-1 is restricted to use for backward compatibility and SHA-2 family is recommended.

The International Grid Trust Federation (IGTF) [4] establishes common policies and guidelines for international cooperation in grid computing. IGTF manages authentication profiles that provide the minimum requirements on X.509 PKI CAs. For example, the Authentication Profile for Classic X.509 Public Key Certification Authorities with secured infrastructure (Classic CA) describes the minimum requirements on traditional X.509 PKI CAs. The traditional X.509 PKI CA, actually the registration authority (RA), itself vets the identity of any entity by means of a face-to-face meeting. The vetting of identity is confirmed via photo-identification and/or similar valid official documents. Authentication profiles managed by IGTF prescribe for cryptography as certificate and CRL profiles. CRL stands for certification revocation list. The certificate and CRL profiles prescribe for handling X.509 information.

The High Performance Computing Infrastructure (HPCI) in Japan offers a distributed computing infrastructure. In order to easily access the computing resources constituting HPCI, a single sign-on system is needed. HPCI introduced GSI [10], [11] as a component technology for single sign-on. Since GSI is based on PKI, we have to discuss who or which organization issues what kind of certificates and confirm whether all applications used in HPCI are able to use the certificates.

Taking into account the compromise of ciphers in PKI, IGTF started discussion about SHA-2 migration from 2012 and made the plan to issue SHA-2 based certificates from August 2013. IGTF's discussion obviously focuses on CA related issues: how CA issues SHA-2 based certificates and when CA should start to issue SHA-2 based certificates. Although SHA-2 based certificates can be issued by CA, it would be meaningless if all applications used in the PKI domain cannot use the SHA-2 based certificates. However, there is no comprehensive discussion including whether

the applications used in the PKI domain can use SHA-2 based certificates.

In this paper, we present the design and implementation of HPCI CA, and also mention the authentication system of HPCI. First, we discuss how HPCI CA issues SHA-2 based certificates. Next, we investigate whether the GSI-enabled applications such as GSI-OpenSSH [12] can use SHA-2 based certificates on all platform that should be considered. Finally, we conduct the operation tests for the HPCI authentication system where SHA-2 based certificates are deployed, and show that the HPCI authentication system including not only the application server but also the client works properly. Moreover, this paper will contribute to the grid community by clarifying our knowledge obtained through the production-level operation of HPCI where SHA-2 based certificates are deployed.

The reminder of this paper is organized as follows. In Section 2 we present an overview of the authentication system in HPCI and discuss requirements and issues for HPCI CA and PKI participants. Section 3 presents the design and implementation of CA for HPCI. Section 4 evaluates a test environment deployed SHA-2 based certificates. Section 5 makes discussion and presents future issues. Finally, Section 6 concludes the paper.

2. Requirements and issues

This section presents an overview of the HPCI authentication system and discusses requirements and issues for HPCI CA and its PKI participants.

2.1 HPCI authentication system

The High Performance Computing Infrastructure (HPCI) in Japan is a distributed computing infrastructure composed of high performance computing resources located in Japan. HPCI started the production-ready operation from the end of September 2012. As of March 2013, ten supercomputer sites, including the K computer [6], and two high performance shared storage sites organize HPCI and the resources are connected with the SINET-4 network [7], the Japanese academic backbone network connecting more than 700 universities and research institutions. Figure 1 shows an overview of HPCI.

Since HPCI is a distributed computing infrastructure, it is necessary to offer a single sign-on system to HPCI users. HPCI adopts Shibboleth [8], [9] and Grid Security Infrastructure (GSI) [10], [11] as component technology to realize the single sign-on system. The Shibboleth federation system is used as the initial authentication in HPCI. Namely, HPCI imposes Shibboleth authentication on users when access to the HPCI portal, which offers the issuance service of a user certificate as a Shibboleth service provider (SP). HPCI account is an account provided by the Shibboleth identity provider (IdP). After the Shibboleth authentication, HPCI users can obtain their certificate via the HPCI portal. The user certificate is stored in the certificate repository managed by HPCI CA system. The user can download it into user's terminal. GSI offers the single sign-on system to access the supercomputers and shared storages. To access the supercomputers, the HPCI user issues the proxy certificate via the HPCI portal. According to the proxy certificate handling in our default setting, the HPCI user can store his/her proxy certificate in the proxy certificate repository or can download it into his/her terminal. Then, using GSI-enabled SSH such as GSI-OpenSSH [12], The HPCI user can log in the front-end of the supercomputer.

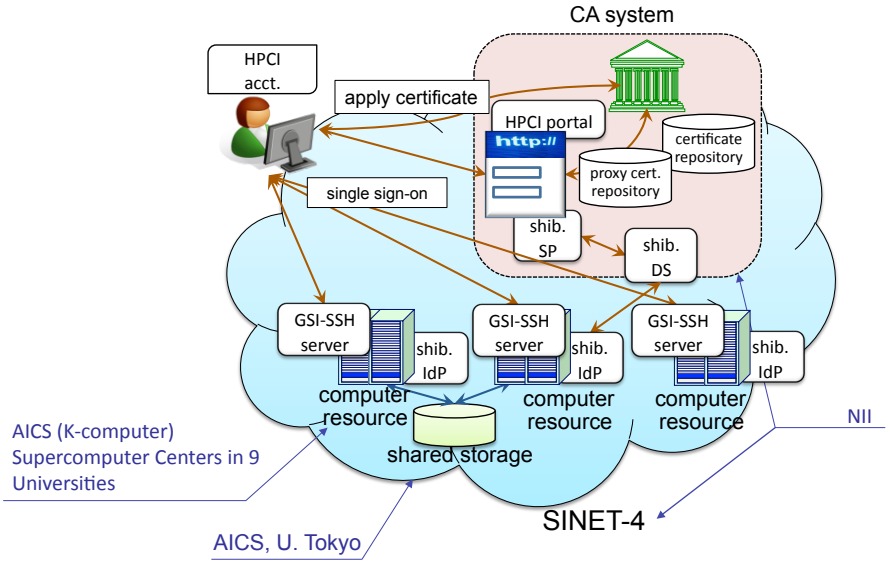

Figure 1: An HPCI overview.

2.2 Requirements and issues

As described in Section 2.1, HPCI uses GSI as a technology to realize the single sign-on to the computing resources. We have to discuss details of certificates deployed to HPCI. The requirement for HPCI authentication system with respect to PKI (GSI) is simple. The IGTF requirements mentioned in Section 1 should be considered because HPCI plans to interoperate with world-wide grid communities. We started to build HPCI in 2011 toward the production-ready operation in September 2012. Although the start of the production-ready operation of HPCI was about one year before the end of SHA-1 based certificate issuance planned by IGTF, we decided to build GSI in HPCI with SHA-2 based certificates at the start to avoid a burden of work to shift from the SHA-1 based system to the SHA-2 based system, because shifting the system within one year requires too much cost.

In order to use SHA-2 based certificates, the following issues should be addressed:

1. CA should be able to issue SHA-2 based certificates.
2. GSI-enabled SSH server should be able to use a SHA-2 based server certificate.
3. GSI-enabled SSH client should be able to use a SHA-2 based client certificate.

The first and second issues will be trivial. However, the third issue is not trivial because there are several platforms that should be supported for client, for example, Linux, OS X and MS Windows. It has been unknown whether or not GSI-enabled SSH clients using SHA-2 based certificate on such platforms work properly.

3. Design and implementation

This section presents the design and implementation of the certificate authority for HPCI

	UN*X	OS X	Windows
CA	NAREGI-CA	-	-
Server	GSI-OpenSSH	-	-
Client	GSI-OpenSSH	GSI-OpenSSH	GSI-SSHTerm

Table 1: Software for GSI in HPCI.

sha224WithRSAEncryption	id-dsa-with-sha224	ecdsa-with-SHA224
sha256WithRSAEncryption	id-dsa-with-sha256	ecdsa-with-SHA256
sha384WithRSAEncryption		ecdsa-with-SHA384
sha512WithRSAEncryption		ecdsa-with-SHA512

Table 2: SHA-2 based digital signature algorithms supported by AiCrypto.

(HPCI CA) and software packages for its PKI participants. Table 1 shows software packages on each platform.

3.1 CA

Since CA is the cardinal point of security, it is important how to operate CA. CA in production-level has a Certificate Policy and Certification Practice Statement (CP/CPS) [13]. We defined the CP/CPS of HPCI CA based on the Member Integrated X.509 PKI Credential Service (MICS) profile authorized by IGTF, because MICS seems fit for the operation of the HPCI authentication system. As mentioned in Section 2.1, an HPCI user is able to issue a certificate with his/her HPCI account. An HPCI account is provided by the supercomputer center, which operates the Shibboleth IdP. We can regard the supercomputer center as an organization having an identity management (IdM) system described in the MICS profile.

To solve the issue for CA, namely, to issue SHA-2 based certificates, we adopt NAREGI-CA software for implementation of HPCI CA. The NAREGI-CA [14] software is an open source software package to easily build a PKI in grid computing. This was developed by the National Institute of Informatics (NII) in Japan as a component of the NAREGI Grid Middleware [15]. NAREGI stands for National Research Grid Initiative. The NAREGI-CA includes a cryptographic library, called AiCrypto, CA/RA daemon programs, command-line utilities and web user interfaces for users and RA administrators. AiCrypto provides many kinds of cryptography used in PKI, concretely, DES, 3DES, RC2 and AES for block ciphers, RSA, DSA and ECC for public-key ciphers, MD2, MD5, SHA-1 and HMAC for hash functions, and PKCS libraries (#8, #10, #11 and #12). AiCrypto supports SHA-2 family and digital signature algorithms using SHA-2 listed in Table 2, and provides a set of APIs to daemon programs and command-line utilities. Moreover, NAREGI-CA defines a protocol prescribing on-line interactions between PKI components, called the lightweight certificate management protocol (LCMP). LCMP is designed based on CMP [16].

The CA system in the HPCI authentication system is composed of IA (issuing authority), the RA servers, the certificate management system and the portal. Figure 2 shows the architecture of the CA system. IA and RA daemon programs, aicad and airad, run on the IA server and the RA server respectively. The private key of CA is stored in a hardware security module (HSM). The aicad creates a certificate and publishes a CRL. The airad receives CSR from the applicant, check

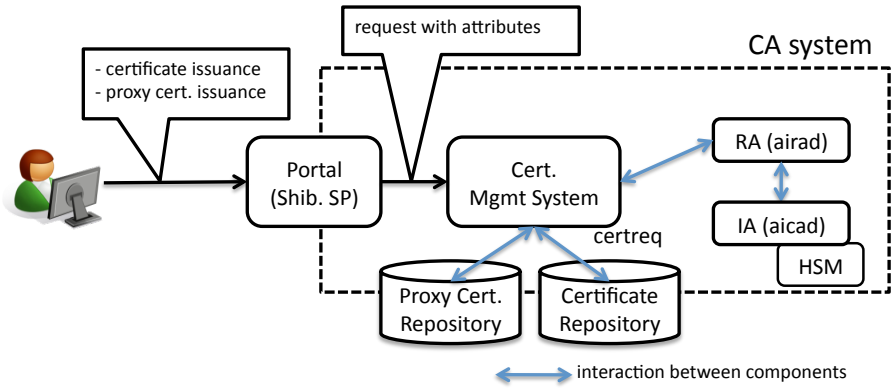

Figure 2: Architecture of the CA system.

the CSR, then send it to the aicad. A command-line utility, certreq, creates CSR and sends it to the airad. Communications between aicad, airad and certreq are based on LCMP. The certificate repository and the proxy certificate repository are implemented using MyProxy [17]. The certificate management system is a special product for HPCI. This system manages each user certificate, handles a set of user information to issue a certificate and issues a command to the RA server and the repositories according to user's request via the HPCI portal. Let us consider a user certificate issuance at the first time. An HPCI user make a request to the HPCI portal for certificate issuance after Shibboleth authentication. If the user succeeds in the Shibboleth authentication, the HPCI portal will send the request with some attributes used to set to X.509 information to the certificate management system. Receiving the request with the attributes, the certificate management system executes the certreq command to obtain the user's certificate. CSR created by the certreq is sent to the IA server via the RA server, then the aicad signs the CSR and sends back to the RA server. The RA server sends the certificate to the certificate management system. Finally, the certificate is stored in the certificate repository in our default setting.

3.2 Server

In order to build a GSI-enabled SSH server, we use GSI-OpenSSH [12]. GSI-OpenSSH is a modified version of OpenSSH that adds support for the GSI authentication and credential forwarding, providing a single sign-on and file transfer service. The GSI implementation of Globus Toolkit [18] uses OpenSSL [19] to handle certificates. Since recent OpenSSL supports SHA-2, the server is able to use a SHA-2 based certificate.

3.3 Client

There are at least three platforms that should be supported for GSI-enabled SSH client, namely, UN*X, Mac OS X and MS Windows. Among such platforms, many HPCI users make use of MS Windows. As shown in Table 1, HPCI recommends the GSI-SSHTerm as a GSI-enabled SSH client for MS Windows. Since Mac OS X is a UNIX OS, HPCI recommends the GSI-OpenSSH as a client for UN*X.

The GSI-SSHTerm [20] is a GSI-enabled terminal emulator for multi-platform and an open source Java application maintained by the UK's National Grid Service (NGS). One of the advan-

tages of the GSI-SSHTerm is that its installation is quite easy. So, HPCI adopts the GSI-SSHTerm. However, the official latest release of the GSI-SSHTerm, version 0.91h, does not support SHA-2 based certificates. The GSI-SSHTerm uses a Java implementation for GSI, jGlobus 1.8, which does not support SHA-2 family. The jGlobus 1.8 uses the PureTLS [21] to verify certificates. However, the PureTLS is no longer actively maintained and does not support recent Java such as the Oracle Java Standard Edition 6. Thus, we came to a conclusion that it is difficult to improve the GSI-SSHTerm, that is the PureTLS, in order to be able to handle SHA-2 based certificate.

We decided to use the other development branch, jglobus2_branch (0.91i-ncsa). This branch was committed by the National Center for Supercomputing Applications (NCSA), US, and is based on the jGlobus 2 [22] that supports SHA-2 family. Therefore, an improvement in the GSI-SSHTerm 0.91i-ncsa for handling SHA-2 based certificates is unnecessary. Moreover, we added the improvements of the GSI-SSHTerm 0.91i-ncsa as follows:

- "null" as host-key algorithm in key exchange algorithm negotiation.

  The GSI-enabled ssh daemon provided by GSI-OpenSSH, gsisshd, selects a host-key algorithm in key exchange algorithm negotiation with the client. When no SSH host keys such as ssh_host_rsa_key is provided by the server, gsisshd selects only "null" algorithm. The original GSI-SSHTerm does not understand "null" algorithm; thus, we enabled the GSI-SSHTerm to understand "null" algorithm.

- more consistent drawing process presenting the progress of file-transfer in SFTP session.

  The graphical user interface (GUI) of the GSI-SSHTerm is implemented by using Swing, which is the primary Java GUI widget toolkit. In addition, the GSI-SSHTerm has an exclusive Thread to file-transfer, which manages a progress bar window in the file-transfer. Since the file-transfer Thread runs independently of the Thread of Swing, both Threads often conflicted. We improved the GSI-SSHTerm so that the drawing process in the file-transfer runs in cooperation with the Swing's Thread.

4. Operation Tests

Before the start of the production-ready operation of HPCI, we conducted operation tests for HPCI authentication system where SHA-2 based certificates were deployed.

For the CA system, the IA and RA servers run on Red Hat Enterprise Linux 6.1. The IA server is 32-bit operating system (OS), because the device driver for the HSM, which we adopted the SafeNet's Luna G5 [23], did not support 64-bit Linux OS. We created the RSA key-pair of HPCI CA and stored the private key in the HSM by using NAREGI-CA 2.4. The public key of HPCI CA had 2048 bit length, and was signed with sha256WithRSAEncryption by NAREGI-CA.

All end-entity certificates used RSA as public-key cryptography, had 2048 bit public-key length, and were also signed with sha256WithRSAEncryption by NAREGI-CA. For user certificates, the CSRs were created by only NAREGI-CA as mentioned in Section 3.1. For server certificates, the CSRs were created by NAREGI-CA or OpenSSL. For OpenSSL, for example, an applicant executed interactively the following command:

OS	Java
OS X 10.6.8	Oracle Java 1.6.0_35
OS X 10.7.5	Oracle Java 1.6.0_35
OS X 10.8.2	Oracle Java 1.6.0_35
openSUSE 12.2	OpenJDK Java 1.7.0
Red Hat Enterprise Linux 6 (32bit)	Oracle Java 1.6.0_38
Ubuntu 12.04 LTS	OpenJDK Java 1.6.0_24
Windows XP SP3 (32bit)	Oracle Java 1.6.0_35
Windows Vista SP2 (32bit/64bit)	Oracle Java 1.7.0_07
Windows 7 SP1 (32bit/64bit)	Oracle Java 1.7.0_07

Table 3: Platforms on which the modified version of GSI-SSHTerm works.

```
$ openssl genrsa -des3 -out hostkey.pem 2048
$ openssl req -new -key hostkey.pem -out hostreq.pem \
> -config ./openssl.cnf -reqexts v3_req
```

where the configuration file, openssl.cnf, was used to set the subject alternative name extension. For NAREGI-CA,

```
$ certreq csr -key hostkey.pem -req hostreq.pem -size 2048 \
> -s -g "SSL server" -cn gsissh.example.org -em . -alt-dns
```

where the option, -alt-dns, enables the certreq to use the common name (CN), gsissh.example.org, as DNS name in the subject alternative name extension.

Most of the GSI-enabled SSH server were built on the Red Hat compatible Linux distribution, like CentOS 5.6 or later. We used the GSI-OpenSSH included in Globus Toolkit 5.0.4. Since the GSI implementation of Globus Toolkit uses the OpenSSL library, it is necessary for the GSI-enabled server to have the OpenSSL library supporting the SHA-2 family. There is no problem in above Linux distributions. However, for example, CentOS 4.9 is not suitable OS for the GSI-enabled SSH server because CentOS 4.9 has OpenSSL 0.9.7a that does not support the SHA-2 family. In our case, the OS of the GSI-enabled SSH server at a certain supercomputer center was Red Hat Enterprise Linux 4.9 with which CentOS 4.9 is compatible. It was impossible to update the OS to suitable OS because of the circumstances of the supercomputer center. To solve the problem, we installed the OpenSSL suitable for GSI, e.g., version 0.9.8, separately from OpenSSL 0.9.7a. Two OpenSSL libraries should coexist in the system because the OpenSSL library is a core library for the system.

For the GSI-enabled SSH client on UN*X platforms, we used the same GSI-OpenSSH as the server. For the HPCI users, the client software for Mac OS X or MS Windows is more important. For MS Windows platform, we used the modified software based on the GSI-SSHTerm 0.91i-ncsa. Table 3 shows combinations of OS and Java implementation at our operation tests, on which the modified version of the GSI-SSHTerm works.

We confirmed that SHA-2 based certificates work well in our test environment.

5. Discussion and future issues

In this section we mention further GSI-SSHTerm improvement that solves a problem recognized after the start of the production-ready operation of HPCI, and discuss future issues.

5.1 GSI-SSHTerm improvement

After the start of the production-ready operation of HPCI, we have encountered a troublesome happening in using the modified version of GSI-SSHTerm. It is that a client fails to connect a certain server due to an error that occurred in the connecting or in authenticating. The client can rarely log in the server. The error message of the GSI-SSHTerm at the failure shows that "Expecting a non-zero length token from GSS_Init_sec_context". This problem happened when we use GSI-SSHTerms to connect GSI-SSH servers in question. However the problem did not happen when we use GSI-OpenSSH as client. The essential causes of the problem has not been specified yet. It may be a certain network problem between the client and server rather than GSI-SSHTerm problem. Since it was hard to pinpoint the cause of the problem, we further improved the GSI-SSHTerm to avoid the problem.

The GSI-SSHTerm returns the error at the following:

method: performClientExchange()
class: com.sshtools.j2ssh.transport.kex.GssGroup1Sha1

Investigating the source code of GSI-SSHTerm, we found that the method, gsscontext.initSecContext(), returned "null", so that the handshake token from GSS_Init_sec_context would become zero. Further investigating why the token was zero, it was found that the handshake token could not be read in the method, nextHandshakeToken(), of the class, edu.illinois.ncsa.BCGSS.TlsHandlerUtil, because of unspecified reasons.

The class, TlsHandlerUtil is defined in ncsa-lcrypto.jar. The source code of ncsa-lcrypto.jar is available at the git repository of BouncyCastleSSLv3 [24]. Investigating the source code of ncsa-lcrypto.jar, we found that the method, readData(), provided by the class, RecordStream, breaks off reading halfway the handshake token from the server under unspecified situations. Then, we improved the readData() to preserve the negotiation of the handshake token. It was confirmed that our improvement solved the problem.

5.2 CA accreditation by IGTF

The IGTF requirements refers to not only SHA-2 based certificates but also the following topics:

- Accessibility of CRL distribution point over IPv6.
- Support for OCSP responder.

OCSP [25] stands for Online Certificate Status Protocol.

HPCI CA aims at accreditation by IGTF for interoperations with world-wide grid communities. Thus, HPCI CA should satisfy the remainder of the requirements. Currently, HPCI CA is not ready for the requirements, however, there is no technical problem because the NAREGI CA,

which is an accredited CA by IGTF in Japan and operated by NII, already has knowledge of OCSP responder by OpenCA [26] through pilot operation for the OCSP. We plan to hand the role of the NAREGI CA over the HPCI CA in the near future.

6. Summary

In this paper, we presents a design and implementation of certificate authority for HPCI authentication system that uses GSI as component technology. Our contribution in this paper is as follows:

- We proposed a GSI-enabled software environment deployed SHA-2 based certificates, which covers not only CA but also the GSI-enabled SSH server and the client.
- We discussed the improvements of GSI-enabled SSH client.

We believe that our improvement of GSI-SSHTerm would be more useful for a grid computing infrastructure that encounters the happening mentioned in Section 5.1, because it would be difficult to analyze the essential cause of the problem due to network complexity.

Acknowledgments

This work is partially supported by the Ministry of Education, Sports, Science and Technology in Japan.

References

[1] ITU-T Recommendation X.509, http://www.itu.int/rec/T-REC-X.509/en

[2] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid, *Recommendation for key Management – Part 1: General (Revision 3)*, http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57_part1_rev3_general.pdf

[3] Cryptography Research and Evaluation Committees in Japan, http://www.cryptrec.go.jp/english/

[4] International Grid Trust Federation, http://www.igtf.net/

[5] High Performance Computing Infrastructure Portal Site, https://www.hpci-office.jp/

[6] K computer – RIKEN Advanced Institute for Computational Science, http://www.aics.riken.jp/en/

[7] Science Information NETwork 4, http://www.sinet.ad.jp/index_en.html?lang=english

[8] R. L. Morgan, S. Cantor, S. Carmody, W. Hoehn, and K. Klingenstein, *Federated Security: The Shibboleth Approach*, *EDUCAUSE Quarterly*, **27**, (2004).

[9] Shibboleth, http://shibboleth.net

[10] V. Welch, F. Siebenlist, I. Foster, J. Bresnahan, K. Czajkowski, J. Gawor, C. Kesselman, S. Meder, L. Pearlman, and S. Tuecke, *Security for Grid Services*, in proceedings of *the 12th IEEE International Symposium on High Performance Distributed Computing*, (2003).

[11] V. Welch, *Globus Toolkit Version 4 Grid Security Infrastructure: A Standard Perspective*, http://www.globus.org/toolkit/docs/4.0/security/GT4-GSI-Overview.pdf

[12] GSI-OpenSSH, http://grid.ncsa.illinois.edu/ssh/

[13] S. Chokhani, W. Ford, R. Sabett, C. Merrill and S. Wu, *Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework*, http://tools.ietf.org/html/rfc3647

[14] NAREGI-CA development, http://ca-dev.naregi.org/

[15] NAREGI project, http://www.naregi.org/

[16] C. Adams and S. Farrell, *Internet X.509 Public Key Infrastructure (PKI) Certificate Management Protocols*, http://tools.ietf.org/html/rfc2510

[17] MyProxy Credential Management Service, http://grid.ncsa.illinois.edu/myproxy/

[18] Globus Toolkit, http://www.globus.org/toolkit/

[19] OpenSSL: The Open Source toolkit for SSL/TLS, http://www.openssl.org/

[20] GSI-SSHTerm Application, http://www.ngs.ac.uk/use/tools/gsisshterm

[21] Claymore PureTLS, http://www.rtfm.com/puretls/

[22] jGlobus, https://github.com/jglobus

[23] Safenet Luna G5, High Assurance Hardware Security Module with a USB interface, http://www.safenet-inc.com/products/data-protection/hardware-security-modules/luna-g5/

[24] SSLv3 support for Bouncy Castle's TLS implementation, https://github.com/jsiwek/BouncyCastleSSLv3

[25] M. Myers, R. Ankney, A. Malpani, S. Galperin and C. Adams, *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP*, http://tools.ietf.org/html/rfc2560

[26] OpenCA Research Labs. http://www.openca.org/