A National-Scale Authentication Infrastructure

The authors' Grid Security Infrastructure lets users access resources at any participating site without repeated authentication, while preserving a site's ability to use site-specific security mechanisms and enforce local access control.

Randy Butler
Von Welch
National Center for Supercomputing Applications

Douglas Engert
Ian Foster
Steven Tuecke
John Volmer
Argonne National Laboratory

Carl Kesselman
USC Information Sciences Institute

Today, individuals and institutions in science and industry are increasingly forming virtual organizations to pool resources and tackle a common goal. In one example, the National Science Foundation's Partnerships for Advanced Computational Infrastructure program provide a next-generation infrastructure for computational science. PACIs, relatively large and long-lived virtual organizations funded for five to 10 years, link some 50 institutions and thousands of researchers. Other virtual organizations, however, may be smaller and more fleeting.

Participants in virtual organizations commonly need to share resources such as data archives, computer cycles, and networks—resources usually available only with restrictions based on the requested resource's nature and the user's identity. Thus, any sharing mechanism must have the ability to authenticate the user's identity and determine whether the user is authorized to request the resource. Virtual organizations tend to be fluid, however, so authentication mechanisms must be flexible and lightweight, allowing administrators to quickly establish and change resource-sharing arrangements. Nevertheless, because virtual organizations complement rather than replace existing institutions, sharing mechanisms cannot change local policies and must allow individual institutions to maintain control over their own resources.

Our group has created and deployed an authentication and authorization infrastructure that meets these requirements: the Grid Security Infrastructure.[1] GSI offers secure single sign-ons and preserves site control over access policies and local security. It provides its own versions of common applications, such as FTP and remote login, and a programming interface for creating secure applications. Dozens of supercomputers and storage systems already use GSI, a level of acceptance reached by few other security infrastructures.

MULTISITE AUTHENTICATION

Virtual organizations must have a reliable means for identifying requestors, but participant independence complicates authentication across multiple sites. PACIs provide a good example of both the issues and the technical requirements for a multisite authentication infrastructure.

The PACI community

The NSF PACIs—two consortia of some 50 universities and government laboratories—are dedicated to developing next-generation scientific problem-solving tools. Virtual organizations in their own right, the PACIs independently provide resources to an even larger and less-formal national user community of many thousands of researchers, educators, and students. The various subsets of the PACI community interact in different ways, whether to develop a next-generation software system, operate a remote electron microscope, or access a supercomputer. Most of these interactions involve some form of authentication and authorization.

PACI institutions all have long histories of running computing facilities, and each has well-established policies and procedures for various aspects of facility operation, including computer security and, in particular, authentication and authorization. Many member sites run standard Unix authentication using DES-encrypted passwords, several run various flavors of Kerberos and the distributed computing environment (DCE), and a few use one-time password mechanisms.

In the PACIs' early days, efforts were made to convince a core set to run Kerberos, with the vague hope of eventually establishing it as the predominant cross-realm authentication tool. However, this approach quickly proved infeasible for various technical, financial, and political reasons. Clearly, member institutions will continue to use both Kerberos and

0018-9162/00/$10.00 © 2000 IEEE

non-Kerberos solutions within their sites for the foreseeable future. The resource-sharing mechanisms the PACI community uses must be able to coexist with these different local mechanisms.

Technical requirements

Without an integrated authentication and authorization solution, virtual organizations have used a variety of ad hoc schemes to achieve resource sharing, such as giving users an account at each institution with distinct login names and passwords. Some put information on Web sites with access controlled via other passwords. This multiplicity of mechanisms and passwords makes access difficult, discouraging information sharing and collaboration. It also hinders the creation of software that securely spans resources at multiple institutions or that allows secure collaboration between users at multiple institutions.

Users. For users, the primary requirement is simplicity: Access to the virtual organization's resources should not be significantly different from access to the local organization's resources. There should be a single sign-on, where users need to log on only once to access all permitted resources. Programs running on a user's behalf should possess a subset of the user's rights and have access to the permitted resources.[2]

The solution, then, must transparently interface with common remote access tools: remote login via Telnet and rlogin, file access via FTP, Web browsers, and programming libraries such as CORBA and MPI. It must also allow implementation of new intersite applications by providing standardized APIs for accessing security functions. For example, a group developing collaborative design tools should be able to easily integrate authentication and authorization mechanisms.

Sites. The concerns of resource-providing sites constrain an authentication and authorization infrastructure in two ways:

- Sites typically cannot easily replace or modify their intradomain security solution, so we need a distinct interdomain solution that interoperates with local security solutions, is at least as strong as local solutions so that it does not weaken site security, and is easy to understand so that site administrators can trust it.
- Site administrators must have tight control over policies governing access to their resources, including how users establish their identity and which users have access to which resources.

The "Technical Alternatives for Multisite Authentication" sidebar explains why the two most popular authentication approaches —Kerberos[3] and secure shell—did not meet these requirements, prompting us to develop GSI.

Technical Alternatives for Multisite Authentication

Two widely used authentication approaches—Kerberos and secure shell—do not meet our requirements.

Kerberos

Kerberos—used alone or under the distributed computing environment—authenticates users through a secure transaction with a centrally maintained key server. Kerberos achieves interorganizational, or cross-realm, authentication by designating trustworthy key servers in other organizations. Kerberos meets many of the basic requirements for virtual organization authentication, but it presents two problems:

- Using Kerberos for intersite authentication also means using it for intrasite authentication, which is often not feasible because of equipment and staffing costs.
- Sites must negotiate many cross-realm authentication agreements, and many sites resist surrendering too much control over local policy.

Secure shell

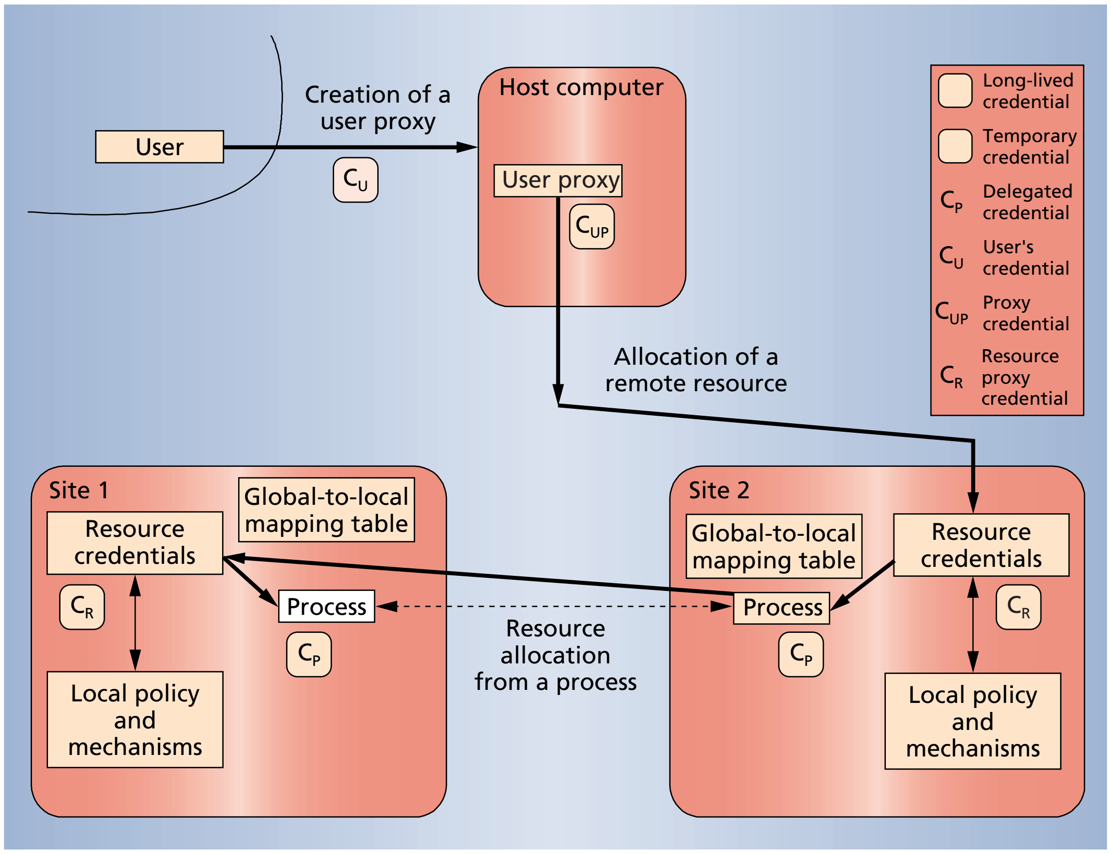
Secure shell (SSH), a widely used login technology, meets a number of our requirements: It is based on public-key authentication technology, uses link encryption to protect user credentials, and is easily deployed. Users like SSH because it provides basic remote login and file copy capabilities without a lot of complexity. SSH, however, has two significant drawbacks:

- It requires users to manage their own cross-site authentication relationships by copying public keys (or keeping track of passwords) for all sites they access, a task that can be burdensome if they access many sites. Moreover, SSH does not give sites control over authorization, so they cannot, for example, deny access to a particular user without invading user privacy.
- SSH supports only limited capabilities—remote shell and file transfer—but not others that require authentication, such as collaborative environments and Web browsers.

GRID SECURITY INFRASTRUCTURE

GSI is an alternative approach to intersite security. We began developing it under the Globus research project[4] to support distributed computing environments, or Computational Grids,[5] which are similar to virtual organizations. GSI deals with interdomain operations, bridging the different local security solutions of constituent sites. As Figure 1 shows, GSI includes several significant features:

- Credentials, using standard X.509v3 certificates as the private keys, represent the identity of each entity—user, resource, program—specifying the entity's name and additional information, such as a public key. A certification authority (CA)—a trusted third party—ties an identity to a public-private key pair by signing a certificate.
- An authentication algorithm, defined by the



Creation of a user proxy
User
C_U
Host computer
User proxy
C_{UP}
Allocation of a remote resource
Long-lived credential
Temporary credential
C_P Delegated credential
C_U User's credential
C_{UP} Proxy credential
C_R Resource proxy credential
Site 1
Resource credentials
Global-to-local mapping table
C_R
Process
C_P
Local policy and mechanisms
Resource allocation from a process
Site 2
Global-to-local mapping table
Resource credentials
Process
C_P
C_R
Local policy and mechanisms

Figure 1. Schematic showing the basic operations that GSI supports. Following the dark line from the top left-hand corner, we first see user authentication via public-key mechanisms applied to the user's credential (C_U), followed by creation of a temporary user proxy credential (C_{UP}), then subsequent requests to remote resources, represented by resource proxies holding resource proxy credentials (C_R), and finally authorization and global-to-local identity mapping at an individual site, resulting in the creation of a remote process at Site 2, with its own delegated credential (C_P). We also see how such a remote process can use its delegated credential to initiate further requests to other sites (in this case, a process creation request to Site 1) and engage in authenticated interprocess communication (the dashed line).

Secure Socket Layer Version 3 (SSLv3) protocol, checks the entity's identity. The veracity of an entity's identity is only as good as the trust placed in the CA that issued the certificate, so the local administrator installs these certificates, which are then used to verify the certificate chains.

- An entity can delegate a subset of its rights—such as a process a program creates—to a third party by creating a temporary identity called a proxy. Proxy certificates can form a chain, beginning with the CA and growing, as first the user, then the user's proxies, sign certificates. By checking the certificate chain, processes started on separate sites by the same user can authenticate to one another by tracing back along the certificate chain to find the original user certificate.
- Each resource can specify its policy for determining whether to accept incoming requests. The initial GSI used a simple access control list, but the current version uses other techniques.
- The authentication protocol verifies the global identity of involved parties, but GSI must convert this name to a local subject name—such as a login name or Kerberos principal—before the local security system can use the name. GSI does this by consulting a simple text-based map file—under the local site's control—that defines the binding between global and local names.
- The standard interface GSS-API provides access to security operations.[6] GSI uses OpenSSL or SSLeay, the free implementation of SSLv3, for its authentication protocols and support for proxy certificates. SSLv3 is used widely for Web security, has been well scrutinized for security problems, and has broad acceptance as a mature protocol.

Even though relatively simple, our architecture meets all user and system requirements we consider vital:

- For users, the global name and proxy credentials mean the user needs only one authentication to access all resources, and proxy credentials and delegation allow programs running on a user's behalf to access resources. The use of X.509, SSLv3, and GSS-API standards facilitates the development of common GSI-enabled tools and more complex applications.
- For sites, the architecture does not require changing the local security infrastructure; instead, sites can simply install relatively simple GSI-enabled servers that use well-known standards. Sites control policy through the access control list and map file, so administrators feel comfortable with the code and are willing to deploy it alongside SSH and other remote access mechanisms.

GSI EXTENSIONS

As part of the Globus Toolkit, GSI now runs on more than 80 sites. To deploy GSI, we needed to develop several extensions that address production facilities' operation concerns. The most significant of these extensions support multiple certification, interface with the local Kerberos environment, support smart cards, and facilitate Web-based computing.

Multiple certification authorities

In our initial implementation, we assumed that all user credentials would be associated with the Globus project's single CA, but in practice users must be able to present credentials obtained from any source: their site's CA, a CA associated with a virtual organization such as a PACI, or a commercial CA. Thus, sites must be able to handle credentials verified by different CAs; at the same time, as part of their access control policy, they must retain control over what CAs they can trust and what they can trust these CAs to do.

Web browsers, for example, often provide control over which CAs they trust—often by maintaining a list of trusted CA certificates—but they do not control what they can trust these CAs to do. In practice, however, we may want to trust only some certificates that a given CA signs. For example, a site providing educational materials might want to trust a high school's CA but not trust its certificates for nonstudent users.

To address these issues, we extended GSI with a general-access control function specified using the Generic Authorization and Access Control API.[7] This function lets an application reject an authentication operation based on the subject name and chain of certificate signers rather than simply on the issuing CA's identity. Users, then, can present credentials from any CA, and the site can decide whether to accept them.

Because of policy differences between sites regarding acceptable CAs, users may sometimes need to provide multiple credentials to access resources at multiple sites—an issue we are still investigating. With single sign-on, users may need different credentials at different times, which can require the complicated delegation of multiple credentials. When accessing a particular resource, a user with multiple credentials must determine which to supply—a process better handled by a protocol than by hit-or-miss guessing. Further, we need to enhance the protocol used to authenticate processes running on a user's behalf because each process may have its own credentials.

With the GSI deployment, both PACIs have created a CA for sites that don't want to run their own. Creating these CAs was a significant effort because of the need to define a certificate policy acceptable to all participating institutions. In particular, the National Computational Science Alliance, based in Urbana, Illinois, put considerable effort into developing such a policy, building on the Federal Public Key Infrastructure Project's model certificate policy.[8]

To deploy GSI, we developed extensions that support multiple certification, interface with the local Kerberos environment, support smart cards, and facilitate Web-based computing.

Obtaining Kerberos credentials

To interface with a local security environment in the simplest case, GSI only needs to map from the global name to a local subject name based on a map file entry. Many computing sites, however, use Kerberos intrasite security, so GSI must also obtain a set of local credentials in the form of Kerberos tickets. To obtain these credentials, GSI must be able to authenticate to a Kerberos realm, a DCE cell, or an Andrew file system cell, and Kerberos must be able to issue tickets based on GSI authentication, including proxies. To facilitate this, we developed SSLK5D, a modified Kerberos key distribution center, which returns a ticket that can be used like any Kerberos forwarded ticket. By controlling SSLK5D and its database for mapping certificate names to Kerberos principals, the security administrator retains control of the Kerberos realm or DCE cell.

SSLK5D functionally resembles the IETF's PK_INIT draft standards and DCE RFC 68.4, with one distinction: We use SSL rather than a specialized protocol to communicate to the SSLK5D server. However, as PK_INIT and RFC 68.4 become part of the standard environment, they can replace SSLK5.

As Figure 2 shows, this support for the creation of Kerberos credentials means that GSI can interoperate with, rather than replace, DCE.

Alternative credential management

Like many other public-key systems, GSI maintains the user's private key in encrypted form in the user's local file system. During sign on, the user provides a pass phrase to decrypt the private key—an approach

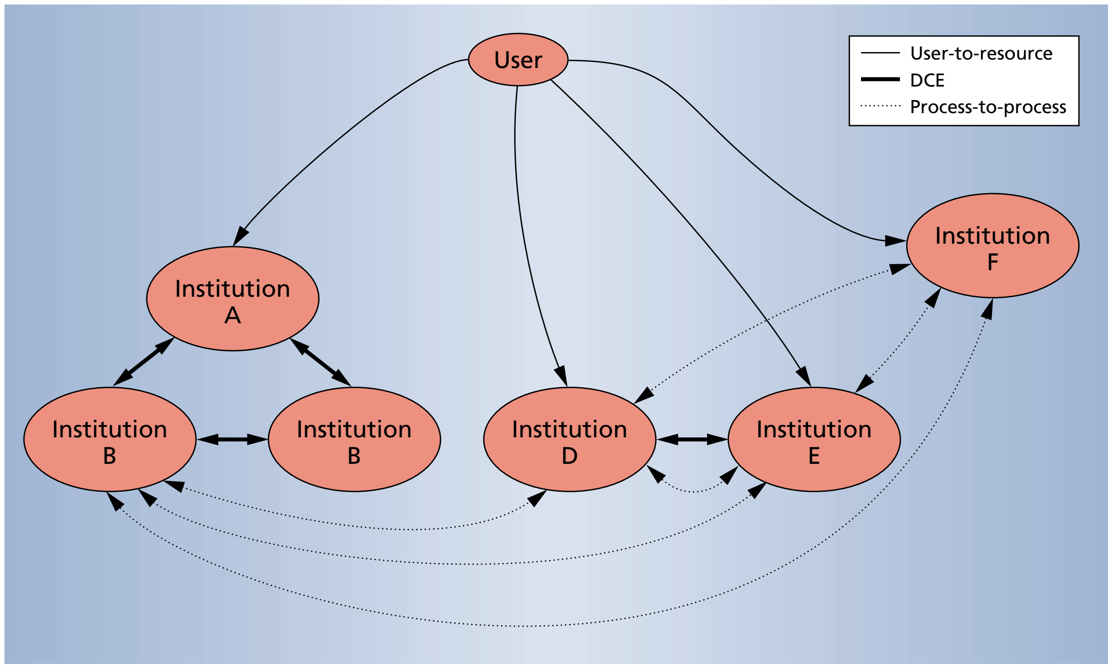


Figure 2. GSI used in an environment where some sites—A, B, and C, and D and E—use the distributed computing environment for intersite authentication. GSI allows a user to access resources in multiple "DCE clouds," while also allowing DCE use between sites when it is available.

that has three disadvantages:

- The private key may not be accessible when the user travels to a remote location.
- A careless user may expose the pass phrase when authenticating from a networked terminal.
- An adversary may retrieve the encrypted private key, then subject it to a pass-phrase-guessing attack.

These concerns prompted us to extend GSI so that it allows storage of the user's private key on a smart card, a credit-card-sized device that contains 4 to 16 Kbytes of memory and a microprocessor that can perform 1,024-bit cryptographic signing operations. The user's private key never leaves the card: During logon, the proxy generation code negotiates with the card to obtain a signed proxy certificate using the PKCS #11 protocol.

MyProxy credential server

One benefit of basing GSI on X.509 certificates and SSLv3 is that many existing software products can easily take advantage of it. Without modification, common Web browsers can use GSI certificates for authentication. Browsers, however, cannot perform delegation, a limitation for portals and other forms of Web-based computing.

To work around this limitation, we developed MyProxy, a trusted server and set of client applications. A user can delegate a proxy to the MyProxy server along with a tag and pass phrase. A client program can then connect to the MyProxy server later, present the tag and pass phrase, and receive a proxy for that user.

In Web-based environments, the user delegates a proxy to the MyProxy server, then connects to a Web-based portal with any off-the-shelf browser to provide the tag and pass phrase to the portal. The portal contacts the MyProxy server, requests a proxy for that user, then acts on the user's behalf.

We plan to enhance MyProxy so that it can manage long-term credentials—private keys and certificates—for users. Many users find managing their credentials uninteresting and difficult, and security administrators are concerned that users may fail to protect their long-term private keys. By managing these, MyProxy could ease the burden for users and help maintain security.

We are also investigating using MyProxy as a "credential wallet," in which users could deposit all their credentials for multiple sites. When requests for credentials come in, the MyProxy server could intelligently select which credential to delegate based on the requester's identity.

BUILDING THE INFRASTRUCTURE

Besides the augmented GSI, multisite authentication needs various other elements, including a set of GSI-enabled applications and various GSI-enabled toolkits.

GSI-enabled applications

True single sign-on means that PACI users should be able to use their GSI credentials for all PACI authentications and for more sophisticated applications such as distributed supercomputing and collaborative data analysis. To accomplish this goal, we developed GSI-enabled versions of several common applications and verified that commercial Web browsers can accept our credentials.

To GSI-enable SSH, we modified it to use GSS-API as one of its authentication mechanisms. Van Dyke Technologies also implemented these modifications in their commercial SSH product, SecureCRT. Users can present their GSI credentials for authentication to a GSI-enabled sshd daemon, including doing delegation. Because SSH supports multiple mechanisms—Kerberos, RSA key pairs, and passwords—it can support GSI and still be fully backward compatible. We have also used GSS-API to develop GSI-enabled FTP clients and servers, including the widely used wu-ftpd server, ncftp client, Unitree ftpd server, and HPSS pftpd server.

GSI-based toolkits

While these GSI-enabled applications provide a good basis for remote resource access, virtual organizations such as the PACIs also want security incorporated into a wide variety of other applications. We've therefore developed various tools for incorporating GSI mechanisms into applications.

One tool, gss_assist, provides a set of convenience functions for accessing GSS functions. GSS-API is rich and robust but also complex, and many applications require only a subset of GSS-API features. Gss_assist shields application developers from unneeded complexity, providing a simpler API that still implements full GSI security. Many gss_assist functions are simple wrappers around their GSS-API counterparts, with appropriate default values. Some are more complicated, such as the wrappers around the security context initialization and acceptance functions that perform the full looping and network communication users need for GSS-API authentication and context establishment.

The Globus Toolkit is a GSI-based application that provides a set of services for constructing distributed applications. Because the Globus Toolkit uses GSI functionality, any application that uses its mechanisms gets security essentially for free. For example, MPICH-G2, an extended version of the popular Message Passing Interface standard, uses Globus Toolkit mechanisms for initiating remote computation and hence does not need to do anything special to address authentication and authorization issues when running across multiple sites.

GSI's basic techniques are applicable in many different contexts. For example, we are currently employing this technology to create smaller testbeds for individual scientific collaborations and are investigating its feasibility for large high-energy physics projects. We are enhancing GSI to reduce the cost of establishing a virtual organization security environment, to add support for advanced features such as smart cards, and to restrict delegation for more fine-grained access control. ✷

Acknowledgments

This work was supported in part by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, US Department of Energy, under Contract W-31-109-Eng-38; by the Defense Advanced Research Projects Agency under contract N66001-96-C-8523; by the National Science Foundation; and by the NASA Information Power Grid program.

References

1. I. Foster et al., "A Security Architecture for Computational Grids," *Proc. ACM Conf. Computers and Security*, ACM Press, New York, 1998, pp. 83-91.
2. M. Gasser and E. McDermott, "An Architecture for Practical Delegation in a Distributed System," *Proc. 1990 IEEE Symp. Research in Security and Privacy*, IEEE CS Press, Los Alamitos, Calif., 1990, pp. 20-30.
3. J. Steiner, B.C. Neuman, and J. Schiller, "Kerberos: An Authentication System for Open Network Systems," *Proc. Usenix Conf.*, Usenix Assoc., Berkeley, Calif., 1988, pp. 191-202.
4. I. Foster and C. Kesselman, "Globus: A Toolkit-Based Grid Architecture," *The Grid: Blueprint for a Future Computing Infrastructure*, I. Foster and C. Kesselman, eds., Morgan Kaufmann, San Francisco, 1999, pp. 259-278.
5. I. Foster and C. Kesselman, eds., *The Grid: Blueprint for a Future Computing Infrastructure*, Morgan Kaufmann, San Francisco, 1999.
6. J. Linn, "Generic Security Service Application Program Interface, Version 2, Update 1," RFC 2743, Internet Engineering Task Force, 2000, http://www.ietf.org/rfc/rfc2743.txt (current Oct. 2000).
7. T. Ryutov and C. Neuman, "Access Control Framework for Distributed Applications," Internet Draft, Internet Engineering Task Force, Nov. 1998; http://www.ietf.org/internet-drafts/draft-ietf-cat-acc-cntrl-frmw-04.txt.
8. Legal Policy Working Group, "Part B—The Model Certificate Policy," Government Information Technology Services, Federal PKI Steering Committee, http://gits-sec.treas.gov/model_cert_policy_cert.htm (current Oct. 2000).

Randy Butler *is division director for the Alliance Computational Environment and Security Division at the National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign. His work focuses on the planning and deployment of the Computational Grid infrastructure for the NCSA PACI environment. He received a BS in computer science from Illinois State University. Contact him at rbutler@ncsa.uiuc.edu.*

Von Welch *is a senior systems-engineer developer at the National Center for Supercomputing Applications. His research interests include networking, computer security, and wide-area distributed computation. He received a BS in computer engineering from the University of Illinois. Contact him at vwelch@ncsa.uiuc.edu.*

Douglas Engert *is on the staff of the Electronics and Computing Technologies Division at Argonne National Laboratory. His research interests include Kerberos, DCE, the Andrew file system, and the Globus Security Infrastructure. He received an MS in information sciences from the University of Chicago. Contact him at deengert@anl.gov.*

Ian Foster *is a senior scientist and associate director of the Mathematics and Computer Science Division at Argonne National Laboratory and a professor of computer science at the University of Chicago. His research interests include high-performance computing, distributed computing, and computational science. Foster received a PhD in computer science from Imperial College, London. He is a member of the ACM and the American Association for the Advancement of Science. Contact him at foster@mcs.anl.gov.*

Steven Tuecke *is the manager of the Distributed Systems Laboratory in the Mathematics and Computer Science Division at Argonne National Laboratory. His research interests include high-performance distributed Grid computing. He received a BA in mathematics and computer science from St. Olaf College. He is a fellow of the Computation Institute at the University of Chicago. Contact him at tuecke@mcs.anl.gov.*

John Volmer *heads the Information Technology Security Section in the Electronics and Computing Technology Division at Argonne National Laboratory. His particular focus has been the development of cross-organizational security policies. He received an MS in computer science from Northern Illinois University and an MBA from the University of Chicago. Contact him at volmer@anl.gov.*

Carl Kesselman *is a senior project leader at the University of Southern California's Information Sciences Institute. His research interests include distributed computing and networking. Kesselman received a PhD in computer science from the University of California at Los Angeles. He is a member of the IEEE. Contact him at carl@isi.edu.*