Security Architectures for Large-Scale Remote Environments[1]
(A DOE/ER/MICS DICCE Project Report)

Case Larsen, LBNL (clarsen@george.lbl.gov)
William Johnston, LBNL (wejohnston@lbl.gov)
May 14, 1997

Abstract

This report describes a project that is designing and implementing a prototype of a generalized public-key certificate infrastructure that can provide a highly scalable and transparent infrastructure for both authentication and authorization of access to network based services. The goal is to produce a generalized security architecture that can encode, distribute, and protect the information needed to enable routine secure availability of remote instrumentation in a distributed collaboratory environment. We report here on the progress of a prototype implementation that will demonstrate the concepts, identify the issues, and contribute to the evaluation of currently competing approaches for several aspects of the problem.

1.0 Introduction

1.1 The need

The need for strong, flexible security for on-line research instrumentation is motivated by the following observation:

> *"The access to a remote collaborative environment, whether it is for monitoring or for control of experiments, requires that security and access limitation mechanisms be in place, so that safeguards against unauthorized access and privacy of proprietary data exist.*
>
> *The advent of collaboratories brings a new class of user to the ALS. These users are likely to be much more occasional and less experienced with the equipment than has been the case in the past. Collaboratories will provide network based access to very expensive equipment and must be designed to avoid several potential security and safety problems. They must also be designed to have automated equipment failure modes with sanity checks on all incoming data and be resistant to network-based tampering. With respect to remote users, one significant issue is to ensure that use-conditions of the remote resource have been met."*
>
> *(From "Spectro-Microscopy Collaboratory at the Advanced Light Source Project Summary", http://www-itg.lbl.gov/~deba/ALS.DCEE)*

---

1. This work is supported by the U. S. Dept. of Energy, Energy Research Division, Mathematical, Information, and Computational Sciences office, under contract DE-AC03-76SF00098 with the University of California. Author's address: Bld. 50B, 2270, Lawrence Berkeley National Laboratory, Berkeley, CA 94720. Fax: +1-510-486-6363. This document is report LBL-38650.

1.2 The objectives

Provide communication integrity to any application and authorization to support any resource use model.

Give direct control over the granting of access to the responsible party. No intermediary (person, e.g. administrator) is involved in effecting the decision.

1.3 non-technical and technical hurdles

- **economic/market** -- Even though the idea of public-key cryptography has been around for over 20 years and that of general cryptography for even longer, the market for cryptographic software still behaves as a young market, with multiple proposed standards to choose from. There are currently a few different proposed cryptographic APIs and cryptographic frameworks [9] and two different certificate formats proposed as standards [3, 4]. The core component interfaces, such as those that provide private-key management, are at an early stage of standardization and are not uniform across commercial products yet.
- **political/regulatory** -- The regulation of encryption restricts the possible cryptographic functions that are easily accessible in commercial software. The Netscape browser, for instance, cannot provide RSA private-key operations to applications, not because it is technically difficult to provide such functions, but because these operations would allow key-exchange to occur and thereby conflict with existing cryptographic policy[2].
- Authentication and authorization is pervasive because it affects communication. All communication channels or communicating entities can have an identity associated with them and can potentially make use of authentication. Potentially any application written in any language and operating in any run-time environment can make use of authorization services.

2.0 Review of existing approaches to authorization

PolicyMaker [2] makes use of signed authorization "filters" defined in a safe general-purpose language. The authorization decision for whether a particular public-key is allowed to perform an action is made by evaluating a chain of filters over the action starting with a filter that is specified as local policy.

ADAGE [5] is a comprehensive environment for describing and enforcing security policies. Their main goal is to provide a usable policy description tool called "visual policy builder" (VPB) combined with algorithms for evaluating policy. They base their work on the Distributed Computing Environment (DCE) [10].

2.1 Our approach

We need to "own" the cryptographic implementation since the most basic operations of private-key signature and key-exchange were not available in commercial implementations. The use of OS provided cryptographic APIs (CAPIs) will eventually make such functionality widely available since the OS can enforce the cryptographic policy. The International Cryptography Experiment (ICE) [9]

2. Correspondence with Jim Roskind, Netscape.

stands a good chance of producing APIs that make cryptography available according to local policy across multiple platforms.

We use certificates (signed documents) to bind public-keys to identity, identities to attributes, and resource requirements to attributes.

We adopted the Generic Security Services Application Programming Interface (GSS-API) to support multiple security mechanisms (protocols). Although the Simple Public-Key Mechanism (SPKM) [1] is currently the only public-key cryptography based mechanism with a GSS-API interface, the interface is general enough that other protocols could be accommodated. There exist both DCE and Kerberos bindings [8] of the GSS-API which makes it possible to write applications that can easily switch among multiple mechanisms (SPKM, Kerberos, DCE).

To provide access to the private-key operations of signature and key-exchange without revealing the actual private-key data, we use the SSH agent present in the SSH suite of tools [11] which provides a "smartcard" abstraction that protects the user's private-key from disclosure by performing signature and key-exchange operations in the agent process.

The pervasiveness of authentication and authorization requires availability across multiple platforms and embedding of security services in multiple environments (stand-alone in servers, in applications with user-interfaces, across multiple run-time environments: C, Java, CORBA, and in applications with existing authentication protocols, e.g. SSL) We try to address this for authorization by using a separate authorization component that is not dependent on a particular strong authentication process and so is independent of authentication protocol. For authentication in stand-alone services, we allow the private-key to be loaded into the application's address space.

3.0 System architecture

The authentication architecture is as shown in Figure 1. The user submits his or her request for an

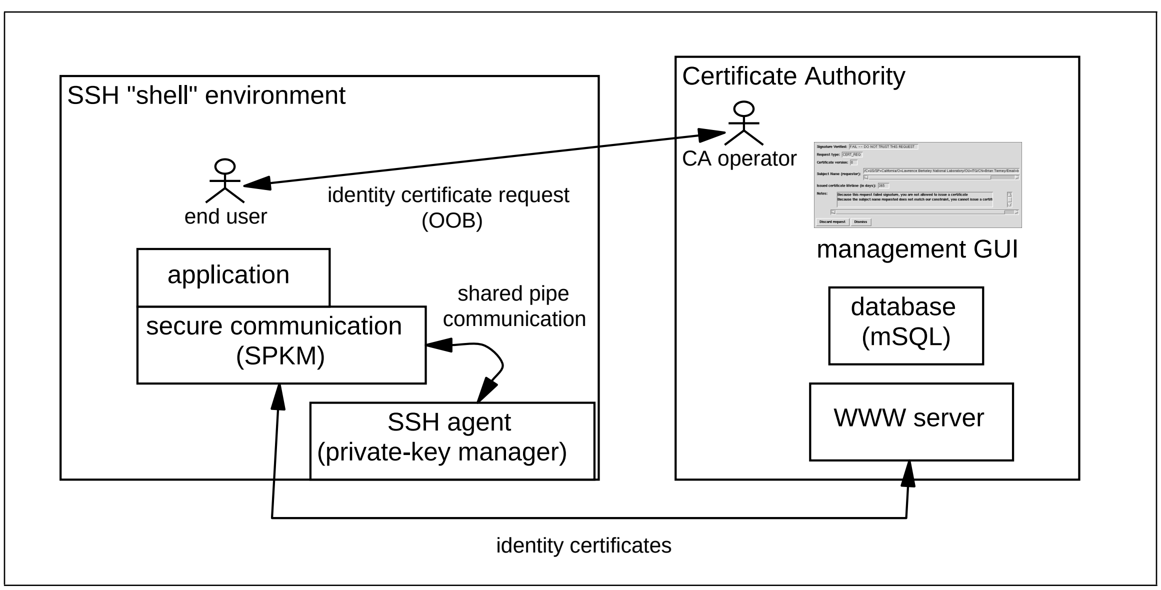

Figure 1 Authentication architecture

identity certificate to the certificate authority by out-of-band means, such as e-mail or floppy disk. The CA operator verifies the identity, generates a signed certificate and places it in a database which is served by a WWW front-end. When the user needs to authenticate or communicate with another party, he or she first decrypts and installs the private-key in a parent process (the SSH agent) which is accessible to all processes invoked from within child processes: command-line shells, xterms, applications, etc. The application uses the SPKM protocol for authentication. The protocol calls the agent to perform private-key signature and key-exchange during establishment. The protocol makes use of a web based directory service provided by the certificate authority to look up verification certificates for remote parties.

4.0 Results

We have:

- implemented the SPKM protocol using the SSLeay library of cryptographic primitives.
- implemented a WWW based directory service backed by a database.
- integrated certificate verification into the SSH application.
- defined general requirements for the private-key management component [6].
- explored the possible uses of public-key technology within the DOE business environment and product support for public-key technology under the DOE digital signature applications group. [7]

The policy evaluation components have not yet been implemented. If ADAGE [5] which will consume 6 man-years of effort when its contract is complete is typical of a policy specification support infrastructure, then there is much work remaining to be done, or a more restricted approach must be explored. Building infrastructure requires up-front effort before any of its benefits can be perceived. Although we chose to build the CA infrastructure from existing components because they were easily available and could be customized, it would have been more efficient to purchase a suitably customizable COTS product that would provide the infrastructure, the management functions, the management and access protocols and APIs. However, none was available at the time.

5.0 Notes and References

[1] Carlisle Adams. IDUP and SPKM: Developing public-key-based APIs and mechanisms for communication security services. In *Proceedings of the Symposium on Network and Distributed Systems Security (SNDSS'96)*. ISOC, 1996. Available at http://bilbo.isu.edu/sndss/adams.ps, See also http://bilbo.isu.edu/sndss/sndss96.html.

[2] Matt Blaze, Joan Feigenbaum, and Jack Lacy. Decentralized trust management. In *Proceedings of the Symposium on Research in Security and Privacy*. IEEE, 1996. Available at ftp://research.att.com/dist/mab/policymaker.ps,

[3] Carl M. Ellison, Bill Frantz, and Brian M. Thomas. Simple public-key certificate, March 1997. , Available at http://www.clark.net/pub/cme/spki.txt.

[4] R. Housley, W. Ford, and D. Solo. Internet public-key infrastructure, Part I: X.509 Certificate and CRL profile. Internet Draft, February 1996. Available at ftp://ftp.ietf.cnri.reston.va.us/internet-drafts/draft-ietf-pkix-ipki-part1-% 01.txt, Internet Draft (PKIX Working Group).

[5] Marty Hurley, Nimsha Meta, Rich Simon, and Mary Ellen Zurko. Authorization for distributed applications and groups. Technical report, OSF, 1996. Available at http://www.osf.org/www/adage/index.html,

[6] Case Larsen and William Johnston. Approaches to open, interoperable, and interchangeable interfaces for managing private-key cryptographic data. Technical report, Lawrence Berkeley National Laboratory, December 1996. , Available at http://www-itg.lbl.gov/ clarsen/security-ps/privkeyusage.1.3.fm.ps.

[7] Case Larsen and Frank Ploof. Scope and milestones for the use of PKI in DOE applications. Technical report, Lawrence Berkeley National Laboratory and Lawrence Livermore National Laboratory, February 1997. ,

[8] John Linn. The Kerberos version 5 GSS-API mechanism. Technical Report RFC 1964, June 1996. , Available at ftp://ds.internic.net/rfc/rfc1964.txt.

[9] TIS (Trusted Information Systems). International cryptography experiment, December 1996. Available at http://www.tis.com/docs/research/crypto/ice/index.html, From TIS's description: The International Cryptography Experiment (ICE) calls for a series of experiments to promote the international use of cryptography in common computer software applications (i.e., word processors, spreadsheets, electronic mail systems) in a manner that honors individual national controls on the import, export, and use of cryptography. The general approach being advocated by numerous organizations in several countries is the use of a Cryptographic Application Programming Interface (CAPI) to separate the actual performance of the cryptographic functions from the software applications that call for encryption of specific messages or files. ICE is an informally structured program to coordinate the many efforts underway in the U.S and several other countries to advance the general understanding of CAPIs and their use to promote international cryptography.

[10] The Open Group. Distributed Computing Environment (DCE) home page, 1997. ,

[11] Tatu Ylonen. The SSH (Secure Shell) remote login protocol. Internet Draft, 1996. , See also "SSH (Secure Shell) Remote Login Program" (http://www.cs.hut.fi/ssh/).