SecurSight: An Architecture for Secure Information Access

John G. Brainard
RSA Laboratories

Abstract

This paper describes SecurSight, an architecture that combines authentication, authorization, and secure communications. The primary goal of this architecture is to secure access to network resources, while providing a smooth migration path from legacy authentication and authorization methods to a public-key infrastructure. Authentication may utilize either shared secrets or public/private key pairs. Authorization is public-key based and provides both direct support for PKI-aware applications and indirect support for legacy applications. Authorization credentials are portable, and may be used in location-independent fashion, without the need for cumbersome export and import procedures.

1. Overview

The use of public-key technology for authentication and authorization in enterprise environments has been slow in deployment due to a number of factors. The use of "soft" public-key credentials in desktop applications such as Web browsers is convenient, but the portability of such credentials is limited. Truly portable public-key devices, such as smartcards, require infrastructure, in the form of smart card readers, which is not yet widely available.

Access control in applications faces something of a "chicken and egg" problem. Users do not want to go to the trouble of obtaining key pairs and certificates until their applications require it. Major application developers, on the other hand will not use public-key technology in their products until a critical mass of users has keys and certificates. This makes the conversion to public-key an all-or-nothing proposition, where all users and applications must be upgraded at once.

In this paper we will discuss an architecture, called SecurSight, that helps solve these problems by using existing authentication technology as a front end to public-key based services. It provides authorization services directly to public-key based applications. Older applications that know nothing of PKI are supported using public-key aware wrappers. This wrapping, combined with the storage of legacy credentials, such as passwords, inside public key based containers, allows the public-key infrastructure to exist without explicit awareness of it on the part of either end users or applications.

SecurSight's novelty is less in the specifics of its implementation than in the way it uses public-key infrastructure to enhance, rather than replace, existing security methods. This allows users to begin experimenting with public-key authentication and smartcards, without throwing away old passwords and one-time password generators. It allows new PKI-aware services to be integrated seamlessly, without abandoning existing applications.

1.1 SecurSight Design Principles

A primary goal of SecurSight is to provide centrally administered authentication and authorization for all users of a network, regardless of the resource to be accessed. The users log in to a trusted server and then obtain credentials for the resources they are authorized to use. The credentials have a short validity period, to allow administrators to grant and revoke privileges in a timely fashion.

SecurSight authentication is not restricted to a single method. Public-key based authentication, using both smartcards and keys stored in software, is supported. In addition, both traditional static passwords and various forms of one-time password (OTP) are allowed. The mechanism support is flexible enough that new mechanisms, such as biometrics, may be added without change to the basic design.

The authorization mechanism in SecurSight is the same for all users, regardless of the method used to authenticate. This gives all users potential access to the same set of resources. Administrators may require a particular type of authentication for a particular resource, but the choice is not dictated by the architecture.

With SecurSight, a user may log in and obtain authorization credentials from any enabled client system. Any additional credentials, beyond those used for initial authentication, are maintained by the authentication server and downloaded to the user after successful authentication.

The SecurSight system maintains a list of trusted authorities that is used to verify chains of public-key certificates at authentication time. This, in conjunction with the use of an external certificate validation service, moves much of the trust management problem away from the user's desktop where it may lead to both confusion and security errors. All trust paths are processed as part of a user's initial authentication; subsequent authorizations use only a trusted issuer, with no certificate chaining required.

1.2 Components

A SecurSight installation consists of six main components, desktops, managers, Privilege-Attribute Certificate (PAC) issuers, agents, certificate authorities, and a directory service. Each component is described briefly below.

The desktop is an application that runs on a user's desktop or portable computer. It provides the interface through which a user logs in and is authorized. It also provides a repository for the obtained authorization credentials. The desktop software also provides local security services, such as selective encryption of local files. The desktop communicates with the manager using the Cryptographic Security Services Protocol [CSSP]. TCP connections from the desktop to protected applications are redirected through an SSL connection to an agent.

The manager acts as both the authentication service and as the long-term repository for users' access rights. Users authenticate, from the desktop to the manager, then receive a privilege attribute certificate containing their authorization information

The manager must reside in a secure facility. It maintains an internal database containing authentication and authorization data for its users. The manager may be replicated to improve reliability and increase performance.

The manager is also referred to as the CSSP server, after the protocol it uses in communication with the desktop. CSSP provides for user authentication, using a variety of methods, and the subsequent delivery of authorization credentials. CSSP requests and responses are formatted as Lisp-style S-expressions. CSSP requires an underlying secure communications layer. This layer must authenticate the server to the client and provide confidentiality and integrity for the exchanged information. In SecurSight, this layer is provided by the Secure Sockets Layer [SSL] protocol, with server-side authentication only.

The PAC issuer is responsible for creating the short-term authorization credentials (PACs) for users after the manager authenticates them. The credentials are constructed using information from the manager's database and the directory service. In most implementations, the PAC issuer will be the manager, but it may be set up as a distinct service.

Agents come in two varieties: remote access agents and application connect agents. Remote access agents act as proxies for users not directly connected to the local network. These can be considered simplified versions of the desktop, with varying protocol requirements.

Application connect agents protect application-based services on the network. The applications may invoke SecurSight services directly, using an API. Existing applications, not modified to use SecurSight, may be protected with wrappers. These wrappers add both public-key based authorization and secure communications to database servers and other applications that normally provide no such security.

The SecurSight certificate authority issues the identity certificates used by the desktop and for other applications. This is an optional service; either an alternate CA or certificates from an external certification service may be used.

The directory service is an optional component in SecurSight that allows a subset of the authentication and authorization data to be maintained outside of the manager's internal database. Any service that supports version 2 of the Lightweight Directory Access Protocol [LDAP] may be used. Long-term secrets, such as passwords, symmetric keys, and private asymmetric keys may not be exported to the directory service.

2. Authentication: PSDs

The central concept of SecurSight authentication is the Personal Security Device, or PSD. A PSD is a unifying construct defined to enclose a user's authentication credentials, independent of whether the user performs authentication with a password, an OTP, or using a smart card. The PSD may be instantiated either in "hard" form, in a public-key based smart card, or in "soft" form, as data temporarily resident on the desktop. To allow for user mobility, the "soft" PSD data is maintained by the manager and downloaded to the user's desktop after a successful authentication.

PSD Field	Description
Version	An alphanumeric string representing the version number of the PSD format
PackageID	An alphanumeric string containing an identifier for the PSD. The identifier should be unique within the domain of the manager issuing the PSD.
Owner	The Distinguished Name of the PSD's owner.
PublicKey Info	An X.509 *SubjectPublicKeyInfo* structure containing the public key.
PrivateKey Info	An *EncryptedPrivateKeyInfo* structure, as described in PKCS #8 [PK8], containing the encrypted value of the private key corresponding to the above public key.
Attributes	A set of certificates and encrypted private keys belonging to the PSD's owner. This includes a certificate containing the key from the PublicKeyInfo above.
Usage	An integer value indicating whether the PSD may be used for encryption, signatures, or both.
Trusted Keys	A list of trusted public keys and certificates.

Table 1: PSD Fields

2.1 PSD Definition

The PSD consists of a user's private key, a corresponding public-key certificate, and a set of additional attributes. These attributes may be passwords, asymmetric key pairs, symmetric keys, or other user-specific information.

The individual fields of the PSD are described in Table1.

2.2 PSD Generation and Distribution

The manager generates a Soft PSD when a user is enrolled. PSDs may also be generated externally and uploaded to the manager, after authentication. The PSD may be downloaded to the user's desktop after a successful authentication. The manager may also be configured to download PSDs without authentication, with authentication required only to obtain the unlocking keys.

The desktop makes an authentication request to the manager, by sending a CSSP request message. The type of authentication may be negotiated with the server, or a default method may be used. A simplified exchange is represented below, with D representing the desktop and M representing the manager.

```
D->M: <UserID><Methods>
M->D: <Method><Challenge>
D->M: <Response>
M->D: <Result><Cookie>
D->M: <Acknowledge>
```

Figure 1: Authentication

The cookie value returned by the manager contains information specific to the authentication session, encrypted under a symmetric key known only to the manager. The cookie may be retained on the desktop and presented to the manager, as proof of authentication, in subsequent requests. Session encryption, as provided by the secure communications layer (SSL), protects the cookie from interception and replay.

Once the authentication is successful, the desktop may request the user's PSD from the manager. This is done using another CSSP exchange.

```
D->M: <PSDRequest><Cookie>
M->D: <PSD>
D->M: <Acknowledge>
```

Figure 2: PSD Download

The private key in the soft PSD is protected under a key-encrypting key (KEK). This key is stored in the PSD, encrypted using any of several methods, depending on the level of protection desired. If the PSD is protected by a static password, the KEK may be stored in the PSD encrypted under a key derived from the password. This option allows the PSD, once retrieved, to be used off-line, without interacting with the manager. The PSD may also be protected by a one-time password, either software generated or from a hardware token. In this case, the KEK is stored in the PSD encrypted under a symmetric key that is stored on the manager. This key may be downloaded, over the secure channel, from the manager after a successful authentication.

```
D->M: <PSDKeyRequest><Cookie>
M->D: <PSD Key>
D->M: <Acknowledge>
```

Figure 3: PSD Key Request

As an optimization, the desktop may cache the user's PSD, with private fields encrypted. If the PSD is cached, only the "unlocking" key needs to be obtained from the manager. Since the manager may modify the contents of the PSD, the desktop must check if the cached PSD is up to date.

If emergency access, in the case where a user's password is forgotten or token lost, is desired, the KEK may also be stored in the PSD encrypted under a PSD unlocking key or PUK. The PUK is kept in a secure facility, under control of the SecurSight administrator.

2.3 Soft PSD Usage

Once downloaded and decrypted, the private key in the soft PSD may be used to create secure connections to SecurSight protected resources. The desktop initiates an SSL session with the application connect agent at the protected resource. The PSD key is used to provide client-side authentication for the SSL session.

In addition to its role in SecurSight authorization, the soft PSD may be used in other applications. It may be used to sign or decrypt electronic mail. It may be used to gain access to SSL-protected web sites. Any application that is PSD aware, or is built from PSD aware libraries, may use the soft PSD as though it were a physical smartcard. In particular, the SecurSight soft PSD is designed to be used with standard APIs such as PKCS #11 [PK11] and Microsoft's CryptoAPI [CAPI].

The Soft PSD may contain more than one private key. If this is the case, the additional keys are kept in the attributes field of the PSD. The X.509 keyUsage extension is used to distinguish between keys used for authentication, encryption, or non-repudiation. The primary private key, in the PrivateKeyInfo field, is used for SSL client authentication with connect agents.

2.4 Comparison with Other Authenticators

Soft PSDs offer significant advantages over other forms of authentication. Static passwords are portable and may be cached to provide a form of single sign-on. They are subject to a number of attacks, and must be synchronized with every application that requires them. SecurSight soft PSDs provide secure authentication and require no synchronization with applications.

One-time passwords, from software or hardware tokens, provide stronger authentication than static passwords. To use one-time passwords with multiple applications, however, requires that each protected application either stores token secrets or communicates with a server that stores them. In addition, for hardware tokens, the user must perform a new authentication for each resource accessed, which quickly becomes burdensome.

SecurSight soft PSDs are most similar to the soft credentials used by web browsers. The browser-based credentials, however, have limited portability. Some browsers offer the ability to export and import credentials in the PKCS #12 interchange format [PK12]. This requires explicit action on the part of the user, and is not practical for a truly mobile user. The Soft PSD user, on the other hand, may authenticate from anywhere within the enterprise, without any special procedures.

Smartcards using RSA and other asymmetric algorithms, offer greater portability than SoftPSDs, as well as better physical protection of the private key. The smartcards require an infrastructure, in the form of card readers, which is not always present or easy to add.

3. Authorization: PACs

All the authorization information in SecurSight is encapsulated in a form of public-key certificate called a Privilege-Attribute Certificate, or PAC.

3.1 PAC Definition

Privilege Attribute Certificates (PACs) are used in SecurSight to convey information about what privileges or authorizations exists for a subject. PACs are short lived (with a lifetime on the order of hours) and are generated on demand by the PAC Issuer (PI) for access to user applications. A PAC is an internal structure used only by SecurSight components.

PACs are used to simplify the trust path management between desktop users and the application agents they are accessing. A user's identity certificate contained within the PSD may be generated by a range of different CAs interconnected in complex trust chains. These trust chains are validated when the certificate in the PSD is presented during desktop authentication. The user's identity and privileges are then stored as part of the PAC. This allows optimized trust processing when secure links are established within SecurSight. The application agent is required only to validate the PI and need not be concerned with all possible CAs that may have issued identity certificates.

PACs are exchanged and processed by several SecurSight components:

Desktop — The local access agent can request one or more PACs when an application that might need a PAC is to be started. When the local access agent sends a request for a PAC to the PAC Issuer, it may specify which applications it wishes to access.

PAC Issuer—The PAC Issuer receives requests for PACs from desktops. The issuer extracts the appropriate information from the user's Identity Certificate (DN, public key, etc.), adds the privilege extensions, sets the Issuer field to its own DN, and then signs the PAC.

Application — An application can be a client/server system (such as a database) or local to the desktop (for example, a local login). The connect agent verifies the signature and validity time of the PAC, extracts the access control information, and initiates the access control sequence for the application.

3.2 PAC Generation and Distribution

PACs are generated at the manager, in response to a PAC request from the desktop. PACs are provided only after a successful authentication. After authentication, the desktop may request a PAC for the user. The request may be generic, in effect requesting all of the user's authorizations, or contain "hints" as to the particular authorizations desired.

```
D->M: <PACRequest><Hints>
      <Cookie>
M->D: <PAC>
D->M: <Acknowledge>
```

Figure 4: PAC Request

3.3 Use of PACs by Connect Agents

Figure 5 illustrates the sequence of events when a user logs in at the desktop, then uses an application that requests access to a server. The user's PSD contains a certificate that has been issued by a CA outside of the organization, but one that is trusted.

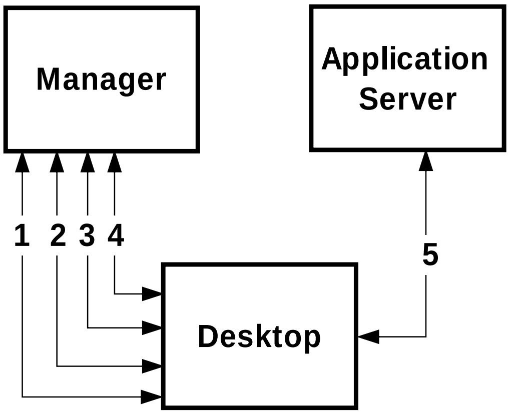

Figure 5: PAC Usage

The numbered items below correspond to the numbered transactions in Figure 5.

1. The user is authenticated at the desktop using a password, a one-time password, or hardware PSD.

2. For software PSDs only, if a current PSD is not cached on the desktop, it may be downloaded from the manager. This transaction requires a cookie from a recent successful authentication.

3. For OTP-protected soft PSDs, the key to decrypt the key-encrypting key in the PSD is then downloaded from the manager. This transaction also requires a prior authentication.

4. After authentication, when access to a network resource is needed, the desktop requests a PAC from the PAC Issuer, in this case the manager.

   The PAC request may be generic or it may contain information regarding the specific resource to be accessed. The PAC request also contains the user's identity certificate, from the PSD, and a cookie, to demonstrate prior authentication.

   The PAC issuer creates a PAC, using the public key and distinguished name from the supplied identity certificate and the user's authorization rights, or the requested subset. This PAC is signed by the issuer and returned to the desktop.

5. The desktop establishes a secure, authenticated channel to the Application Access Agent at the Application Server using the SSL protocol [SSL], with both server and client authentication. The PAC replaces the user's identity certificate in the client authentication portion of the SSL handshake.

   The Application Access Agent extracts the users login credentials from the PAC and communicates the users credentials to the application server. This process is dependent on the application server. The results of the login attempt are communicated back to the desktop.

   Once the session is established, the SSL channel is used to securely exchange data between the desktop and the application server.

3.4 PAC format

A PAC is a short-lived X.509 version 3 certificate [X509], with a defined extension handling access rights. PACs may be thought of as the public-key equivalent of the tickets used in systems like Kerberos [KERB]. The PAC may be used in place of an identity certificate in many applications. Some special constraints on the basic certificate fields are described below.

Serial Number – Unlike X.509 identity certificates, SecurSight PACs are short-lived, do not appear on revocation lists, and are not searched for by serial number, so the serial number is not required to be unique.

Validity – As specified in the Internet Public Key Infrastructure [PKIX], UTCTime is used for dates before 2050, and GeneralizedTime for dates after 2050. The validity period for PACs may be configured, at the issuer, but is typically on the order of 24 hours.

Subject – The subject distinguished name must match the subject name in the user's identity certificate.

Subject Public Key – Like the subject name, the public key is copied from the user's identity certificate.

Note that the optional Subject Unique Identifier and Issuer Unique Identifier attributes, as defined by X.509, are not used in SecurSight PACs.

3.5 PAC Extensions – EARs

The only extension vital to the PACs is the entity access right or EAR. The PAC will contain an EAR for each service for which the PAC authorizes access. The EAR contains any service-specific information for the user. Sensitive attributes, such as passwords, are kept in encrypted form.

EAR Field	Description
Service	The name of the service for which the user is authorized.
Method	The access method. This string is specific to the application connect agent on the target system.
Host	The host name on which the service resides. This should match the host name field of the distinguished name in the host PSD for the application connect agent.
Application Data	Application specific data, including access rights and passwords. The data is UTF-8 encoded as set of attribute value pairs, in the form VAR=Value. The values may be a mixture of cleartext and encrypted fields. The encrypted fields are protected with a session key that is, in turn, encrypted under the connect agent's public key.

Table 2: Entity Access Right (EAR) Fields

3.6 Comparison with Other Authorization Methods

SecurSight PACs play a similar role to the tickets used in Kerberos. Both are obtained by the user after an initial authentication, both are short-lived, and both contain information specific to a target service or resource. Unlike Kerberos tickets, however, SecurSight PACs are public-key based, so no shared secret needs to be maintained between the protected resource and the manager.

Both the privilege attribute certificate defined in Sesame [SESM], and the attribute certificate specified by X.509 provide a mechanism for binding authorization information to a user's public key. In both cases, unlike SecurSight, the authorization is in another certificate, linked to the user's identity certificate, and signed by a trusted authority.

Keeping authorization distinct from the identity certificate is useful in distributing management of authorization decisions, but it requires the end application to process and validate two certificates or chains instead of one. In addition, by maintaining the authorization information in a standard identity certificate, this information can be communicated to the target application as part of an SSL handshake, with no additional protocol requirements.

The use of PACs helps make SecurSight a scalable architecture, by requiring transactions with the central server only at authentication time, rather than every time a resource is accessed.

4. Trust Management

The desktop must be configured with the public keys of the managers from which it will obtain services. This trust relationship can be leveraged by allowing the manager to maintain a list of trusted certificate issuers. This list may then be downloaded to the desktop on request. The list is called the Issuing authority table or IAtable.

4.1 Certificate Authority

The Certificate Authority is an optional component of SecurSight that provides certification services. This CA may be used to issue certificates for use with a PSD on the desktop or it may be used independently of desktop and manager to register keys used in desktop browser or other applications

4.2 Certificate Validation Service

The Certificate Validation Service (CVS) is a verification entity within SecurSight. Two types of verification are supported, a simple CRL check, and a full certificate chain validation. The service is used by the desktop, primarily to verify the host certificates for application servers.

The CVS uses the CSSP protocol to communicate validation requests from the desktop to the validation service and to return the responses. SSL is used to protect the integrity of the communication and to authenticate the CVS server to the desktop The desktop supplies only the certificate to be validated. Any additional certificates needed for a full chain validation are retrieved from a directory, using LDAP.

A CVS transaction is quite simple. The desktop or other client application sends a request including the type of service and the certificate to be checked, if any. The CVS server responds with a status code indicating the result of the validation. The transaction can be represented as follows, where D represents the desktop and S represents the CVS server.

```
D->S: <Request><Type>[<Cert>]
S->D: <Status>
```

Figure 6: CVS Request

The types of validation service provided by CVS are described in the table below.

CVS Service	Description
Complete	Complete verification with both revocation check, and certificate signature verification.
no_crl	Verification of certificate signature, but no revocation check.
crl_only	Revocation check only, no signature verification.

Table 3: **CVS Services**

The status returned by CVS indicates if the validation succeeded, partly succeeded, or failed, and the reasons for failure, if any. The possible statuses are listed below.

CVS Status	Definition
0	Verified, in accordance with request
1	Verified, but unable to check revocation status
2	Expired
3	Revoked
4	Verification failed (bad signature, format, etc.)

Table 4: **CVS Status Codes**

The CVS server may cache the result of a request, to make subsequent validations of the same certificate more efficient.

CVS makes the desktop requirements for certificate validation much smaller. In traditional X.509 implementations, the client requests, or receives periodically, a certificate revocation list (CRL) signed by a trusted issuer. The client then checks whether the certificate in question is present on the CRL. CVS avoids the need to download and parse such CRLs, by delegating this responsibility to the CVS server.

The PKIX Online Certificate Status Protocol, or OCSP [OCSP], also centralizes processing of revocation lists, but it does not provide the additional services that CVS does. In particular, CVS verifies the signature on certificates, in addition to checking revocation status. Unlike OCSP, CVS may perform a full chain validation including the retrieval of issuer certificates. CVS also maintains the list of trusted roots, available for download to the desktop.

In contrast, the proposed Data Certification Service [DCS] provides a more extensive set of services than CVS. Specifically, DCS can validate arbitrary signed data, with timestamps, in addition to certificate chain validation. These are valuable services, but they are beyond the scope of the current SecurSight architecture.

The CVS architecture also allows for the addition of new validation methods, without any change required at the desktop.

5. Future Work

SecurSight may be enhanced in a number of ways. New authentication methods, such as biometric devices, may be supported. SecurSight's proprietary PACs may be enhanced or replaced with X.509-style attribute certificates. The Certificate Validation Service may be extended to provide a more generic set of PKI services, like those in DCS.

6. Conclusions

SecurSight provides authentication and authorization services that are similar to those provided by Kerberos, Sesame, or the Distributed Computing Environment [DCE]. The mechanisms used by SecurSight, while differing in specific details, are similar to the mechanisms in those architectures.

SecurSight places emphasis on support for legacy applications and hiding, to the extent possible, the infrastructure from the end user. These features may make it more acceptable in an enterprise environment. This may allow organizations to make a faster and easier transition from their current security methods to a public-key infrastructure.

Finally, SecurSight uses public-key credentials to establish secure connections from user desktops to existing applications. This allows critical data to be protected now, rather than waiting for new applications that employ public-key technology.

Acknowledgments

Special thanks to Alan Abrahams, Bill Duane, Peter Röstin, and the other architects at Security Dynamics for developing the architecture described here. Thanks to Vipin Samar of Sun Microsystems for his work on key formats, upon which much of the PSD design is based. Also thanks to Burt Kaliski, John Linn, and Magnus Nyström of RSA Labs as well as the anonymous referees for their suggestions on improving the content.

References

[CAPI] Microsoft Corporation, *Microsoft® CryptoAPI*, Version 2.0, September 1996

[CSSP] J. Brainard, “The CSSP Protocol: An Architectural Overview.” In *Proceedings of the 1998 RSA Data Security Conference*, January 1998

[DCE] The Open Group**,** *DCE 1.1: Authentication and Security Service,* Open Group Technical Standard C311, August 1997

[DCS] C. Adams and R. Zuccherato, *Internet X.509 Public Key Infrastructure Data Certification Server Protocols* , Internet Draft, draft-ietf-pkix-dcs-00.txt. work in progress, Internet Engineering Task Force, September 1998

[KERB] J. Kohl and C. Neuman, *The Kerberos Network Authentication Service (V5),* RFC 1510, Internet Engineering Task Force, September 1993

[LDAP] M. Wahl, T. Howes, and S. Kille, *Lightweight Directory Access Protocol (v3),* RFC 2251, Internet Engineering Task Force, December 1997

[OCSP] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams, *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP*, Internet Draft, draft-ietf-pkix-ocsp-07.txt, work in progress, Internet Engineering Task Force, September 1998

[PK8] RSA Laboratories, *PKCS #8: Private Key Information Syntax Standard*, version 1.2, November 1993.

[PK11] RSA Laboratories, *PKCS #11: Cryptographic Token Interface Standard*, Version 2.0, April 1997.

[PK12] RSA Laboratories, *PKCS #12: Personal Information Exchange Syntax Standard*, version 1.0, April 1997.

[PKIX] R. Housley, W. Ford, W. Polk, and D. Solo, *Internet Public Key Infrastructure: Part I: X.509 Certificate and CRL Profile*, RFC 2459, Internet Engineering Task Force, January 1999

[SESM] T. Parker, *A Secure European System for Applications in a Multi-Vendor Environment,* March 1992

[SSL] A. Frier, P. Karlton, and P. Kocher, *The SSL 3.0 Protocol*, Netscape Communications Corp., Nov 18, 1996

[X509] CCITT. *Recommendation X.509: The Directory - Authentication Framework*. 1988.