WLSE Express AAA Server Certificate Configuration Guide

December 2005

Corporate Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 526-4100

Customer Order Number:
Text Part Number: OL-8880-01

WLSE Express AAA Server Certificate Configuration Guide

About This Guide

Revised: December 6, 2005, OL-8880-01

This guide provides information to help you understand public key infrastructure (PKI) and Rabin-Shamir-Adelmann (RSA) certificates, how to generate certificates to be used with the Cisco Wireless LAN Security Engine (WLSE) Express, and how to configure AAA certificates to be used on WLSE-Express.

This guide contains the following chapters:

- Chapter 1, "Introduction," provides an introduction and overview of PKI and RSA certificates and general information about their use.
- Chapter 2, "Generating Certificates," provides a general overview of the steps involved in generating RSA keys and certificates without reference to specific tools. This chapter also includes examples of certificate generation based on OpenSSL and Windows Certificate Authority.
- Chapter 3, "Configuring AAA Certificates on WLSE," provides instructions for installing certificates and private keys onto the WLSE AAA Server.

This guide also contains an Index.

Obtaining Documentation

Cisco documentation and additional literature are available on Cisco.com. Cisco also provides several ways to obtain technical assistance and other technical resources. These sections explain how to obtain technical information from Cisco Systems.

Cisco.com

You can access the most current Cisco documentation at this URL:

http://www.cisco.com/techsupport

You can access the Cisco website at this URL:

http://www.cisco.com

You can access international Cisco websites at this URL:

http://www.cisco.com/public/countries_languages.shtml

Product Documentation DVD

Cisco documentation and additional literature are available in the Product Documentation DVD package, which may have shipped with your product. The Product Documentation DVD is updated regularly and may be more current than printed documentation.

The Product Documentation DVD is a comprehensive library of technical product documentation on portable media. The DVD enables you to access multiple versions of hardware and software installation, configuration, and command guides for Cisco products and to view technical documentation in HTML. With the DVD, you have access to the same documentation that is found on the Cisco website without being connected to the Internet. Certain products also have .pdf versions of the documentation available.

The Product Documentation DVD is available as a single unit or as a subscription. Registered Cisco.com users (Cisco direct customers) can order a Product Documentation DVD (product number DOC-DOCDVD=) from Cisco Marketplace at this URL:

http://www.cisco.com/go/marketplace/

Ordering Documentation

Beginning June 30, 2005, registered Cisco.com users may order Cisco documentation at the Product Documentation Store in the Cisco Marketplace at this URL:

http://www.cisco.com/go/marketplace/

Nonregistered Cisco.com users can order technical documentation from 8:00 a.m. to 5:00 p.m. (0800 to 1700) PDT by calling 1 866 463-3487 in the United States and Canada, or elsewhere by calling 011 408 519-5055. You can also order documentation by e-mail at tech-doc-store-mkpl@external.cisco.com or by fax at 1 408 519-5001 in the United States and Canada, or elsewhere at 011 408 519-5001.

Documentation Feedback

You can rate and provide feedback about Cisco technical documents by completing the online feedback form that appears with the technical documents on Cisco.com.

You can send comments about Cisco documentation to bug-doc@cisco.com.

You can submit comments by using the response card (if present) behind the front cover of your document or by writing to the following address:

Cisco Systems
Attn: Customer Document Ordering
170 West Tasman Drive
San Jose, CA 95134-9883

We appreciate your comments.

Cisco Product Security Overview

Cisco provides a free online Security Vulnerability Policy portal at this URL:

http://www.cisco.com/en/US/products/products_security_vulnerability_policy.html

From this site, you can perform these tasks:

- Report security vulnerabilities in Cisco products.
- Obtain assistance with security incidents that involve Cisco products.
- Register to receive security information from Cisco.

A current list of security advisories and notices for Cisco products is available at this URL:

http://www.cisco.com/go/psirt

If you prefer to see advisories and notices as they are updated in real time, you can access a Product Security Incident Response Team Really Simple Syndication (PSIRT RSS) feed from this URL:

http://www.cisco.com/en/US/products/products_psirt_rss_feed.html

Reporting Security Problems in Cisco Products

Cisco is committed to delivering secure products. We test our products internally before we release them, and we strive to correct all vulnerabilities quickly. If you think that you might have identified a vulnerability in a Cisco product, contact PSIRT:

- Emergencies—security-alert@cisco.com

  An emergency is either a condition in which a system is under active attack or a condition for which a severe and urgent security vulnerability should be reported. All other conditions are considered nonemergencies.
- Nonemergencies—psirt@cisco.com

In an emergency, you can also reach PSIRT by telephone:

- 1 877 228-7302
- 1 408 525-6532

Tip We encourage you to use Pretty Good Privacy (PGP) or a compatible product to encrypt any sensitive information that you send to Cisco. PSIRT can work from encrypted information that is compatible with PGP versions 2.*x* through 8.*x*.

Never use a revoked or an expired encryption key. The correct public key to use in your correspondence with PSIRT is the one linked in the Contact Summary section of the Security Vulnerability Policy page at this URL:

http://www.cisco.com/en/US/products/products_security_vulnerability_policy.html

The link on this page has the current PGP key ID in use.

Obtaining Technical Assistance

Cisco Technical Support provides 24-hour-a-day award-winning technical assistance. The Cisco Technical Support & Documentation website on Cisco.com features extensive online support resources. In addition, if you have a valid Cisco service contract, Cisco Technical Assistance Center (TAC) engineers provide telephone support. If you do not have a valid Cisco service contract, contact your reseller.

Cisco Technical Support & Documentation Website

The Cisco Technical Support & Documentation website provides online documents and tools for troubleshooting and resolving technical issues with Cisco products and technologies. The website is available 24 hours a day, at this URL:

http://www.cisco.com/techsupport

Access to all tools on the Cisco Technical Support & Documentation website requires a Cisco.com user ID and password. If you have a valid service contract but do not have a user ID or password, you can register at this URL:

http://tools.cisco.com/RPF/register/register.do

Note Use the Cisco Product Identification (CPI) tool to locate your product serial number before submitting a web or phone request for service. You can access the CPI tool from the Cisco Technical Support & Documentation website by clicking the **Tools & Resources** link under Documentation & Tools. Choose **Cisco Product Identification Tool** from the Alphabetical Index drop-down list, or click the **Cisco Product Identification Tool** link under Alerts & RMAs. The CPI tool offers three search options: by product ID or model name; by tree view; or for certain products, by copying and pasting **show** command output. Search results show an illustration of your product with the serial number label location highlighted. Locate the serial number label on your product and record the information before placing a service call.

Submitting a Service Request

Using the online TAC Service Request Tool is the fastest way to open S3 and S4 service requests. (S3 and S4 service requests are those in which your network is minimally impaired or for which you require product information.) After you describe your situation, the TAC Service Request Tool provides recommended solutions. If your issue is not resolved using the recommended resources, your service request is assigned to a Cisco engineer. The TAC Service Request Tool is located at this URL:

http://www.cisco.com/techsupport/servicerequest

For S1 or S2 service requests or if you do not have Internet access, contact the Cisco TAC by telephone. (S1 or S2 service requests are those in which your production network is down or severely degraded.) Cisco engineers are assigned immediately to S1 and S2 service requests to help keep your business operations running smoothly.

To open a service request by telephone, use one of the following numbers:

Asia-Pacific: +61 2 8446 7411 (Australia: 1 800 805 227)
EMEA: +32 2 704 55 55
USA: 1 800 553-2447

For a complete list of Cisco TAC contacts, go to this URL:

http://www.cisco.com/techsupport/contacts

Definitions of Service Request Severity

To ensure that all service requests are reported in a standard format, Cisco has established severity definitions.

Severity 1 (S1)—Your network is "down," or there is a critical impact to your business operations. You and Cisco will commit all necessary resources around the clock to resolve the situation.

Severity 2 (S2)—Operation of an existing network is severely degraded, or significant aspects of your business operation are negatively affected by inadequate performance of Cisco products. You and Cisco will commit full-time resources during normal business hours to resolve the situation.

Severity 3 (S3)—Operational performance of your network is impaired, but most business operations remain functional. You and Cisco will commit resources during normal business hours to restore service to satisfactory levels.

Severity 4 (S4)—You require information or assistance with Cisco product capabilities, installation, or configuration. There is little or no effect on your business operations.

Obtaining Additional Publications and Information

Information about Cisco products, technologies, and network solutions is available from various online and printed sources.

- Cisco Marketplace provides a variety of Cisco books, reference guides, documentation, and logo merchandise. Visit Cisco Marketplace, the company store, at this URL:

  http://www.cisco.com/go/marketplace/

- *Cisco Press* publishes a wide range of general networking, training and certification titles. Both new and experienced users will benefit from these publications. For current Cisco Press titles and other information, go to Cisco Press at this URL:

  http://www.ciscopress.com

- *Packet* magazine is the Cisco Systems technical user magazine for maximizing Internet and networking investments. Each quarter, Packet delivers coverage of the latest industry trends, technology breakthroughs, and Cisco products and solutions, as well as network deployment and troubleshooting tips, configuration examples, customer case studies, certification and training information, and links to scores of in-depth online resources. You can access Packet magazine at this URL:

  http://www.cisco.com/packet

- *iQ Magazine* is the quarterly publication from Cisco Systems designed to help growing companies learn how they can use technology to increase revenue, streamline their business, and expand services. The publication identifies the challenges facing these companies and the technologies to help solve them, using real-world case studies and business strategies to help readers make sound technology investment decisions. You can access iQ Magazine at this URL:

  http://www.cisco.com/go/iqmagazine

  or view the digital edition at this URL:

  http://ciscoiq.texterity.com/ciscoiq/sample/

- *Internet Protocol Journal* is a quarterly journal published by Cisco Systems for engineering professionals involved in designing, developing, and operating public and private internets and intranets. You can access the Internet Protocol Journal at this URL:

  http://www.cisco.com/ipj

- Networking products offered by Cisco Systems, as well as customer support services, can be obtained at this URL:

  http://www.cisco.com/en/US/products/index.html

- Networking Professionals Connection is an interactive website for networking professionals to share questions, suggestions, and information about networking products and technologies with Cisco experts and other networking professionals. Join a discussion at this URL:

  http://www.cisco.com/discuss/networking

- World-class networking training is available from Cisco. You can view current offerings at this URL:

  http://www.cisco.com/en/US/learning/index.html

Introduction

Revised: March 27, 2006, OL-8880-01

The CiscoWorks Wireless LAN Solution Engine (WLSE) Express AAA Server supports several Extensible Authentication Protocol (EAP) methods that use Rabin-Shamir-Adelmann (RSA) certificates and private keys. Currently, the supported mechanisms are Protected EAP (PEAP) versions 0 and 1 and EAP-Transport Level Security (TLS). Each of these protocols requires the administrator to perform certain preparation and configuration steps before they can be used. Typically these steps involve generation of one or more RSA certificates and private keys which are then made available to WLSE-Express using the graphical user interface (GUI).

The first part of the configuration process, generating certificates and private keys, can be somewhat complex depending on the tools used and the requirements placed on the certificate content and hierarchy. Certificate manipulation tools differ widely in their capabilities, style, and usability features. Few of the tools are automated and many of them require you to be knowledgeable about the Public Key Infrastructure (PKI) technologies involved. Local security policy, which can vary significantly by organization, typically specifies certain requirements and constraints placed on certificates and private keys. PEAP and EAP-TLS also impose some additional requirements on the certificates intended for their use.

The second part of the process, configuring WLSE Express with the certificates and private keys, is fairly simple and uses common procedures for the different EAP methods. Most of the challenge involved in configuring WLSE PEAP and EAP-TLS mechanisms exists in the first part of the process, generating the proper certificates and private keys. Since that activity is variable based on the tools used and is also influenced by external factors (primarily local security policy) a single general-purpose user guide cannot possibly document it in detail.

This document describes a typical configuration using two commonly available PKI tools, OpenSSL and Windows 2000 Server Certificate Authority. With a good understanding of the fundamental steps involved and the ability to experiment with a working installation, the WLSE administrator should be able to manage more complex configurations.

The section PKI and RSA Certificates, page 1-2, provides information necessary for a basic understanding of RSA certificates and private keys and how they are used by PEAP and EAP-TLS. Subsequent sections show detailed steps for generating certificates and private keys and configuring WLSE Express with them.

Acronyms

This document uses the following terms and acronyms.

Table 1-1 Acronyms

Term	Definition
AAA	Authentication, Authorization, and Accounting
CA	Certificate Authority
CLI	Command Line Interface
EAP	Extensible Authentication Protocol
PEM	Privacy Enhanced Mail
PKCS	Public Key Cryptography Standards
PKI	Public Key Infrastructure
RSA	Rabin-Shamir-Adelmann
TLS	Transport Level Security
WLSE	Wireless LAN Security Engine

PKI and RSA Certificates

This section provides a basic understanding of Public Key Infrastructure (PKI) and Rabin-Shamir-Adelmann (RSA) certificates.

PKI Basics

Public Key Infrastructure (PKI) is an extremely large and complex subject area. This section provides a brief outline of the essential information related to RSA certificates and keys to help you understand subsequent sections. Administrators having little or no prior experience with PKI are strongly recommended to seek out additional references that provide more substance such as Internet Request for Comments (RFCs), Internet-Drafts, technical books including those from Cisco Press, and other product manuals, and listings of frequently asked questions (FAQs).

PKI refers to a conceptual infrastructure (or model) for managing information related to public key cryptography. Public key cryptography is a security technology that uses keys that come in matched pairs. Each key pair consists of a widely distributed public key and a private key that is secret and closely protected by its owner. Information encrypted using the public key can only be decrypted using the matched private key (for example, by the key pair owner). This is the difference between public key encryption and conventional secret key cryptography where a single key is used for both encryption and decryption.

It is also possible to encrypt information using the private key that can then only be decrypted using the public key (the reverse of the scheme mentioned above). This technique can be used for creating digital signatures which are useful for proving data origin and integrity.

Public Key Distribution

One of the primary goals of public key cryptography is to solve the key distribution problem which is fundamental to most cryptographic technologies. This refers to the difficulty of securely distributing, storing, and managing keys needed to decrypt sensitive information. Public key cryptography solves this problem by defining a public key that can be distributed freely without protection. Only the corresponding private key must be kept secret, but that is typically not difficult because it is usually stored locally (close to point of origin) and rarely transferred.

Public Key Verification

Since public keys can be freely distributed, they can be placed in directories or databases for ease of access, or stored in local computer registries, or sent in e-mail messages. However, there is still a security problem with distributing and storing public keys. Although they do not need to be kept secret, they still need to be integrity protected and verified by their users Otherwise, an attacker could possibly substitute a bogus public key in place of the actual key (in a directory, for example) that would permit decryption of sensitive information intended for the original key owner. In fact, the attacker might be able to impersonate the actual key owner without detection. Therefore, public keys must somehow be authenticated prior to use.

PKI solves this problem by using digital signatures. As mentioned earlier, a digital signature can be created by encrypting some information (or a derivative of it) with a private key. Anyone can verify the digital signature by decrypting it with the corresponding public key. An attempt to alter the original information would cause the verification process to fail (decryption would yield an incorrect value) and therefore be easily detected. Digital signatures are analogous to conventional pen and paper signatures used to authenticate documents.

Therefore, to authenticate a public key, it can be digitally signed using another key pair. The public key used to verify the digital signature is known as a verification key. Verification keys are frequently pre-distributed and locally stored.

Certificates

Public key distribution and verification are very common operations in PKI-based systems. To promote interoperability, standards have been created to describe the precise format and encoding rules for signed public keys as well as details of the algorithms used for signing and verification. The data structure containing a signed public key is called a certificate. Certificates contain the value of the public key itself as well as additional information that describes the key and its intended purpose, identifies its owner and signer, and assists in verification, etc.

X.509

The most commonly used commercial standard is X.509 Version 3 (X.509v3) from the X.509 set of recommendations, although several standards exist for specifying certificate formats and operations.

The X.509v3 structure has two parts: a fairly rigid fixed part that contains common fields and a second part for optional fields known as extensions. A large number of optional extensions have been defined for a variety of purposes and certificate structures can be quite complex. To facilitate processing, many applications define a certificate profile which specifies the allowable field and extension content and whether extensions are mandatory, optional, or require special processing rules. Table 1-2 shows the X.509v3 certificate fields with some typical values.

Table 1-2 X.509v3 Certificate Fields

Field Name	Description or Typical Value
Version	Indicates the version of the certificate specification, usually version 3, displayed as: Version: 3 (0x2).
Serial Number	Unique identifier
Signature Algorithm	RSA-MD5
Issuer	Name of certificate issuer
Subject	Name of certificate owner
Not before	Certificate validity start date and time, such as Not Before: Sep 29 17:18:54 2005 GMT
Not after	Certificate validity end date and time, such as Not After: Sep 29 17:28:54 2006 GMT
Public key type and size	RSA / 2048 bits
Public key value	Actual key raw value
Extensions	Optional; content variable

Note In this document, the term certificate implies X.509v3 certificate.

RSA Algorithm

Several different types of public keys exist. A key's type implies its general structure and the algorithms used with it. One of the most common types is known as RSA, for the initials of its three inventors (Rabin-Shamir-Adelmann). The RSA algorithm is based on the computational difficulty of factoring large numbers. This document will focus exclusively on RSA keys and certificates.

To better understand how RSA certificates are used, you need to understand how RSA digital signatures work. Earlier it was mentioned that signing was accomplished by encrypting the source data with the private key. However, the RSA algorithm is extremely compute-intensive and performs poorly for bulk data encryption. For this reason, RSA is usually constrained to operate on only small amounts of data at a time. To sign efficiently, the source data is digested using a special algorithm into a small fixed-length value (typically 128-160 bits) that is encrypted instead of the source data. To verify the signature, the digest value is independently computed and compared with the decrypted value.

A digest algorithm is similar to a cyclical redundancy check (CRC) but with special cryptographic properties. MD5 (128 bits) and SHA-1 (160 bits) are two of the most common digest algorithms used with RSA. A digest algorithm takes an arbitrary amount of input and produces a unique, fixed-length output.

RSA Signing and Verification

Given the description of an RSA algorithm, digital signing can be described as follows:

Step 1 Compute a fixed-length digest of the source data.

Step 2 Encrypt the digest using the private key.

Step 3 Include the encrypted digest value as the signature value.

Signature verification can be described as follows:

Step 1 Compute the digest of the source data.

Step 2 Decrypt the signature value using the public key.

Step 3 Compare the computed with the decrypted digest value.

Step 2 of signature verification implies that the verifier is able to locate the public key that corresponds to the private key used to sign the certificate. The certificate's Issuer field usually contains the name of the entity that issued (or signed) the certificate. That information (sometimes combined with extensions that contain additional key identification information) can be used to help find the appropriate key.

Certificate Hierarchies

Normally, the public key used to verify a certificate is itself packaged in a certificate. This implies that certificates are organized into chains and that verification is a recursive process. Certificate chains are frequently hierarchical and tree-shaped. The root of the hierarchy is the most trusted certificate; since it does not have a signer, it signs itself. The root certificate usually signs one or more intermediate certificates which might then sign additional intermediate certificates until the final certificate in the chain is reached.

Certificate hierarchies are typically composed of three types of certificates:

- *Root* or *self-signed* certificates at the root of hierarchy.
- *Intermediate* certificates signed by the root or another intermediate certificate and used to sign other intermediate or end-entity certificates.
- *End-entity* certificates signed by a root or intermediate certificate and used to identify a component in the infrastructure, such as a user, computer, or process.

Local certificate stores are frequently organized around the types of certificates they contain. For example, web browsers usually contain one or more separate certificate folders for people, intermediate CA certificates, and root certificates.

While intermediate and end-entity certificates can be verified using the next higher certificate in the chain, the root certificate has no high certificate and cannot be verified (since it signs itself). Root certificates are always trusted implicitly and might also be verified using some out of band means. Web browsers and other applications typically bundle one or more root and intermediate certificates with the product, and trust in the root certificates is established by way of product installation.

Certificate Authorities

The organizational entity that issues certificates is known as a Certificate Authority (CA). Certificates used to sign other certificates are called CA certificates. In the hierarchy described above, levels 1 (root) and 2 (intermediate) are CA certificates and level 3 (end-entity) is not. Certificates used by servers and clients are end-entity certificates while certificates above them in the hierarchy are CA certificates. The WLSE Express can use either level 3 certificates from an external CA which must be uploaded to the server or its own self-signed (root) certificate.

Certificate Lifecycle

Good security practice dictates that keys be time-limited and subject to eventual expiration and possible renewal. This limits the potential exposure of keys that have been compromised or weakened by technological advance. Each certificate contains a pair of date-time fields that define its valid lifetime. The validity period must be checked during verification and expired certificates are considered invalid.

Certificates experience a life cycle similar to the following:

1. The public-private key pair is generated.
2. A certificate request is created.
3. The certificate request is submitted to the Certificate Authority (CA).
4. The CA issues the requested certificate (assuming the request is valid and accepted).
5. The certificate and private key are deployed. This might involve placing the private key into some form of secure storage and installing the certificate into one or more public repositories.
6. The certificate and private key are used for their intended purpose(s) until the certificate expires.
7. Following expiration, it should not be possible to use the certificate and verification should fail.
8. It might be acceptable to reissue a new certificate for the existing key pair, or they may permanently retired.

There are also mechanisms for revoking a certificate at any time during its life prior to its normal expiration. This might be useful, for example, if the private key has been compromised. Revocation status checking is beyond the scope of this document.

Certificate Encoding

X.509v3 specifies that certificates are encoded using ASN.1, a language for describing arbitrary information in a machine-independent fashion (conceptually similar to XML). ASN.1 is both a notation for describing data and a specification for how it should be encoded at the bit level. Tools are available for automatically converting data to and from ASN.1 form, given its ASN.1 description.

Several different encoding rules have been defined for ASN.1 to suit various purposes. The Basic Encoding Rules (BER) is the original ASN.1 encoding rule set that uses a Tag-Length-Value (TLV) encoding style. X.509v3, however, specifies that certificates are encoded using the Distinguished Encoding Rules (DER). The advantage of DER is that it eliminates ambiguity—there is only one valid encoding possible for every data value and independently developed encoding tools will always produce identical output given the same input.

Another data format that is commonly used for certificates and keys is PEM, for Privacy Enhanced Mail, which was an early specification for secure email. PEM defines a transformation for representing binary information as text with special encapsulating tags to help identify the encoded data. Figure 1-1 shows an example of a PEM-encoded certificate.

Figure 1-1 PEM-Encoded Certificate

```
-----BEGIN CERTIFICATE-----
MIICSDCCAbGgAwIBAgIBATANBgkqhkiG9w0BAQQFADBeMQswCQYDVQQGEwJVUzEL
MAkGA1UECBMCV0ExEDAOBgNVBAcTB1NlYXR0bGUxHjAcBgNVBAoTFUludGVybmV0
IFdpZGdldHMgSW5jLjEQMA4GA1UEAxMHUm9vdCBDQTAeFw0wNDExMDUyMzQ0MTNa
Fw0wNTExMDUyMzQ0MTNaMF0xCzAJBgNVBAYTAlVTMQswCQYDVQQIEwJXQTEQMA4G
A1UEBxMHU2VhdHRsZTEeMBwGA1UEChMVSW50ZXJuZXQgV2lkZ2V0cyBJbmMuMQ8w
DQYDVQQDEwZTZXJ2ZXIwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBANXdXewU
ehqIeA/GEMf5XBvprBBsayF1PFCEq5cBHO9Uj0rEEdcxyE8V6IYCHeNeaGtp7XQ4
Z1NmBFqeJ8/fN5kTCaNTYvQsSNp6e7F804JQRgBxcHpa8a9wgXxcJaBj+x8NARpR
CrE5W6fw40p4SK+xVnlvY0/FZTPXx3qYzGztAgMBAAGjFzAVMBMGA1UdJQQMMAoG
CCsGAQUFBwMBMA0GCSqGSIb3DQEBBAUAA4GBAKjYQ1eQiHpkRmb0NIA4aUSgJ2hX
PbG6RmuuyeUCfRftSuUC7cqdpEP6agkeaVGlUKZsHXOWjwrGR2R4nzvFxY0BK+7Z
vijegykLBdpB73Jr8Wv3P4Ja/HLleqDLIH9oPYhDKMP9/AbWbaiqNrzqXcQPeIz8
RrWfuDjKbdXJJ7ks
-----END CERTIFICATE-----
```

Note The WLSE AAA Server requires PEM-encoded certificates and private keys.

Many PC applications support an encoding format known as PKCS #12: Personal Information Exchange Syntax Standard (or simply PKCS12) which is one of the Public Key Cryptography Standards (PKCS) promoted by RSA Security Inc. PKCS is a set of specifications dealing with various aspects of public key cryptography. Many of the PKCS specifications have been accepted as industry standards and are now published as RFCs.

PKCS12 specifies a format for encoding certificates, keys, and other cryptographic information. Although PKCS12 is also based on ASN.1 it uses a completely different internal format and structure from X.509v3 and converting from one to the other requires a tool that understands both. Since PKCS12 is a binary format it cannot be processed with a text editor. A special tool that understands the PKCS12 syntax is required. On Windows PCs, files with a **.P12** or **.PFX** extension are treated as PKCS12 content and opening them with Internet Explorer will invoke the certificate (and key) import wizard. (PKCS12 is sometimes referred to as PFX, a name for an older version of the standard.)

Table 1-3 shows three common certificate and key encoding standards you should be familiar with.

Table 1-3 Certificate and Key Encoding Standards

Standard	Description
DER	Based on X.509v3 ASN.1 DER
PEM	Text encoding of ASN.1 DER information
PKCS12	Supported on the PC by **.P12** and **.PFX** extensions

Certificate-Based Authentication

Public key systems such as RSA can also be used for authentication of people, computers, and other entities. A simple authentication protocol is for the client to pick a random number, sign it, and send the signature to the server. The server can authenticate the client by verifying its certificate and confirming the validity of the signature on the random number.

Since only the entity that possesses the private key corresponding to the certificate could have produced the signature, the server can be confident that the client is who it claims to be. The server can rely on CA and end-entity certificates stored locally, or the client can send the certificate chain (minus the root) along with the signature. To achieve mutual authentication, the roles can be reversed and the server can supply the client with a signed random number and its CA and end-entity certificates.

This protocol is far too simple and insecure to be practical; as stated it has a number of problems including lack of message replay protection and the issue of how to securely link the actual entity to the certificate it provided. However, it is conceptually similar to how actual protocols use RSA for authentication.

TLS Authentication

Transport Level Security (TLS) is a protocol for securely negotiating a session that can be used to exchange protected information. It supports a variety of different authentication mechanisms, one of which is selected during the negotiation phase.

When RSA is the selected mechanism, the client always authenticates the server. But as commonly deployed, the server typically does not authenticate the client (although it might). RSA authentication of the client with TLS is strictly optional and commonly not done. If the application requires client authentication, it will usually be performed as a separate step inside the secure session.

For example, when a user accesses an SSL-protected website using a browser, the SSL layer at the client computer will authenticate the web server using the CA certificates that were pre-installed in the local certificate store. When server authentication is successful and the session has been established, you might be asked to enter a name and password into a form thus effecting client authentication. The client's credentials (username and password in this case) are protected by the SSL session.

The reason for not authenticating the client with RSA is to avoid the requirement for distributing certificates and keys to each client device. Many system administrators believe that large-scale deployment of client certificates and keys is difficult and expensive to manage, and might even pose security risks.

PEAP and EAP-TLS Authentication

PEAP and EAP-TLS leverage the TLS protocol to provide strong mutual authentication and other services such as identity protection, session key derivation, and fast reconnect (session resumption).

PEAP is a 2-phase protocol; the first phase negotiates a TLS session with server authentication and optional client authentication, and the second phase executes another complete EAP conversation protected within the TLS session. The second, or "inner", EAP conversation is used to authenticate the client thereby eliminating the need for client-side certificates. Commonly used inner methods are EAP-GTC used with Cisco's implementation of the PEAP version 1 (PEAPv1) draft and EAP-MSCHAPv2 used with Microsoft's implementation of PEAP version 0 (PEAPv0) draft.

EAP-TLS is essentially identical to phase one of PEAP; there is no second phase or inner EAP conversation used by EAP-TLS so clients must be authenticated during TLS session negotiation using certificates.

CHAPTER 2

Generating Certificates

Revised: March 27, 2006, OL-8880-01

Overview

This chapter provides a general overview of the steps involved in generating RSA keys and certificates without reference to specific tools. Following the overview, the sections Generating Certificates with OpenSSL, page 2-2 and Certificate Generation with Windows CA, page 2-6 provide examples based on OpenSSL and Windows Certificate Authority.

The actual mechanics of certificate generation are highly dependent on the tools used as well as the local security policies in effect. Some tools and policies might condense the three steps shown below into fewer (possibly one) steps or expand them into more steps. The degree of automation and direct user involvement also varies greatly and can range from a simple web form-based model with automatic certificate distribution to a more complicated procedure with multiple user interactions. Some CAs are set up to support online operations including certificate production while others might operate strictly offline and require more manual involvement.

RSA Key Generation

RSA keys have certain mathematical and cryptographic properties that require special software tools for the generation. Some tools will ask you to type on the keyboard during generation to create a source of randomness. This is because RSA keys are based on large random numbers.

RSA key pairs have two essential parameters that must be specified during creation. The first parameter is the key type which is always RSA. The second parameter is the key length in bits which can vary from 512 to 4096 bits (or even more). The key length is usually specified as part of the customers' security policy and it is difficult to give a generally applicable recommendation for it.

Certificate Request Creation

A Certificate Request (CR) is information packaged with the public key that specifies the type and general content of the desired certificate. It is usually packaged in a format based on PKCS#10 (one of the PKCS standards documented by RFC 2986) or Certificate Request Message Format (CRMF), an emerging standard from the IETF. The format of the CR is usually not important as long as the tools used to create and process it are compatible.

The CR usually contains the following:

- An RSA key-pair
- Subject name (possibly in DN format)
- Desired lifetime of the certificate
- Name or identification of the issuing (signing) CA
- Certificate extensions

PEAP and EAP-TLS require server certificates to include an extendedKeyUsage extension of *TLS Server Authentication* and client certificates to include an extendedKeyUsage extension of *TLS Client Authentication*. The method used to specify extensions and their values depends on the tool. The extendedKeyUsage extension contains one or more Object Identifier (OID) values which are specified as strings of dot-separated decimals. The appropriate values are shown below:

Table 2-1 Required Values of extendedKeyUsage

Server or Client	OID Value	Meaning
Server	1.3.6.1.5.5.7.3.1	TLS Server Authentication
Client	1.3.6.1.5.5.7.3.2	TLS Client Authentication

Certificate Generation

The CR is submitted to the CA to generate the actual certificate. This might happen immediately, upon request (such as using a web form) or there might be delays if the CA is operated offline or requires manual management approval to issue certificates.

Generating Certificates with OpenSSL

This section provides example of creating certificates with OpenSSL. The OpenSSL open source project includes a command line tool, **openssl**, used to create keys and certificates. OpenSSL has many other useful capabilities. For more information about the OpenSSL open source project, check their website:

http://www.openssl.org

The examples create an extremely simple certificate hierarchy consisting of two levels and three certificates. First a self-signed root certificate is created and then used to sign a server certificate and a client certificate. Most realistic certificate hierarchies contain one or more levels of intermediate CA certificates.

The following steps assume a Linux or BSD-style shell is active, but the commands for most other command line environments are similar.

The openssl.cnf Configuration File

The **openssl** command tool uses a configuration file usually named **openssl.cnf** for various parameters and other information related to creating certificate requests.

There are two ways to make the **openssl.cnf** file available to the command tool:

- Using the OPENSSL_CONF environment variable

export OPENSSL_CONF /opts/open/openssl.cnf

- Specifying the *–config* option on the command line

  openssl <*additional parameters*> config ./openssl.cnf

The CA and CA_Default sections of the **openssl.cnf** file are particularly important because they specify information related to the configuration of a CA. Figure 2-1 shows an example of an **openssl.cnf** file, from the OpenSSL distribution.

Figure 2-1 Example openssl.cnf File

```
####################################################################
[ ca ]
default_ca      = CA_default                # The default ca section

####################################################################
[ CA_default ]

dir             = ./ca                      # Where everything is kept
certs           = $dir/cert                 # Where the issued certs are kept
crl_dir         = $dir/crl                  # Where the issued crl are kept
database        = $dir/index.txt            # database index file.
new_certs_dir   = $dir/newcerts             # default place for new certs.

certificate     = $dir/root-cert.pem        # The CA certificate
serial          = $dir/serial               # The current serial number
crl             = $dir/crl.pem              # The current CRL
private_key     = $dir/private/root-key.pem  # The private key
RANDFILE        = $dir/private/.rand        # private random number file

x509_extensions = usr_cert                  # The extentions to add to the cert

# Comment out the following two lines for the "traditional"(and highly broken) format.
name_opt        = ca_default                # Subject Name options
cert_opt        = ca_default                # Certificate field options

# Extension copying option: use with caution.
# copy_extensions = copy

# Extensions to add to a CRL. Note: Netscape communicator chokes on V2 CRLs
# so this is commented out by default to leave a V1 CRL.
# crl_extensions  = crl_ext

default_days      = 365                     # how long to certify for
default_crl_day  s= 30                      # how long before next CRL
default_md        = md5                     # which md to use.
preserve          = no                      # keep passed DN ordering

# A few difference way of specifying how similar the request should look
# For type CA, the listed attributes must be the same, and the optional
# and supplied fields are just that :-)
policy            = policy_match
```

Required Certificate Extensions

PEAP and EAP-TLS require server certificates to include an extendedKeyUsage extension of *TLS Server Authentication* and client certificates to include an extendedKeyUsage extension of *TLS Client Authentication*. These extensions can be placed in a configuration file referenced on the **openssl** command line.

The following is an example of the required **certs-exts.cnf** extensions file:

```
[ server_exts ]
extendedKeyUsage = 1.3.6.1.5.5.7.3.1
[ client_exts ]
extendedKeyUsage = 1.3.6.1.5.5.7.3.2
```

Creating Test Certificates and Keys

Use the **openssl** command line tool to create certificates and keys for testing PEAP. The following sections provide examples of how to create a simple certificate hierarchy that consists of a single CA certificate, a single server certificate, and a single client certificate. Additional certificates and keys can be produced as needed for testing purposes.

Note Long commands are shown on multiple lines, and some of the commands will prompt you for additional input.

Creating a CA Directory

To create a CA directory, enter the following commands as a root user:

mkdir ca

cd ca

mkdir certs private reqs

echo '01' > serial

touch index.txt

chmod 0700 private

cd ..

Creating a Self-signed CA Root Certificate and RSA Key

Use the following command sequence to create a self-signed CA root certificate and RSA key.

openssl req -x509 -newkey rsa:1024 –keyout ./ca/private/root-key.pem -keyform PEM -out ./ca/certs/root-cert.pem -outform PEM -config ./openssl.cnf

Use the following command to display the certificate:

openssl x509 –in ./ca/certs/root-cert.pem –text

Converting a CA Certificate to PKCS#12

Use the following command sequence to convert a CA certificate to PKCS#12 format. This process is useful for importing a CA certificate to a Windows PC for testing purposes.

cat ./ca/certs/root-cert.pem ./ca/private/root-key.pem > ./ca/private/root-all.pem

openssl pkcs12 -export -in ./ca/private/root-all.pem -out ./ca/certs/root-cert.p12

Creating a Server Certificate Request and RSA Key

Use the following command sequence to create a server certificate request and RSA key.

openssl req -newkey rsa:1024 -keyout ./ca/private/server-key.pem -keyform PEM -out ./ca/reqs/server-req.pem -outform PEM -config ./openssl.cnf

Creating a Server Certificate from the Request

Use the following command sequence to create a server certificate from the request and reference the certificate extensions file and required server certificate extension.

openssl x509 -req -days 365 -in ./ca/reqs/server-req.pem -CA ./ca/certs/root-cert.pem -CAkey ./ca/private/root-key.pem -CAserial ./ca/serial -extfile ./ca/cert-exts.cnf -extensions server_exts -out ./ca/certs/server-cert.pem

Use the following command to display the server certificate:

openssl x509 –in ./ca/certs/server-cert.pem –text

Creating a Client Certificate Request

Use the following command sequence to create a client certificate request.

openssl req -days 365 -newkey rsa:1024 -keyout ./ca/private/client-key.pem -keyform PEM -out ./ca/reqs/client-req.pem -outform PEM -config ./openssl.cnf

Creating a Client Certificate from the Request

Use the following command sequence to create a client certificate from the request and reference the certificate extensions file and required client certificate extension.

openssl x509 -req -days 365 -in ./ca/reqs/client-req.pem -CA ./ca/certs/root-cert.pem -CAkey ./ca/private/root-key.pem -CAserial ./ca/serial -extfile ./ca/cert-exts.cnf -extensions client_exts -out ./ca/certs/client-cert.pem

Use the following command to display the server certificate:

openssl x509 –in ./ca/certs/client-cert.pem –text

Converting a Client Certificate and Private Key to PKCS#12

Use the following command sequence to convert a client certificate and private key to PKCS#12. This process is useful for importing a client certificate to a Windows PC for testing.

cat ./ca/certs/client-cert.pem ./ca/private/client-key.pem > ./ca/private/client-all.pem

openssl pkcs12 –export –in client-all.pem –out client-all.p12

Certificate Generation with Windows CA

This section provides examples of creating certificates using the Windows Certificate Authority (Windows CA). The Windows CA provides a web-based interface for requesting and retrieving certificates. The web forms permit you to create a new key pair or use an existing key, specify the desired certificate fields and attributes, and to submit the request to the CA for processing.

Note The Windows CA component is only available on Windows Server OS, not on client OS (such as Windows 2000 Pro or Windows XP). To generate certificates you will need a Windows Server set up and the Windows CA configured.

Usually an administrator will be required to manually review and grant or deny the request before the certificate can be accessed. (Windows CA can also be configured to automatically grant requests without administrator intervention.) The Certification Authority snap-in of the Microsoft Management Console (MMC) is used to review certificate requests and take the appropriate action. It can also be used for other purposes such as certificate revocation, renewal, etc.

After a certificate has been issued by the Windows CA it must be exported to a file so that it can be transported to the machine where it will be used. Although Windows can export certificates in DER or PEM format, if the corresponding private key is required (as it is for server and client certificates) then the certificate and private key will be bundled into a PKCS#12-formatted file. Since the required format for our purposes is PEM, the PKCS#12 content must be reformatted appropriately.

The following examples show an extremely simple certificate hierarchy consisting of two levels and three certificates. Most realistic certificate hierarchies will contain one or more levels of intermediate CA certificates. Since the root-level certificate is created when the Windows CA product is installed and configured, those steps are not shown here. The examples assume that the Windows CA has been configured for standalone operation, but the steps are essentially the same for other configurations.

The following examples assume that the Windows Certificate Authority product has been installed and configured. Since the exact installation steps vary depending on the version of Windows Certificate Authority and its configuration, those steps are not shown here. Refer to the appropriate Microsoft documentation for information about how to install Windows Certificate Authority.

Generating a Server Certificate

This section describes how to generate a server certificate.

Step 1 Use your browser to access the Windows Certificate Services web form using a URL like the following: **http://server-name/certsrv**. In the example below, the server name is w2ks.

Figure 2-2 Windows Certificate Services Form

Step 2 Select **Request a Certificate** and click **Next**.

The next window enables you to select the type of certificate request.

Figure 2-3 Selecting Certificate Request Type

Step 3 Select **Advanced request** and click **Next**.

The next window enables you to select the method used to request the certificate.

Figure 2-4 Advanced Certificate Requests

Step 4 Select **Submit a certificate request to the CA using a form**, then click **Next**.

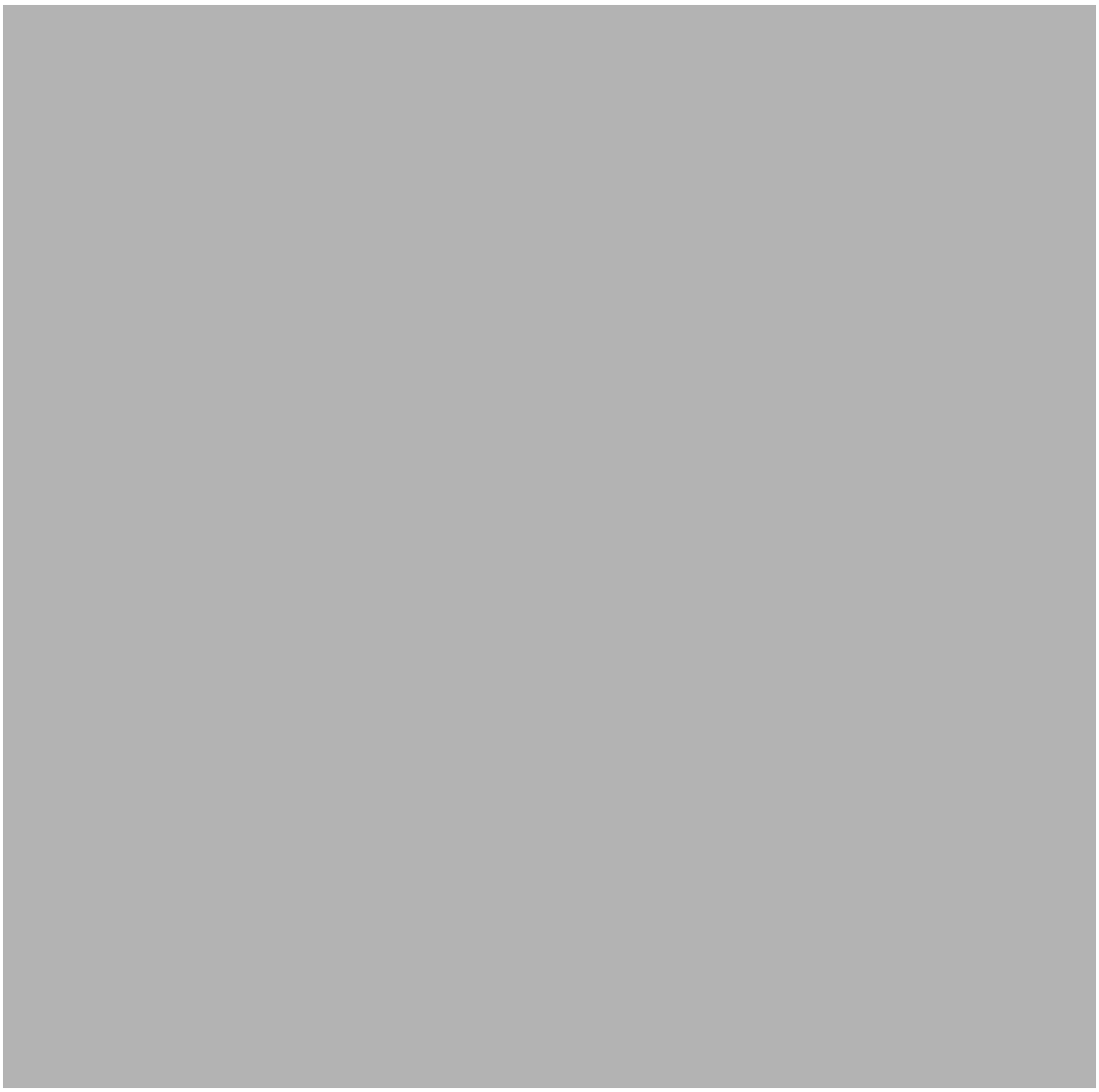
The Advanced Certificate Request form, Figure 2-5, allows you to specify some of the certificate's information content. You need only specify a few items; use the default values for the others.

In the *Identifying Information* section, **Name** is usually the name of the server. Use the default values of the other fields.

In the *Intended Purpose* section, select **Server Authentication Certificate**.

In the *Key Options* section, select **Mark keys as exportable**.

Figure 2-5 Advanced Certificate Request Form



Step 5 After you provide the information required for the form, click **Submit** to submit the request.

Note Depending on your configuration, you might be asked to confirm your request.

The next window acknowledges receipt of the request and advises you to check back later to retrieve the certificate.

Figure 2-6 Certificate Pending

Step 6 Click **Home** (near upper right corner of form) to return to the **Certificate Services** home page.

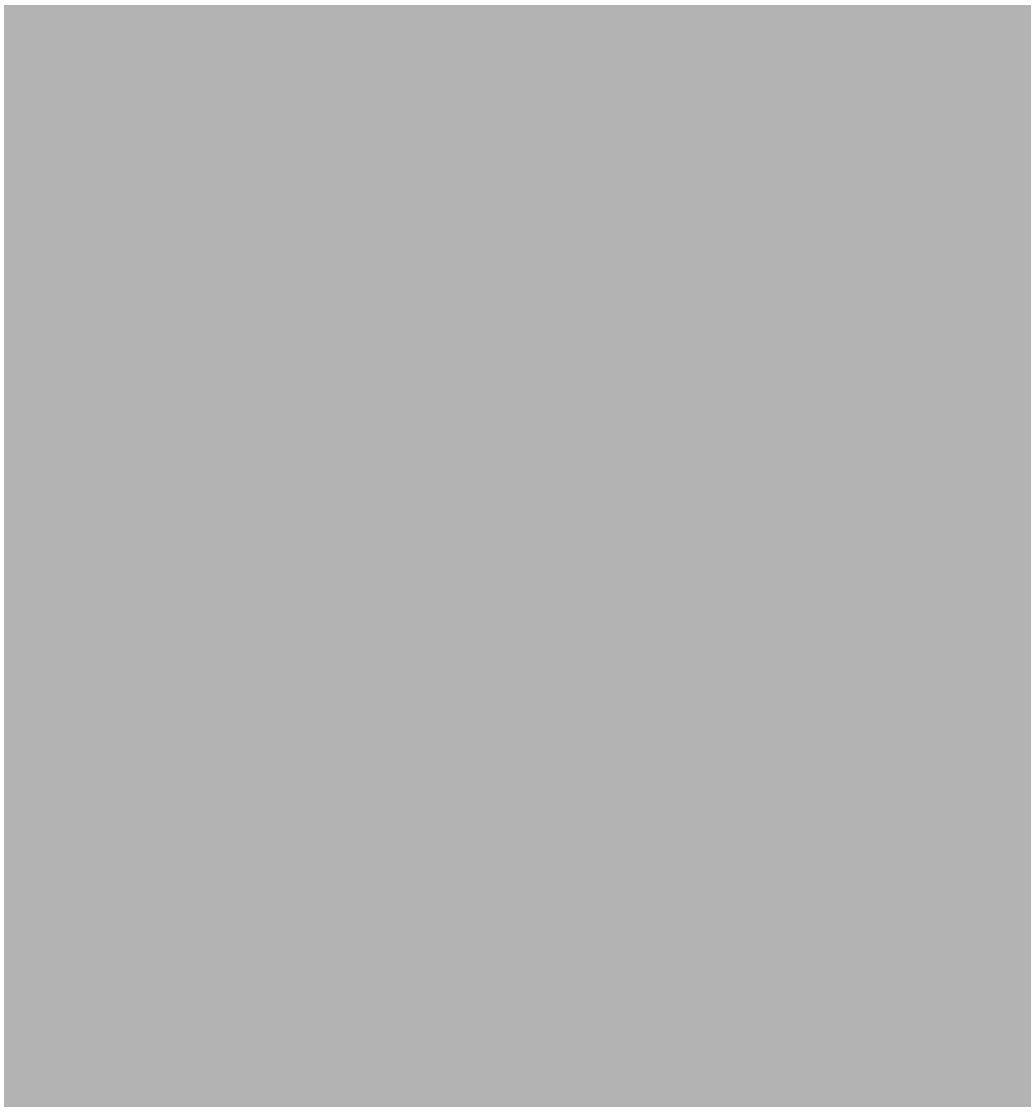
Generating a Client Certificate

The procedure to generate a client certificate is very similar to the procedure to generate a server certificate. The only significant differences are the value of the Name field and the Intended Purpose on the Advanced Certificate Request page.

Because this is a client certificate, the Name field should contain the user ID if the certificate is for an individual or the machine name if the certificate is for a computer. The value of the Intended Purpose field must be set to Client Authentication Certificate.

Figure 2-7 shows an example of the Advanced Certificate Request form for requesting a client certificate.

Figure 2-7 Example of Client Certificate Request Form



Certificate Retrieval

From the Certificate Services home page, select **Check on a pending certificate**.

Figure 2-8 Example of Check Pending Certificate Request

Step 7 Click **Next** to proceed.

Figure 2-9 shows an example of the pending certificates requests.

Figure 2-9 Check Pending Certificate Requests

Step 8 Select the appropriate request from the list and click **Next**.

If the certificate you request has not yet been granted, the Certificate Pending window displays as shown in Figure 2-10. This window provides a button to remove the certificate request.

Figure 2-10 Certificate Pending with Remove Option

Assuming that the certificate request was approved, the server displays the Certificate Issued window shown in Figure 2-11.

Figure 2-11 Certificate Issued

Step 9 Click **Install this certificate** to continue.

Note Depending on your configuration, you might be asked to confirm your request.

Figure 2-12 shows a confirmation of successful certificate installation.

Figure 2-12 Certificate Installed Confirmation

Exporting Server and Client Certificates

The certificate and private key must be exported from Windows before they can be installed on another machine. The easiest way to do this is to use the browser. This example uses Internet Explorer version 5.0. The procedure is the same for server and client certificates.

Navigate to the **Certificates** dialog box with these steps:

Step 1 Open the Tools menu.

Step 2 Select **Internet Options…**

Step 3 Click the Content tab.

Step 4 Click **Certificates**.

The Certificates dialog box will display the certificates installed on this computer, as shown in Figure 2-13.

Figure 2-13 Certificates Dialog

Step 5 Select the appropriate certificate to export and click **Export…** to initiate the Certificate Export Wizard as shown in Figure 2-14.

Figure 2-14 Certificate Export Wizard

Click **Next** to continue. The Export Private Key window displays as shown in Figure 2-15, which enables you to export the private key with the certificate.

Figure 2-15 Export Private Key

Step 6 Select **Yes, export the private key** and click **Next** to continue.

The next window allows you to select the format of the certificate file. Since we are exporting both the certificate and the private key, the only format permitted is PKCS#12.

Figure 2-16 Export File Format

Step 7 Click **Next** to continue.

The next screen prompts you to enter the password used to protect the PKCS#12 content.

Figure 2-17 Export Wizard Password

After entering the password, click **Next** to continue.

The next screen prompts you to specify (or browse to) the name of file to export.

Figure 2-18 File to Export

Step 8 After entering the file name, click **Next** to continue.

Figure 2-19 shows the settings selected through the Certificate Export wizard.

Figure 2-19 Completing the Certificate Export

Step 9 Click **Finish** to complete the export operation.

If successful, the message shown in Figure 2-20 displays to indicate a successful export.

Figure 2-20 Successful Export

Exporting CA Certificates

The procedure shown in Exporting Server and Client Certificates, page 2-16, can also be used to export CA certificates. In the Certificates dialog box, click the appropriate tab for either Intermediate Certification Authorities or Trusted Root Certification Authorities and scroll the list to locate the certificate you want to export. (Other categories might also be present depending on how the certificate store is configured.)

Figure 2-21 shows an example of the certificates to export window.

Figure 2-21 Certificates to Export

The rest of the procedure is the same as for server and client certificates except that usually the private key of CA certificates are not exported. In that case, the enabled exported certificate file format options will be a little different. Instead of just PKCS#12, the two formats will be DER or Base-64 encoded DER as shown in Figure 2-22.

Figure 2-22 Export File Formats

Converting PKCS#12 to PEM

A certificate and private key that have been exported from Windows will generally be encapsulated in a single file in PKCS#12 format. Before they can be installed on WLSE, they must be reformatted into PEM formatted files. It is a good security practice to store the certificate and private key in separate files. You can use the **openssl** command line tool from the OpenSSL open source project to do this.

Most Windows systems will not have the **openssl** tool installed, so it is usually easier to copy the PKCS#12 certificate file to a computer that does support OpenSSL.

There are two steps involved:

Step 1 Convert the PKCS#12 file to PEM format using **openssl**.

Step 2 Split the resulting PEM file into separate certificate and private key files (optional).

The following command line converts the content of **server.pfx** from PKCS#12 to PEM and places the result into **server.pem**:

openssl pkcs12 –in server.pfx –out server.pem

If the PKCS#12 file has been password protected, **openssl** will prompt you for the password. If the conversion succeeds, the **server.pem** file will contain the certificate and private key in PEM format. An example of a converted PEM file follows:

```
Bag Attributes
    localKeyID: 01 00 00 00
    1.3.6.1.4.1.311.17.1: Microsoft Base Cryptographic Provider v1.0
    friendlyName: 9191ccb399024e88287768944c8053cc_e0808bc1-0ea6-4702-85a9-1ccdee37a5c7
Key Attributes
    X509v3 Key Usage: 10
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,B22C8C19F46F9824

CcZ1Y5B5wu0l3bs/4wCv01RutRBLaRMdWb4QyRfPt6CpWnxZDkgvDVRyXnLiaw+g
75zYbWCMFMLsLiUeDr7yp0gSjpsOihwpdY4MEZL3V35iGIBl/dxOZ0dhywz8YoCo
D1XGQXBPTu3Uun/tZeyxISQDQFeGKopg1RvJzCdCEfF3hlg9bIWfFNuRhIOlvZoV
KeEO1TjsqtbH9hZTUyLH0dzfsix+xEw4Assc03KB4bEse+6Uk4Q3H7qFSwQNluxM
8zLpXMcIxSE0I+i9OdLbshUwBhKuW9/ksvvmneuaL7GCQRVvYj+QQmzAFPm0gYch
zDdBm+E86P8+J2/KlxEBWrFSpbgwpr6wM5SmqETH6VOIy2xaT+VTspDnp2jtJfyl
2/fOrG4V2njxuutQfGK9S2W49uVlcp1oss6iF4e7hVIY0+G1c/6LS8QFcQJoLbmw
ymmFRgi8j0jYGQWNpj7Awax+2Bpa+0wuWb/teCMX77/HRIJna+r8J0FbSWm3Z2JV
HAAvmtxO8SDJpuWSid1AVp7CFPESSpQHHy77dSFnGN3GBDmQTgUonbdN1ESCIMBM
LBIbaokDRt2HuoJXCaWa9vot2ssimidYlz0sOoTO2hHr8ai0dQErDGao5jAFzTGq
drKf+jA04GOagjknMfrNAfnnsOU1rKJAeW0oa24Z3YLep0LNhrf5H0ZuvrbDmv5x
KBlgCONuNNDcuEffjY6f/9MVpZX6dM80jjtD71xE/GR0oNUCsOhFTJZwGLSTJPS0
p3wxFswLNmqB2cWZgUifYX+UcQxkjb5quEeAqhY2UJujGu+yP12s9A==
-----END RSA PRIVATE KEY-----
Bag Attributes
    localKeyID: 01 00 00 00
subject=/C=US/ST=WA/L=Seattle/O=Engineering/OU=IT/CN=Server
issuer= /C=US/ST=WA/L=Seattle/O=Engineering/OU=IT/CN=Root CA
-----BEGIN CERTIFICATE-----
MIIDyzCCA3WgAwIBAgIKYbFeTwAAAAAAAzANBgkqhkiG9w0BAQUFADBhMQswCQYD
VQQGEwJVUzELMAkGA1UECBMCV0ExEDAOBgNVBAcTB1NlYXR0bGUxFDASBgNVBAoT
C0VuZ2luZWVyaW5nMQswCQYDVQQLEwJJVDEQMA4GA1UEAxMHUm9vdCBDQTAeFw0w
NTA5MjkxNzE4NTRaFw0wNjA5MjkxNzI4NTRaMGAxCzAJBgNVBAYTAlVTMQswCQYD
VQQIEwJXQTEQMA4GA1UEBxMHU2VhdHRsZTEUMBIGA1UEChMLRW5naW5lZXJpbmcx
CzAJBgNVBAsTAklUMQ8wDQYDVQQDEwZTZXJ2ZXIwgZ8wDQYJKoZIhvcNAQEBBQAD
gY0AMIGJAoGBALmJ4jD5IPDHS36RdmemhLCP0Q14bQzE8ngcFPcIsUY7YeK28tw2
GejsiAIE+kTEUdqPtVhqLT7NAQQykZjIzGNRpyiNuEJYDDsBow6J/SnHdPhTmAGJ
X0+YgCQdgb+3N6fSAOAv0APVcIY1/aN7cGhrSbYH4F11Nl0sfUSWLclZAgMBAAGj
ggHKMIIBxjAOBgNVHQ8BAf8EBAMCBPAwEwYDVR0lBAwwCgYIKwYBBQUHAwEwHQYD
VR0OBBYEFMo+Uu+Kxo8h/RWVtBLKTdIqxSC8MIGaBgNVHSMEgZIwgY+AFKD7ph9y
oLwfDhG6IWO+lIrZZd9ioWWkYzBhMQswCQYDVQQGEwJVUzELMAkGA1UECBMCV0Ex
EDAOBgNVBAcTB1NlYXR0bGUxFDASBgNVBAoTC0VuZ2luZWVyaW5nMQswCQYDVQQL
EwJJVDEQMA4GA1UEAxMHUm9vdCBDQYIQEkPt5Jo2DJ9J/drIo1njrjBjBgNVHR8E
XDBaMCqgKKAmhiRodHRwOi8vdzJrcy9DZXJ0RW5yb2xsL1Jvb3QlMjBDQS5jcmww
LKAqoCiGJmZpbGU6Ly9cXHcya3NcQ2VydEVucm9sbFxSb290JTIwQ0EuY3JsMH4G
CCsGAQUFBwEBBHIwcDA1BggrBgEFBQcwAoYpaHR0cDovL3cya3MvQ2VydEVucm9s
bC93MmtzX1Jvb3QlMjBDQS5jcnQwNwYIKwYBBQUHMAKGK2ZpbGU6Ly9cXHcya3Nc
Q2VydEVucm9sbFx3MmtzX1Jvb3QlMjBDQS5jcnQwDQYJKoZIhvcNAQEFBQADQQDT
mBDMuK1OZR7p8TtsEGH8G3qa7QnHtPGfEtaW5iFq72eqyQNx0lys7sl36iY/+S98
we3LO2Vyo9QegXLm5tv9
-----END CERTIFICATE-----
```

You might want to split the certificate and private key into two separate PEM files for security purposes. Since PEM is a text format, you can use any common editor (such as **vi** or **emacs**) to derive, for example, **server-cert.pem** and **server-key.pem** from **server.pem**.

Configuring AAA Certificates on WLSE

Revised: March 27, 2006, OL-8880-01

This chapter provides instructions for installing and configuring certificates and private keys onto the WLSE AAA Server. The examples assume that three files in PEM format (required by the WLSE) reside in a folder on your computer:

- The CA root certificate **root-cert.pem**
- The server certificate **server-cert.pem**
- The server private key **server-key.pem**

The certificate and private keys files can be created and exported to your computer using the procedures shown in sections Generating Certificates with OpenSSL, page 2-2, or Certificate Generation with Windows CA, page 2-6.

Using the GUI for Certificate Configuration

The following example shows how to configure the settings for Cisco-PEAP. The steps for MS-PEAP and EAP-TLS are essentially the same.

Step 1 Using the WLSE Express GUI, navigate to the Cisco-PEAP Settings page by selecting **Admin > AAA Administration > Cisco-PEAP Settings**.

Step 2 Enter the password used to protect the private key file into the Private Key Password and Confirm Private Key Password fields.

Step 3 Enter the full paths to the server certificate, server private key, and CA root certificate files respectively into the next three fields. The easiest way to do that is to click the Browse buttons and navigate to the files.

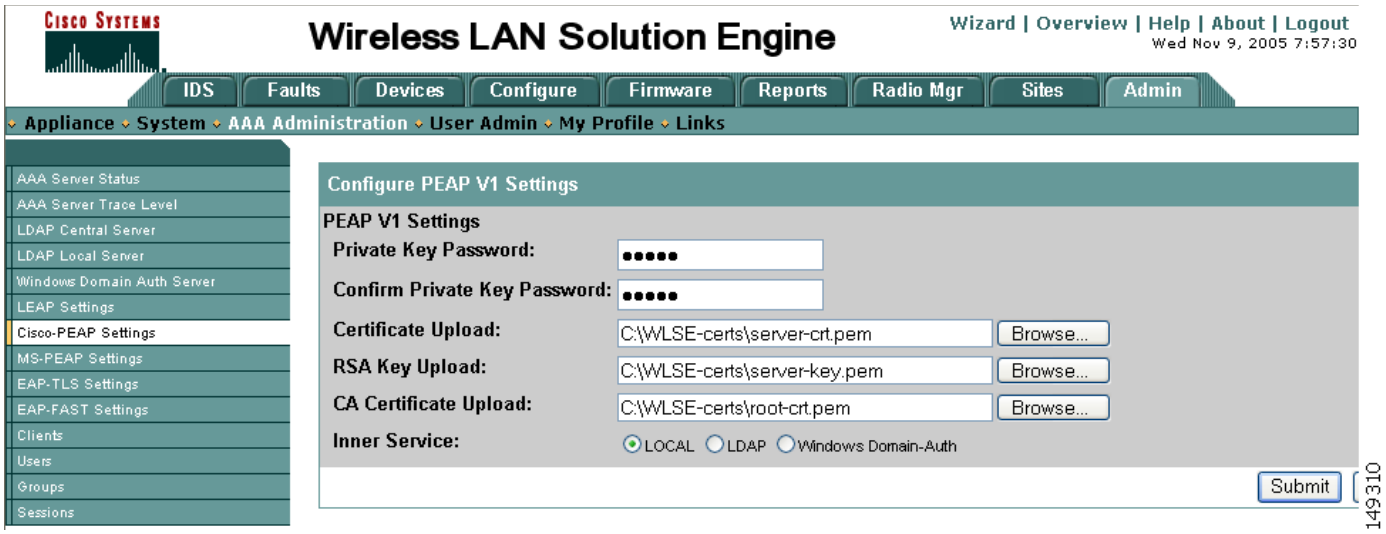
Step 4 Select the appropriate Inner Service from the choices shown.

Inner Service is the service used by PEAP to perform authentication. Select Local to fetch the username and password from a local database. Select LDAP to fetch the username and password from a remote LDAP directory. Select Windows Domain-Auth to use the Windows username and password.

Step 5 After all fields on the form have been completed, click **Submit**.

This uploads the certificates and private key and makes the appropriate configuration changes to the AAA Server. Figure 3-1 shows an example of a completed form.

Figure 3-1 Configuring PEAP Settings



Note The fields where you enter the pathnames to the certificate files are cleared after the upload, so there is no visual confirmation that the certificate upload has succeeded.

Using the CLI for Certificate Configuration

You can use the WLSE CLI to generate default certificates and private keys for PEAP and EAP-TLS with the **mkcert** command option. The **mkcert** command generates a new RSA key pair and a self-signed certificate. You can specify parts of the name included in the certificate, or simply rely on the default values provided. Since the certificate is self-signed, the name will be used for both the Subject and Issuer.

Use the **mkcert** command to return the AAA server to a known valid state that permits the server to be restarted if failing due to an invalid certificate.

An example for Cisco-PEAP follows. Typical input is shown in **bold** font.

```
admin@wlse:
```

aaa-server cisco-peap mkcert

```
file /cisco-ar/certs/cisco-peap/server-cert.pem already exists, do you want to overwrite?
[y/n]:
```

y

```
We will now generate an RSA key-pair and self-signed certificate that
may be used for test purposes
Generating a 1536 bit RSA private key
...++++
.........................++++
writing new private key to '/cisco-ar/certs/cisco-peap/server-key.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [US]:
Organization Name (eg, company) [WLSE]:
Organizational Unit Name (eg, section) [AAA]:
Common Name (eg, server or installation name) [wlse.cisco.com]:
Server self-signed certificate now resides in /cisco-ar/certs/cisco-peap/server-cert.pem
Server private RSA key now resides in /cisco-ar/certs/cisco-peap/server-key.pem

Remember to install additional CA certificates for client verification admin@wlse:
```

Generating default certificates and private keys for MS-PEAP and EAP-TLS is essentially the same. Use a command like the following to generate a certificate for MS-PEAP:

aaa-server ms-peap mkcert

Use a command like the following to generate a certificate for EAP-TLS:

aaa-server eap-tls mkcert

A

C

D

E

G

I

M

O

P

R

S

T

W

X