MiniCA: A web-based certificate authority

James Patrick Macdonell

Follow this and additional works at: https://scholarworks.lib.csusb.edu/etd-project

Part of the Information Security Commons

Recommended Citation
https://scholarworks.lib.csusb.edu/etd-project/3256

This Project is brought to you for free and open access by the John M. Pfau Library at CSUSB ScholarWorks. It has been accepted for inclusion in Theses Digitization Project by an authorized administrator of CSUSB ScholarWorks. For more information, please contact scholarworks@csusb.edu.
MINICA: A WEB-BASED CERTIFICATE AUTHORITY

A Project
Presented to the
Faculty of
California State University,
San Bernardino

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
in
Computer Science

by
James Patrick Macdonell
March 2007
MINICA: A WEB-BASED CERTIFICATE AUTHORITY

A Project
Presented to the
Faculty of
California State University,
San Bernardino

by
James Patrick Macdonell
March 2007

Approved by:  

Dr. Kay Zemoudeh, Chair, Computer Science

Dr. Richard Botting, Computer Science

Dr. David Turner, Computer Science

Dr. Javier Torner, Information Security Officer, Physics
ABSTRACT

The MiniCA project is proposed and developed to address growing demand for inexpensive access to security features such as privacy, strong authentication, and digital signatures. These features are integral to public-key encryption technologies. Digital certificates are vital in improving access to and use of new security technologies. Granting and revocation of digital certificates is overseen by a certificate authority whose procedures and policies are enforced through a software interface. MiniCA is designed to be a user friendly certificate authority interface. Since the certificate authority lays a foundation for a public-key infrastructure, MiniCA promotes the adoption and growth of public-key encryption technologies and the security features they provide.
ACKNOWLEDGMENTS

Thank you, Dr. Kay Zemoudeh, Dr. Dick Botting, and Dr. David Turner for guiding my work and showing me how to improve. Thank you, Dr. Javier Torner, for the early vision of a certificate authority and the support of the Information Security Office. Most importantly, thank you, my wife Claire and my parents Pat and Karlene, for your steadfast encouragement and support.
### TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>iii</td>
</tr>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td>iv</td>
</tr>
<tr>
<td>TABLE OF FIGURES</td>
<td>ix</td>
</tr>
<tr>
<td>CHAPTER ONE: INTRODUCTION</td>
<td></td>
</tr>
<tr>
<td>1.1 Digital Certificates</td>
<td>1</td>
</tr>
<tr>
<td>1.2 MiniCA Overview</td>
<td>4</td>
</tr>
<tr>
<td>1.2.1 Open Source</td>
<td>4</td>
</tr>
<tr>
<td>1.2.2 Nested Certificates</td>
<td>4</td>
</tr>
<tr>
<td>CHAPTER TWO: OVERVIEW OF ENCRYPTION</td>
<td></td>
</tr>
<tr>
<td>2.1 Encryption Tools</td>
<td>7</td>
</tr>
<tr>
<td>2.1.1 One-way Functions</td>
<td>7</td>
</tr>
<tr>
<td>2.1.2 Symmetric-Key Encryption</td>
<td>8</td>
</tr>
<tr>
<td>2.1.2.1 Distribution and Scalability</td>
<td>10</td>
</tr>
<tr>
<td>2.1.3 Public-Key Encryption</td>
<td>12</td>
</tr>
<tr>
<td>2.2 Encryption Services</td>
<td>13</td>
</tr>
<tr>
<td>2.2.1 Confidentiality</td>
<td>14</td>
</tr>
<tr>
<td>2.2.2 Data Integrity and Digital Signatures</td>
<td>14</td>
</tr>
<tr>
<td>2.2.3 Non-Repudiation</td>
<td>16</td>
</tr>
<tr>
<td>2.2.4 Authentication</td>
<td>17</td>
</tr>
<tr>
<td>CHAPTER THREE: THE ROLE OF A CERTIFICATE AUTHORITY</td>
<td></td>
</tr>
<tr>
<td>3.1 Structure of a Certificate Signing Request</td>
<td>18</td>
</tr>
<tr>
<td>3.2 Managing Certificate Signing Requests</td>
<td>21</td>
</tr>
<tr>
<td>3.3 Validating Identities</td>
<td>22</td>
</tr>
<tr>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>3.3.1 Structure of a Certificate</td>
<td>23</td>
</tr>
<tr>
<td>3.3.2 Creating Certificates</td>
<td>27</td>
</tr>
<tr>
<td>CHAPTER FOUR: MINICA'S IMPLEMENTATION</td>
<td></td>
</tr>
<tr>
<td>4.1 Managing Certificate Signing Requests with MiniCA</td>
<td>29</td>
</tr>
<tr>
<td>4.2 Validating Identities with MiniCA</td>
<td>32</td>
</tr>
<tr>
<td>4.3 Creating Certificates with MiniCA</td>
<td>34</td>
</tr>
<tr>
<td>CHAPTER FIVE: MINICA SOFTWARE REQUIREMENTS SPECIFICATION</td>
<td></td>
</tr>
<tr>
<td>5.1 Introduction</td>
<td>38</td>
</tr>
<tr>
<td>5.1.1 Purpose</td>
<td>38</td>
</tr>
<tr>
<td>5.1.2 Scope</td>
<td>39</td>
</tr>
<tr>
<td>5.1.3 Definitions, Acronyms, and Abbreviations</td>
<td>40</td>
</tr>
<tr>
<td>5.1.4 References</td>
<td>46</td>
</tr>
<tr>
<td>5.1.5 Overview</td>
<td>46</td>
</tr>
<tr>
<td>5.2 Overall Description</td>
<td>47</td>
</tr>
<tr>
<td>5.2.1 Product Perspective</td>
<td>47</td>
</tr>
<tr>
<td>5.2.1.1 System Interfaces</td>
<td>47</td>
</tr>
<tr>
<td>5.2.1.2 User Interfaces</td>
<td>47</td>
</tr>
<tr>
<td>5.2.1.3 Hardware Interfaces</td>
<td>48</td>
</tr>
<tr>
<td>5.2.1.4 Software Interfaces</td>
<td>49</td>
</tr>
<tr>
<td>5.2.1.5 Memory Constrains</td>
<td>49</td>
</tr>
<tr>
<td>5.2.1.6 Operations</td>
<td>50</td>
</tr>
<tr>
<td>5.2.1.7 Site Adaptation Requirements</td>
<td>50</td>
</tr>
<tr>
<td>5.2.1.8 Communications Interfaces</td>
<td>50</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>2.1</td>
<td>Analogy for a One-way Function</td>
</tr>
<tr>
<td>2.2</td>
<td>Symmetric-key Encryption Key Usage</td>
</tr>
<tr>
<td>2.3</td>
<td>Confidentiality and Key Exchange</td>
</tr>
<tr>
<td>2.4</td>
<td>Symmetric-key Distribution</td>
</tr>
<tr>
<td>2.5</td>
<td>Digital Signatures</td>
</tr>
<tr>
<td>2.6</td>
<td>Creating a Digital Signature</td>
</tr>
<tr>
<td>2.7</td>
<td>Verifying a Digital Signature</td>
</tr>
<tr>
<td>3.1</td>
<td>Structure of a Certificate Signing Request</td>
</tr>
<tr>
<td>3.2</td>
<td>Example Certificate Signing Request</td>
</tr>
<tr>
<td>3.3</td>
<td>Structure of a Digital Certificate</td>
</tr>
<tr>
<td>3.4</td>
<td>Example Digital Certificate</td>
</tr>
<tr>
<td>3.5</td>
<td>Creating a Digital Certificate</td>
</tr>
<tr>
<td>4.1</td>
<td>MiniCA Certificate Signing Request Submission Interface</td>
</tr>
<tr>
<td>4.2</td>
<td>MiniCA Certificate Signing Interface</td>
</tr>
<tr>
<td>4.3</td>
<td>Example Email Notification</td>
</tr>
<tr>
<td>5.1</td>
<td>Use Case Diagram</td>
</tr>
<tr>
<td>5.2</td>
<td>View Certificates Interface (1 of 2)</td>
</tr>
<tr>
<td>5.3</td>
<td>View Certificates Interface (2 of 2)</td>
</tr>
<tr>
<td>5.4</td>
<td>View Certificate Details Interface</td>
</tr>
<tr>
<td>5.5</td>
<td>Authentication Interface</td>
</tr>
<tr>
<td>5.6</td>
<td>Submit a Certificate Signing Request Interface</td>
</tr>
<tr>
<td>5.7</td>
<td>Certificate Signing Interface</td>
</tr>
</tbody>
</table>
Figure 5.8. View Certificates Sequence Diagram ....... 60
Figure 5.9. Submit Certificate Signing Request
  Sequence Diagram .......................... 61
Figure 5.10. Request Status Sequence Diagram .......... 62
Figure 5.11. Certificate Signing Sequence Diagram ..... 63
Figure 5.12. Certificate Maintenance Sequence
  Diagram ..................................... 64
Figure 5.13. Database Design .......................... 65
CHAPTER ONE
INTRODUCTION

1.1 Digital Certificates

Digital certificates are a necessary, but sometimes expensive, commodity essential for the way sensitive transactions such as on-line banking can take place over the Internet today. Digital certificates will also be essential for the security of sensitive files and databases in the future.

A Certificate Authority (CA) is used to issue digital certificates in much the same way the Department of Motor Vehicles (DMV) issues drivers license as identification cards. If a person applying for a drivers license meets all the criteria and follows all the procedures set forth by the DMV, the DMV will issue that person a drivers license. Similarly, if an entity applying for a digital certificate meets all the criteria and follows all the procedures set forth by a CA, the CA will issue that entity a digital certificate. Both drivers licenses and digital certificates are official documents. Both are also certificates.
A certificate is an official document issued by an authority to associate two (or more) independent pieces of information. For example, a diploma is issued by an authority, such as a high school or University, and is used to associate a student's name with a level of education. A drivers license is another example of a certificate. A drivers license is issued by an authority, the Department of Motor Vehicles, and is used to associate a photograph with a name and address. Digital certificates are no different; they are issued by a Certificate Authority to associate a special number called a public key with an identity, such as the name of a person, the name of a computer, or an email address.

For all certificates, the document is trustworthy only if both the issuing authority is trustworthy and the document is difficult to forge. A diploma can be trusted to a certain degree, provided it was issued by a credible high school or University. However, even if the diploma appears to be issued by trusted authority, the document can be forged and additional certificates such as embossed transcripts are necessary to trust that the name on the diploma actually has the stated level of education. A drivers license can be trusted to a higher degree than a
diploma. When the DMV, a trustworthy agency of the government, associates a photograph with a name and address, it includes an easy-to-recognize but hard-to-duplicate marking such as hologram in order to make its certificate, the drivers license, difficult to forge. Similarly, when a CA associates a private key with an identity, it includes an easy-to-recognize but hard-to-duplicate "marking" called a digital signature. If the Certificate Authority is operated by a trustworthy agency, then the certificates it issues can be trusted to a high degree, just like a drivers license.

Trustworthy certificates can be expensive to obtain. For example, VeriSign ®, a trustworthy Certificate Authority, may charge upwards to $850 per year for a digital certificate. However, a certificate is required for successful deployment of public-key encryption technologies such as secure web sites, which protect the privacy and integrity of data being passed between a web site and a client over the Internet. Despite the benefits, public-key encryption technologies are deployed sparingly or reluctantly because of high cost.
1.2 MiniCA Overview

MiniCA project aims to encourages the deployment of public-key encryption technologies and the security features they provide by reducing the cost and complication of issuing trustworthy certificates. Costs will be reduced as MiniCA will be offered to the public as a free, open source software package. This will allow institutions to create digital certificates as needed. To ease complication, MiniCA will provide an easy-to-install, user friendly web-based interface.

In comparison to certificate authority software products provided by Microsoft® and the OpenCA project, MiniCA possesses some unique features.

1.2.1 Open Source

Unlike Microsoft's certificate authority software, MiniCA will be released as open-source software. This means that institutions which use MiniCA will be able to modify and contribute to the functionally of the software as they see fit.

1.2.2 Nested Certificates

In contrast to OpenCA, MiniCA expressly supports nested certificates. Nesting certificates is a practice of using a CA to issue an identity to a subsidiary CA, which
in turn is used to either issue an identity to a subsidiary of the subsidiary or to issue the actual desired digital certificates. For example a certificate authority with the identity “Tier-One” may issue a certificate to a subsidiary certificate authority with the identity “Tier-Two”, which in turn may issues certificates to be used for secure web sites.

Nesting certificates mitigates the risk posed by an attacker being able to forge digital certificates. Should an attacker compromise a subsidiary certificate authority, an uncompromised, superseding CA would revoke the digital certificate of the compromised CA. Thereafter, clients authenticating certificates issued from the compromised CA will see a break in the chain of trust. This practice also protects the private key of the highest superseding or “root” certificate from exposure. When using nested certificates, this key is only needed two situations: during the initial deployment of the certificate authority to create subordinate certificate authorities, and to revoke a certificate in the rare case that a subordinate certificate authority becomes compromised. Consequently, the root certificate private key may be archived off-line
for essentially the life of the certificate authority, thereby protecting it from exposure.

The alternative security practice of maintaining an on-line registration authority and a separate off-line certificate authority, the method promoted by OpenCA, will not be directly supported by MiniCA. This alternative method protects the private key of the root certificate from exposure by never allowing it to be placed on a network accessible computer. Rather, certificate requests are copied from an on-line registration authority to removable media and carried by hand to the off-line certificate authority for signature. Although this technique provides the maximum protection for the private key against exposure and avoids some certificate installation issues caused by nested certificates, it adds expense to the deployment of a certificate authority with an extra system to maintain and adds tediousness to the role of the certificate authority administrator.
CHAPTER TWO
OVERVIEW OF ENCRYPTION

Encryption is an essential tool used to provide confidentiality, data integrity, authentication, and non-repudiation services. These services are provided through one-way (hash) functions, symmetric-key encryption, and public-key encryption.

2.1 Encryption Tools

One-way functions, symmetric-key encryption, and public-key encryption contribute to providing the desired services of encryption.

2.1.1 One-way Functions

One-way functions, also called hash functions or cryptographic digests, accept arbitrary input data to calculate a hash value or a digest. It is relatively easy to compute a hash value for a given input, however it is "computationally infeasible" to calculate an input value from a hash value "for most [hash values]" [7]. That is, it is easy to calculate a hash value from input data but it
is hard (or impossible) to calculate input data from a hash value, figure 2.1.

Data integrity, authentication, and non-repudiation services all depend on one-way functions that rarely, if ever, produce collisions. A collision occurs when two distinct input values result in the same hash value when passed through a one-way function.

2.1.2 Symmetric-Key Encryption

Symmetric-key encryption, sometimes called shared-key encryption, uses one key for both the encryption function
which translates plaintext into cyphertext and the corresponding decryption function which translates the cyphertext back into plaintext, figure 2.2. When compared to public-key encryption, symmetric-key cyphers are computationally less intensive and therefore can encrypt or decrypt more data and at a faster rate [7]. However, distribution of symmetric-keys creates infrastructure and scalability problems.

![Figure 2.2. Symmetric-key Encryption Key Usage](image)

In order for symmetric-key encryption to remain secure, the encryption key must remain secret and the key must be changed regularly [7]. Keeping symmetric-keys secret requires that a secure channel be established to allow the distribution of keys. This infrastructure problem can be addressed by transferring the key by fax or
telephone (if that network is considered secure) or copying the key to removable media and delivering the key to the receiving party by hand. These methods work when there are only a handful of parties participating. However, they become tedious as the number of participants grows.

2.1.2.1 Distribution and Scalability

As an example of how symmetric keys are distributed, consider three people Alice, Bob, and Candice who communicate with each other using symmetric-key encryption as proposed by Garfinkel [5]. In order to prevent eavesdropping, each pair of participants must generate and exchange a secret symmetric key. Alice and Bob communicate using the key AB to prevent Candice from snooping, Bob and
Candice communicate using the key BC to prevent Alice from snooping, and so on to create three keys in total, figure 2.4(a). Now suppose Daisy wishes to participate. Alice, Bob and Candice each now need to generate and exchange a symmetric key with Daisy (AD, BD, CD). Adding one more person required three more keys, making the total number six, figure 2.4(b). When Eric joins four more keys will need to be generated and exchanged, making a total of 10 keys for five people, figure 2.4(c). As a function of $n$, where $n$ is the number of participants, the number of keys required is $f(n) = (n^2 - n)/2$. Consequently, key exchange using the manual phone/fax or by-hand methods become more impractical as the number of participants grows.
2.1.3 Public-Key Encryption

Public-key encryption, sometimes called asymmetric encryption, uses two different but strongly related keys for its encryption and decryption functions. One key, the private key, is kept secret. The other key, the public key, is publicly distributed. Data encrypted with the public key can only be decrypted with the private key which is useful for general confidentiality protection as well as the exchange of symmetric keys, figure 2.3. Conversely, data encrypted with the private key can only be decrypted
with the public key which is essential in the creation of digital signatures, figure 2.5.

Public-key encryption holds an advantage over symmetric-key encryption when it comes to key distribution. Public keys may be distributed freely and publicly and are available for use by parties that may not have any previous relationship. They do not need to be exchanged in a secure communications channel as is the case for symmetric-key encryption. This feature is what allows banking and e-commerce to take place over the Internet today.

When compared to symmetric-key encryption, present day public-key cyphers are typically computationally more expensive [7]. This limits their practical use to small amounts of data, such as exchanging symmetric keys and digital signatures.

2.2 Encryption Services

Encryption tools such as one-way functions, symmetric-key encryption, and public-key encryption allow for encryption-based services such as confidentiality, data integrity, non-repudiation, and authentication.
2.2.1 Confidentiality

Both symmetric-key encryption and public-key encryption can protect the confidentiality of information while still allowing it to be accessible by authorized parties. When a cryptographic function transforms plaintext into cyphertext it is "computationally infeasible" for even a determined attacker to derive any plaintext from the cyphertext without knowledge of the appropriate key [7]. Authorized parties possessing the proper keys, however, can easily recover the original plaintext.

2.2.2 Data Integrity and Digital Signatures

Data integrity services rely on one-way hash functions and digital signatures. A digital signature is created by
passing data to be signed through a one-way hash function and then encrypting the result with the private key of the signing entity, figure 2.6.

![Diagram of digital signature process]

**Figure 2.7. Verifying a Digital Signature**

The signature is later verified by passing the signed data through same one-way hash function and comparing the result to the value obtained by decrypting the digital signature with the signer's public key, figure 2.7. If the two values match, it is evident that the signed data was not tampered with and that the signing entity participated in the signature. Provided that there is a mechanism in place to properly authenticate that the public key indeed belongs to the signing entity, the digital signature
provides a strong method to verify data integrity. Digital signatures also provide non-repudiation.

2.2.3 Non-Repudiation

Non-repudiation is a subtle but powerful feature of public-key encryption based on the difficulty to forge a digital signature and the privacy of the private key. Provided the private key remains known to only to the signer, non-repudiation prevents the signer of a document from later repudiating or disowning that signature. That is, under public-key encryption, someone can not make a signature and later claim with credibility that they did not make the signature.

Non-repudiation makes the individual creating a digital signature accountable for that signature. Consequently, it allows digital signatures to replace traditional signatures in business processes and, when combined with their difficulty to forge, demonstrates how digital signatures can actually be superior to their traditional counterparts.

In comparison, symmetric-key encryption does not provide non-repudiation. Consider again the scenario proposed in section 2.1.2.1. If Bob receives an encrypted message from Alice, Alice can claim with credibility that
she did not send the message. Since Bob and Alice share the same key (AB), it is possible Bob created the message himself [9].

2.2.4 Authentication

Using certificates for authentication typically involves a trusted third party such as a certificate authority. As discussed in section 3.3, one of the roles of the certificate authority is to validate identities and associate those identities with public keys in the form of digital certificates.

To authenticate that public key belongs to the identity in the certificate, the signature of the certificate must be verified, figure 2.7. The body of the certificate (the signed data) is passed through the one-way hash function specified in the certificate. The result is compared to the value obtained by decrypting the certificate's digital signature with the public key of the signing, trusted certificate authority. If the two values match, the public key can be trusted to belong to the identity in the certificate with a high level of assurance.
CHAPTER THREE
THE ROLE OF A CERTIFICATE AUTHORITY

A certificate authority's primary role is to manage certificate signing requests, a process that requires an interface to accept incoming certificate signing requests, a method to validate identities, and a procedure for signing certificates. Optionally a CA will perform other functions, such as revoke certificates and hold private keys in escrow. These functions are governed by the certificate authority's operating procedures and policies.

3.1 Structure of a Certificate Signing Request

A certificate signing request, or CSR, is a data structure standardized by the company RSA Security in RFC2986. This structure contains "a distinguished name, a public key, and optionally a set of attributes", figure 3.1 [4]. An example of a CSR can be seen in figure 3.2.
The distinguished name typically describes the identity of a computer resource or person. The distinguished name is meant to describe the identity in a globally unique way. For example, although there are many people named James Macdonell in the world, there should only be one James Macdonell working in the Information Security Office of CSU San Bernardino in San Bernardino, California, USA.

The public key is the key the requesting entity wishes to have associated with their identity.
Certificate Request:

Data:
Version: 0 (0x0)
Subject: C=US, ST=California, L=San Bernardino, O=California State University San Bernardino, OU=People, CN=James Macdonell/emailAddress=jmacdone@csusb.edu
Subject Public Key Info:
Public Key Algorithm: rsaEncryption
RSA Public Key: (1024 bit)
   Modulus (1024 bit):
      ... lines deleted ...
      31:79:... lines deleted ...
      31:79:... lines deleted ...
      31:79:... lines deleted ...
      ... lines deleted ...
      31:79:... lines deleted ...
   Exponent: 65537 (0x10001)
Attributes: a0:00
Signature Algorithm: md5WithRSAEncryption
   ... lines deleted ...
78:80

---BEGIN CERTIFICATE REQUEST-----
MIIB/zCCAwCAQAwgb4xCzAJBgNVBAgTA1VTMRMwEQYDVQQIEwpMDQ0NAYDQY
   ... many lines deleted ...
NWDEEnq6q+u3RLavj/POAXU7SytrGJsumKpZBUmGMNvxeIA=
---END CERTIFICATE REQUEST-----

Figure 3.2. Example Certificate Signing Request

The remaining optional attributes may be used to request vendor or application specific extensions be included in the digital certificate. An example optional attribute is dNSName, an attribute describing how the entity is named in the Domain Naming System (DNS). Another example of an optional attribute is the challenge password, which is used by some certificate authorities to allow a requesting entity to revoke a certificate without direct intervention of the certificate authority.
The distinguished name, the public key, and attributes are used by the certificate authority to create an X.509 public-key certificate, a data structure referred to here as a digital certificate.

3.2 Managing Certificate Signing Requests

In order to process certificate signing requests, a certificate authority needs to provide an interface to allow clients of the CA to submit CSRs along with any other verification-related information required by the CA. A certificate authority also needs to provide an interface for the certificate authority administrator to analyze pending certificate signing requests in order to verify the authenticity of the request as well as validate the identity contained within the request. If the CA administrator is able to validate that the newly requested identity belongs to the requesting entity, the administrator may proceed to sign the request. Otherwise, the administrator may reject the request.
3.3 Validating Identities

Digital certificates are used as evidence of an entity's identity and can be used for authentication, authorization and non-repudiation. As the sensitivity and importance of resources protected by digital certificates rise, so does the need to ensure that a certificate belongs to the proper entity.

It is desirable for a trustworthy certificate authority to perform checks as part of its procedures to ensure that the identities on certificates it signs are valid. In order for an identity to be valid, it must be unique and it must belong to the entity who requested the certificate. Failure to verify the identities of all certificates signed by a CA could compromise the trustworthiness of the CA. For example, in 2001 VeriSign issued a certificate used for code signing with the identity "Microsoft Corporation" to a malicious organization that had nothing to do with Microsoft. This created a security threat. The malicious organization could have release malicious code which appeared to have been written and released by Microsoft. If mistakes such as this continue, VeriSign would no longer be trusted as certificate authority. As a result the prompt for running
new software stating "Publisher authenticity verified by VeriSign Commercial Software Publishers CA" becomes meaningless.

Certificates may be verified with varying levels of assurance. Although there is no formal standard, the term "low assurance" commonly describes certificates containing identities that were verified through a simple process such as a password check or email assessment verification. The term "high assurance" describes certificates with identities verified through a vetting procedure which required viewing government certificates, such as driver's license, passport, or require the CA administrator to have personal knowledge of the requesting entity.

3.3.1 Structure of a Certificate

An Internet X.509 public-key certificate, commonly known as a digital certificate, is a data structure standardized in RFC2459 [3]. This structure contains several components: the subject's distinguished name, or DN, the issuer's DN, the subject's public key, a serial number, a notValidBefore date, a notValidAfter date, an optional set of extensions, and a digital signature, figure 3.3. An example of a digital certificate can be seen in figure 3.4. The subject names the identity of the
Figure 3.3. Structure of a Digital Certificate

certificate which is being associated with the public key. The issuer names the certificate authority that signed the certificate. This DN is used when verifying the signature. The issuer DN as well as the serial number are used when checking the certificate's validity against a certificate revocation list. The two dates define the lifetime of the certificate. The extensions contain additional data useful for authentication and authorization schemes. Finally, the
digital signature is used as proof that certificate was issued by the stated issuer.
Figure 3.4. Example Digital Certificate
The core components of the digital certificate are the subject DN, the public key and the digital signature. The digital signature binds the subject identity to the public key, creating an official document of the issuing certificate authority. If an entity using the certificate trusts the issuing certificate authority, it will also trust that the public key indeed belongs to the identity specified in the subject.

3.3.2 Creating Certificates

To create a digital certificate, a certificate authority extracts the requested identity and public key from a certificate signing request and places them into the subject and public key components of a new digital certificate, figure 3.5. The identity may be optionally updated or corrected by the CA administrator before being placed into the digital certificate. The identity of the certificate authority is placed in the issuer component. The *notValidBefore* component is set to the current date and time, and the *notValidAfter* component is set to a date typically 365 to 720 days in the future. Next, a set of extension values are optionally added to the data structure. Finally, a unique serial number is selected for the certificate and added to the data structure.
To create the signature, as described in section 2.2.2 (Page 14), the binary data representing the currently defined components of the certificate are passed through a one-way hash function. This value is encrypted using the private key of the certificate authority. The result is added to the data structure as the digital signature, completing the creation of the digital certificate.
Chapter Four
Minica's Implementation

Minica is intended to be used as a tool to promote the use of digital certificates by providing a user-friendly interface for automating some of the more tedious functions of a certificate authority.

4.1 Managing Certificate Signing Requests with Minica

For small scale deployments, certificate signing requests are submitted to the CA as an email attachment. The CA administrator would then examine the identity contained within the CSR with a tool such as OpenSSL's req tool and would then validate the identity. Should the request meet the requirements of the CA, the CA administrator would proceed to sign the request. Otherwise, the request would be rejected by either replying to the email or by making a phone call. Although an email interface such as this requires almost no new infrastructure to implement, it contains a fair amount of manual tasks to be performed by the CA administrator. Also,
it lacks desirable features such as automatically maintaining a history of certificate signing requests and signed certificates.

There is 1 request pending for jmacdone

1. 2006-03-01 - jmacdone - [ delete ] CN=workhorse.infosec.csusb.edu/OU=Information Security Office/O=California State University San Bernardino

You may submit your CSR one of two ways...

1. Upload the PEM formatted CSR...

2. ... or paste the PEM formatted CSR below

Figure 4.1. MiniCA Certificate Signing Request Submission Interface

The MiniCA interface allows the submission of certificate signing requests through a web-based interface. After authenticating, the requesting entity is presented...
with a web form that allows the requester the option to either select the file containing the CSR to be uploaded to the CA or to paste PEM formatted request text to the form, figure 4.1. After submission, CSR is stored in a database and the requester is presented with a screen showing the status of the request. If the request is accepted, in addition to seeing the status in the interface, the requesting entity is sent an email to verify that the CA received the request and that it is now pending signature. The CA administrator is also sent an email notification that a new certificate signing request is pending signature.

An authenticated requesting entity is given the option to cancel an erroneous request. The creation and submission of certificate signing requests is an infrequent task for most clients of a certificate authority. Giving the client the option to cancel the request allows him or her to remove erroneous requests from the certificate authority without involving the CA administrator. This should ease the burden on the CA administrator while providing an interactive and error-tolerant environment for the client.
The MiniCA interface performs checks to automate other tasks of the CA administrator. Besides ensuring that clients only submit properly formatted CSRs to the CA, the interface collects attributes of the identity (the username and email address) used to authenticate the requesting entity. This data is useful in verifying the identity contained within the CSR.

4.2 Validating Identities with MiniCA

The MiniCA interface for the CA administrator lists the details of pending certificates, namely the authentication information of the requester and the requested identity. It is then up to the CA administrator to manually validate the identity through its established vetting procedures.

For each pending request, the CA administrator is given the option to reject the request or proceed to either sign the request as is or to sign the request after modifications. Giving the CA administrator the option to modify a request before signing it is a somewhat unique feature of MiniCA. It is considered by some a best practice for the CA to only sign requests that contain
error-free identities conforming to all the standards and policies of the CA. In practice, this typically results in situations where a requesting entity will need to submit several CSRs and the CA administrator will need to reject several CSRs because of typographical errors, misspellings, and inconsistent naming. For example, the stateOrProvince component of a distinguished name needs to be spelled out (e.g. "California") in order to comply with the ITU-T X.501 standard. However, it is all too common for requesting entities to abbreviate the value (e.g. "CA" or "Calif."). If the CA administrator has an option to modify the request, that individual can simply correct the stateOrProvince information. Otherwise, the CA administrator is forced to reject the CSR, compelling the requesting entity to submit another CSR with the minor correction. This adds time, tediousness, and complexity to the role of requesting entity which may discourage them from utilizing the certificate authority.

Regardless of whether the CSR is signed or rejected, the requesting entity is sent an email updating them on the status of their request.

Running a local certificate authority, as MiniCA is intended to be run, holds an advantage in the validation
process. The validation is inherently more meaningful because the local certificate authority is closer to the knowledge necessary to properly validate an identity as compared to a global certificate authority, such as VeriSign™ or GeoTrust™. Trust of the local certificate authority is also more explicit. Entities using certificates signed by the local certificate authority must take an active role and choose to install and trust the root certificate of the local certificate authority. This has an advantage of improved security in that it raises awareness of the usage and purpose of digital certificates. Conversely, the explicit trust creates some disadvantage. Clients need extra configuration, and corrective measures may be unclear for clients receiving errors while using digital certificates signed under a local certificate authority for the first time.

4.3 Creating Certificates with MiniCA

Once a requesting entity submits a certificate signing request and the identity of the requested certificate is verified, the CA administrator may proceed with signing the certificate. From the MiniCA interface, figure 4.2, a CA
There is 1 request pending.

workhorse.infosec.csusb.edu requested by jmacdone <jmacdone@csusb.edu>

Subject

- Use requested subject DN
- Override requested Subject DN

<table>
<thead>
<tr>
<th>cn:</th>
<th>workhorse.infosec.csusb.edu</th>
</tr>
</thead>
<tbody>
<tr>
<td>ou:</td>
<td>Information Security Office</td>
</tr>
<tr>
<td>o:</td>
<td>California State University San Bernardino</td>
</tr>
<tr>
<td>l:</td>
<td>San Bernardino</td>
</tr>
<tr>
<td>st:</td>
<td>California</td>
</tr>
<tr>
<td>c:</td>
<td>US</td>
</tr>
<tr>
<td>e:</td>
<td><a href="mailto:admin@infosec.csusb.edu">admin@infosec.csusb.edu</a></td>
</tr>
</tbody>
</table>

subjectAltName: DNS:workhorse.infosec.csusb.edu

Days valid: 730

Optional additional extensions

<table>
<thead>
<tr>
<th>Certificate Authority</th>
<th>Key Options</th>
<th>Certificate Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td></td>
<td></td>
</tr>
<tr>
<td>true</td>
<td></td>
<td></td>
</tr>
<tr>
<td>leave undefined</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Servers/Clients</th>
<th>CAs</th>
<th>Clients</th>
<th>Servers</th>
<th>CAs</th>
</tr>
</thead>
<tbody>
<tr>
<td>nonRepudiation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>digitalSignature</td>
<td></td>
<td>keyCertSign</td>
<td></td>
<td></td>
</tr>
<tr>
<td>keyEncipherment</td>
<td>cRLSign</td>
<td>client email</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>email</td>
<td></td>
<td>objCA</td>
</tr>
</tbody>
</table>

Signing Options

- CN=CSU San Bernardino SSL CA/OU=Information Security Office/O=California State University San Bernardino
- Passphrase: [ ] Sign Selected

Rejection Options

- [ ] Reject Selected

Figure 4.2. MiniCA Certificate Signing Interface
administrator selects one or more certificate signing requests to be signed, optionally modifies what will become the subject of their respective certificates, selects a root or subsidiary certificate to sign the certificates under, enters the appropriate passphrase for the private key of that certificate and selects the “Sign” button.

From: CSU San Bernardino CA <ca@infosec.csusb.edu>
To: James Macdonell <jmacdone@csusb.edu>
Cc: CSU San Bernardino CA <ca@infosec.csusb.edu>
Subject: Certificate request signed

James Macdonell,

Your certificate has been signed by the CSU San Bernardino CA

CN=James Macdonell/OU=People/O=CSU San Bernardino
https://ca.infosec.csusb.edu/minica/cgi-bin/view-cert.pl?issuer=78;serial=0x2

If you have questions regarding the status of the request please contact CSU San Bernardino CA <ca@infosec.csusb.edu>

Figure 4.3. Example Email Notification

If the signature finishes successfully, the new certificate is added to the database and the requester is sent an email notification containing a link back to the MiniCA where the client may retrieve the signed certificate, figure 4.3. If an error occurs (e.g. the CA
administrator provided an incorrect passphrase for the private key of the signing certificate), the database is left unmodified and the MiniCA interface then presents the CA administrator with an error code.
5.1 Introduction

5.1.1 Purpose

MiniCA will provide a software interface to automate and enforce the technical and business processes related to a Certificate Authority, namely the signing of Digital Certificates.

To provide a clear and concise "Software Requirement Specification" for the MiniCA software project, this document will: (1) define the set of desired functionality for the software product, (2) provide a basis for validation and verification of the software product, and (3) serve as a basis for enhancement.

The audience for whom the software project is intended includes:

Technical staff requiring Certificates for use in SSL applications (i.e. a secure web-site) at California State University, San Bernardino.
5.1.2 Scope

MiniCA is intended to be used at small to medium sized institutions with modest resources for supporting a Certificate Authority. Its functionality will focus strongly towards providing certificates to administrators of common public-key encryption technologies, such as secure web sites. As such, MiniCA will provide a convenient interface for web masters and system administrators to submit requests for digital certificates in the form of Certificate Signing Requests (CSRs). In compliment, MiniCA will also provide a convenient interface for Certificate Authority administrators to manage the signing (issuing), revoking and renewing of digital certificates.

An interface to allow people to request and renew digital certificates identifying themselves, so called client certificates, is intended to be implemented not in this iteration, but in future development iterations. However, architecture of the software and data stores will be developed with this feature in mind.
5.1.3 Definitions, Acronyms, and Abbreviations

CA or Certificate Authority

All of the systems, policies, procedures and personnel used for the signing and administration of digital certificates. A Certificate Authority provides the basis for the PKI trust model. Certificate Authorities typically take the role of a "trusted third-party" that issues and verifies identities at varying levels of assurance.

Certificate Chain

A list of digital certificates linked by digital signatures. Certificate chains are used to establish the relationship between a certificate, Intermediate Certificate Authorities and a trusted Certificate Authority.

Certificate Signing Request or CSR

A CSR is a standardized data object containing 1) a distinguished name and 2) a public key. A CSR is typically submitted to a Certificate Authority, which, pursuant to its policies and procedures, will sign the request thereby creating a digital certificate.
Cryptographic Key

Required data given as a parameter to a encryption (or decryption) algorithm when translating plain-text to cipher-text (or vise-versa). The level of privacy protection for a given cipher-text is often directly related to the strength of the cryptographic key.

Digital Certificate

A Digital Certificate is a standardized data object containing 1) a distinguished name 2) a public key and 3) a cryptographic signature. The cryptographic signature binds the public key to the identity. Any entity possessing the private key corresponding to the public key of the certificate can be trusted to be the identity specified by the distinguished name with a level of assurance derived from the trust relationship with the signing Certificate Authority.
**Digital Signature, Cryptographic Signature**

The digital signature is used as a means of authenticating a designated piece of information based on both the information and the private key of the signing entity. A signer typically creates a digital signature passing by the designated information through a cryptographic digest (one-way hash) function and encrypting the result of the digest with a signer's private key. The digital signature can then be later verified by passing the designated information to the same cryptographic digest function and comparing the result to the digital signature decrypted using the signer's public key.

**HTTP**


**HTTPS**

HTTP over SSL/TLS

**Man-in-the-middle attack**

An attack against a encrypted network connection such as an SSL/TLS connection that allows an attacker to compromise both the confidentiality and integrity information being passed.
MySQL
MySQL is the most popular Open Source SQL database management system.

Nested Certificates
Certificates belonging to a Certificate Chain.

OpenSSL
A suite of libraries and executables for SSL. Available: http://www.openssl.org

Perl
Perl is an acronym for "Practical Extraction and Report Language" It is an interpreted language that is optimized for string manipulation, I/O, and system tasks. It is popular for use in CGI programs.

Personally Identifiable Information
Information can be used to uniquely identify an individual, such as combinations of name, date of birth, and social security number. Should personally identifiable information be accessed by an unauthorized party, under California law all individuals listed in the compromised data store must be notified of the compromise which can be costly both monetarily and reputation wise.
Phishing

A social engineering technique, typically involving impersonation of a trusted identity in an effort to obtain sensitive information such as passwords and bank account numbers.

PKI

A public-key infrastructure (PKI) consists of protocols, services, and standards supporting applications of public-key cryptography, such as SSL.

Public-Key Encryption

A set of algorithms and data handling procedures for creating and utilizing public and private keys to encrypt data. For any valid key pair, anything encrypted with the public key can only be decrypted using the corresponding private key. Anything encrypted with private key can only be decrypted with the public key. Public keys are freely distributed; private keys are kept as private as possible.
RA or Registration Authority

A Registration Authority is a typical component of a Certificate Authority. A Registration Authority performs certain tasks delegated by Certificate Authority, such as preparing Certificate Signing Requests by performing evidence of identity checks, to help minimize the exposure of the Certificate Authority.

SSL /TLS

Secure Socket Layer -- The SSL (Secure Sockets Layer) Handshake Protocol was developed by Netscape Communications Corporation to provide security and privacy over the Internet. The SSL protocol maintains the security and integrity of the transmission channel by using encryption, authentication and message authentication codes. SSL was renamed to TLS (Transport Layer Security) after SSL version 3.

Trusted Certificates

A Certificate is considered "Trusted" if 1) it has been added by a user or application developer to an SSL application's trusted certificate database or 2) its Certificate Chain contains a trusted Certificate.
Two-factor Authentication

A procedure for verifying an identity in two independent ways. Common factors include: "something you know" - a password, "something you have" - a certificate or token, and "something you are" - a thumbprint or other biometric.

Valid Digital Certificate

A Certificate is considered "Valid" if 1) Its period of validity has begun and has not expired 2) It is part of a Certificate Chain that contains a trusted Certificate. 3) It has not been revoked

5.1.4 References

Please see page 67.

5.1.5 Overview

The remainder of this document defines the functions and specific requirements of MiniCA in a format consistent with the IEEE Std 830-1998 SRS format.
5.2 Overall Description

5.2.1 Product Perspective

5.2.1.1 System Interfaces

This section will list the software's system interfaces and identify the software functionality necessary to create the interfaces.

5.2.1.2 User Interfaces

Users will interact with the software through a CGI generated interface accessed via a web browser.

5.2.1.2.1 Home Page

The home page will offer links to the features common to all roles.

5.2.1.2.2 Authentication Interface

These pages will prompt the user for credentials for functions requiring authentication to authorize use.

5.2.1.2.3 Submit Certificate Signing Request Interface

These pages will provide the interface for submitting certificate signing requests to the Certificate Authority.

5.2.1.2.4 View Certificates Interface

These pages will dynamically list and individually display the certificates signed under the Certificate Authority.
5.2.1.2.5 Revoked Certificate Interface

These pages will display a dynamically generated list of Certificates revoked by the Certificate Authority.

5.2.1.2.6 Documentation Interface

These pages will provide access to the on-line user documentation, such as a user's guide, FAQ, etc.

5.2.1.2.7 Certificate Signing Interface

These pages will allow the administrator role to sign or reject pending certificate signing requests.

5.2.1.2.8 Certificate Maintenance Interface

These pages will allow the administrator to add previously signed certificates to the on-line certificate store, add or remove the private key associated with a certificate, or revoke a certificate.

5.2.1.2.9 System Configuration Interface

These pages will allow the administrator to configure options such as LDAP authentication and SMTP notification.

5.2.1.3 Hardware Interfaces

The software will rely on the host operating system and existing software to directly interface with the hardware.
5.2.1.4 Software Interfaces

The software will require an SSL-enabled web server, and SQL database system, and Perl 5.8 or greater with the CGI and DBI (DataBase Interface) packages installed and a DBD (DataBase Driver) package appropriate for the SQL database system. Perl LDAP packages will need to be installed for authentication. The software will be developed with Apache 2.x compiles with mod_ssl, and MySQL 3.x as the SQL database system.

The web server will deliver the Perl CGI generated user interfaces to the user's web browser. The SQL database system and related Perl packages will be used to track and archive pending, valid, and revoked Certificates. The web browser will also be used to provide authentication and authorization when necessary.

An SSL-enabled web browser will be required to utilize the User Interface.

5.2.1.5 Memory Constrains

The software will need to run on a single contemporary of-the-self computer and will function with a minimum of 256MB of RAM and 20GB of secondary storage.
5.2.1.6 Operations

Backup and recovery operations will be delegated to the Systems Administrator of the host operating system.

5.2.1.7 Site Adaptation Requirements

An LDAP server needs to be populated with uid, password, and email address to allow for fully functional authentication and SMTP notification interfaces. Some sites may need to generate a temporary self-signed certificate to configure the web server for SSL, as a bootstrapping measure, should no existing PKI exist.

5.2.1.8 Communications Interfaces

The software will rely upon the Software Interfaces to create and manage any network connections.
5.2.2 Product Functions

5.2.3 User Characteristics

5.2.3.1 Client

Users under this role will have access to lists of Certificates as well as the Documentation provided for the PKI. Example users include the clients of SSL enabled web sites.

5.2.3.2 Service Provider

Users under this role will have access to features provided to the Service User role. In addition, they will have the ability to Submit Certificate Signing Requests and request the status of those requests. Example users include
the system administrators of SSL enabled web sites. Users under this role must authenticate to access the Submit CSR and Request Status features.

5.2.3.3 Administrator

Users under this role will have access to features provided to the Server Provider role. In addition, they will have the ability to Sign Certificate Signing Requests and Maintain the status of Certificates in the database. Users under this role must authenticate to access the Certificate Signing and Certificate Maintenance features.

5.2.4 Constraints

5.2.4.1 RFC 2459

The software should provide functionality to implement the policies suggested in this standards document. Key examples include the ability to create a chain of Certificate Authorities and the ability to assign appropriate x509v3 extensions to signed certificates such as 'cA:TRUE' to certificates to be used by Certificate Authorities.

5.2.4.2 Higher Education PKI-Lite

This document prioritizes the implementation of policies suggested in RFC 2459 for Higher Education institutions.
5.2.4.3 Assumptions and Dependencies

5.2.4.4 Root Certificates

Before any other certificates can be signed, a self-signed Root Certificate with its corresponding private key must first be added to the Certificate Authority via the Certificate Maintenance feature. Although the CA will not provide this functionality in this version, documentation on how to create a Root Certificate using the command line features of OpenSSL should be referenced.

5.2.4.5 LDAP

A pre-populated LDAP directory will be required in order to authenticate distinguish users with access to the Service Provider and Administrator roles.

5.3 Specific Requirements

5.3.1 External Interfaces

5.3.1.1 User Interface

5.3.1.1.1 Documentation

A set of HTML pages describing the steps necessary to utilize the functionality of the CA. Topics for documentation include: creating a CSR, submitting a CSR, checking the status of a CSR, installing a signed...
certificate with a certificate chain, and installing the root certificate.

5.3.1.1.2 View Certificates

View Certificates

Certificates signed by: CSU San Bernardino SSL CA

Choose View to view the details of each certificate.

Subjects marked with line-through have been revoked.

<table>
<thead>
<tr>
<th>View</th>
<th>03 blackboard.csusb.edu, Data Center</th>
<th>Expires in 339 days</th>
</tr>
</thead>
<tbody>
<tr>
<td>View</td>
<td>02 ncvpn.csusb.edu, Telecommunications and Network Services</td>
<td>Expires in 324 days</td>
</tr>
<tr>
<td>View</td>
<td>01 ncvpn.csusb.edu, Telecommunications and Network Services</td>
<td>Expires in 318 days</td>
</tr>
<tr>
<td>View</td>
<td>00 ca.infosec.csusb.edu, Information Security Office</td>
<td>Expires in 318 days</td>
</tr>
</tbody>
</table>

Figure 5.2. View Certificates Interface (1 of 2)
<table>
<thead>
<tr>
<th>View Certificates</th>
</tr>
</thead>
<tbody>
<tr>
<td>• CSU San Bernardino Root CA</td>
</tr>
<tr>
<td>○ CSU San Bernardino Intermediate CA</td>
</tr>
<tr>
<td>■ CSU San Bernardino SSI CA</td>
</tr>
<tr>
<td>■ blackboard.csusb.edu 0x00 0x04</td>
</tr>
<tr>
<td>■ <a href="http://www.infosec.csusb.edu">www.infosec.csusb.edu</a> 0x03</td>
</tr>
<tr>
<td>■ ncvpn.csusb.edu 0x02</td>
</tr>
<tr>
<td>■ mail.csusb.edu 0x01</td>
</tr>
<tr>
<td>■ CSU San Bernardino Common Identity CA</td>
</tr>
</tbody>
</table>

Figure 5.3. View Certificates Interface (2 of 2)
View Certificate Details

Subject

C = US, ST = California, L = San Bernardino, O = California State University San Bernardino, OU = Information Security Office, CN = ca.infosec.csusb.edu, emailAddress = admin@infosec.csusb.edu

Issuer

C = US, ST = California, L = San Bernardino, O = California State University San Bernardino, OU = Information Security Office, CN = CSU San Bernardino SSL CA, emailAddress = ca@infosec.csusb.edu

Attributes

- Serial: 0
- Not Valid Before: 2005-06-21 GMT
- Not Valid After: 2006-06-21 GMT
- Expires in: 318 days
- CA: FALSE
- Processed by: James Macdonell

PEM formatted certificate

----BEGIN CERTIFICATE----
MIIEsjCCBBugAwIBAgIBADANBgkqhkiG...more data...
---more data...

----END CERTIFICATE----

Figure 5.4. View Certificate Details Interface
Submit Certificate Signing Request

**Submit CSR**

In order to use this resource, valid credentials must be provided.

**Login**

- Username
- Password
- Authenticate

*Figure 5.5. Authentication Interface*
Submit a CSR

You have one request pending:

1. cn=photos.csusb.edu,ou=ACM,o=CSU San Bernardino (Cancel)

You may submit your PEM formatted certificate request one of two ways...

Upload the request file

Browse...

Paste the request below help

Submit

Figure 5.6. Submit a Certificate Signing Request Interface
5.3.1.1.4 Certificate Signing

Certificate Signing

Preparing to sign...

CN = newservice.infosec.csusb.edu, OU = Information Security Office, O = California State University San Bernardino, L = San Bernardino, ST = California, C = US

Requested 2004-11-16 23:47:39 GMT by jmacdone

Select Signing CA: CSU San Bernardino SSL CA
Passphrase: ____________________________
Valid for: 365 days
CN = newservice.csusb.edu
OU = Information Security Office
O = California State University San Bernardino
L = San Bernardino
ST = California
C = US
emailAddress = ____________________________
subjAltName = ____________________________
CA Certificate = [True | False]
Create Certificate | Abort

Figure 5.7. Certificate Signing Interface
5.3.2 Functional Requirements

5.3.2.1 View Certificates

The system shall provide an interface to list the status of all certificates signed by the CA. It also shall allow the details of individual certificates to be displayed and allow individual certificates to be retrieved in a format usable by other applications.

Figure 5.8. View Certificates Sequence Diagram
5.3.2.2 Submit Certificate Signing Request

The system shall allow the Service Provider role, after authentication, to submit PEM formatted Certificate Signing Requests via a CGI form submission. Improperly formatted submissions will result in an error.

5.3.2.3 Request Status

The system shall provide an interface to allow the Service Provider role, after authentication, to view the status of their CSR submissions.
5.3.2.4 Certificate Signing

The system shall provide interface to allow the Administrator role, after authentication, to sign or reject Certificate Signing Requests.
5.3.2.5 Maintain Certificates

The system shall provide an interface to allow the Administrator role, after authentication, to add existing certificates to the Certificate Authority as well as store and remove the private keys for individual certificates.
5.3.3 Performance Requirements

5.3.3.1 External Interfaces

Users should be notified if an interactive function may take longer than 7 seconds to respond.

5.3.4 Logical Database Requirements

The system will require an SQL Database System and a driver compatible with the Perl DBI package such as DBD::mysql.
5.3.5 Design Constraints

5.3.5.1 Standards Compliance

5.3.5.1.1 RFC 2459

"This memo profiles the X.509 v3 certificate and X.509 v2 CRL for use in the Internet."

5.3.5.1.1.1 X.509 v3 Certificates

Certificate Authorities should produce and handle X.509 v3 certificates. An inventory of the standard fields, standard extensions, and their usage is provided

5.3.5.1.1.2 X.509 CRL v2

"CA [should] periodically [issue] a signed data structure called a certificate revocation list (CRL)"
5.3.5.1.1.3 Certification Authority

"Provision is needed for a variety of different means of certificate and CRL delivery, including distribution procedures based on LDAP, HTTP, FTP, and X.500."

Certificate Authorities need to support a standard set of algorithms used in public-key cryptography.

5.3.5.1.2 Higher Education PKI-Lite

This document is intended to provide a standard for higher education institutions for implementing a public-key infrastructure.

"HE-PKI-Lite is the deployment of PKI technology using existing standard campus mechanisms for identifying individuals affiliated with the institution and for securing systems."

- "HE-PKI-Lite Certification Authorities (CAs) are not required to be able to revoke certificates."

- "Operators... must understand the significance of the CA's private key(s) and take action to protect the key(s) appropriately."

- PKI-Lite certificates may be used for digital signatures and key encipherment.
REFERENCES


