SSL Certificates HOWTO

MartinFranck

franck (at) sopac.org

도운이

박양배

godyang (at) kldp.org

SSL Certificates HOWTO

지은이 MartinFranck

도운이 박양배

인증 기관(Certificate Authority, CA)을 관리하는 방법과, 그 외 웹보안, e-mail보안, 코드 사인 등에서 SSL을 이용한 인증서의 활용 방안을 다룬다. (역주:아직 문서 버전이 낮아서 미완성 부분이 상당히 남아있다. 국승수님께서 작성하신 "Certificate Server의 설치 와 Client인증"이라는 문서가 매우 잘되어 있으므로 그 문서를 참조하면서 보시면 된다.)

고친 과정

고침 v0.1-kr-2 2002-03-18 고친이 godyang

고침 v0.1-kr 2002-03-12 고친이 godyang

고침 v0.1 2001-11-18 고친이 fm

차례

1장. 서론

1.1. 문서의 역사

V0.1-kr-1 - 박양배 <godyang (at) kldp.org >

V0.1-kr 문서의 잘못된 태그를 수정했다.

V0.1-kr - 박양배 <godyang (at) kldp.org>

V0.1을 기준으로 번역했다.

V0.1 - Franck Martin <franck (at) sopac.org>

SSL Certificate HOWTO를 처음 제작했다.

1.2. 소개

이 글을 읽는 사람이라면 필자처럼 OpenSSL (http://www.openssl.org) 프로젝트에 있는 문서들을 달달 읽었을 것으로 예상된다. 하지만 그 곳의 내용만으로는 무엇부터 시작해야되는지, 인증서를 어떻게 관리해야할 것인지 알아내기 힘들었을 것이다. 이 문서는 그러한 독자들의 질문에 대해 해답을 줄 것이다.

이 HOWTO는 Linux외의 어플리케이션들도 다룰 것이다. 모든 어플리케이션을 다루는 것은 힘들겠지만, 혹시 추가할 사항이나 수정할 사항이 있다면 franck at sopac.org (mailto: franck at sopac.org)로 메일을 보내주시기 바란다.

1.2.1. 책임의 한계와 라이센스

이 문서는 자유롭게 배포될 수 있으나 어떠한 보증도 없다.; 필자는 시장성이나 그와 관련된 어떠한 보증도 제공하지 않는다.

간단하게 말해서, 이 문서를 참조하다가 상용 어플리케이션의 보안이 망가졌다면 그건 전적으로 그 사람의 책임인 것이다. 필자의 잘못이 아닌 것이다.

(역주:저작권에 관련된 문구는 원문으로 남겨둔다.)

1. Send your derivative work (in the most suitable format such as sgml) to the LDP (Linux Documentation Project) or the like for posting on the Internet. If not the LDP, then let the LDP know where it is available.
2. License the derivative work with this same license or use GPL. Include a copyright notice and at least a pointer to the license used.
3. Give due credit to previous authors and major contributors. If you're considering making a derived work other than a translation, it's requested that you discuss your plans with the current maintainer.

만약 이 문서를 인쇄하게 된다면 샘플을 저자에게 보내주시기 바란다. 리뷰 용도로 사용할 것이다. ;-)

1.2.2. 선행 지식

서론에서 말했다시피 이 문서는 HOWTO이며, 독자들에게 이 문서를 읽기 전에 OpenSSL man-page를 비롯한 보안 관련 책들을 읽어볼 것을 권한다. 그런 다음 이 문서를 읽으면 보다 효율적일 것이다. 인증서(Certificate)를 사용함으로써 사무 처리에 대한 보안을 강화시킬 수 있으며, 그렇게 만들어진 보안이 어떻게 유지되는지 알게 될 것이다. 또한 OpenSSL이 제공하지 않는 보안에 대해서도 이해할 수 있을 것이다.

1.3. SSL이란 무엇이며 인증서(Certificate)란 무엇인가?

SSL(Secure Socket Layer) 프로토콜은 처음에 Netscape사에서 웹서버와 브라우저 사이의 보안을 위해 만들었다. SSL은 Certificate Authority(CA)라 불리는 서드 파티로부터 서버와 클라이언트의 인증을 하는데 사용된다. 아래는 SSL이 어떻게 작동하는지에 대한 간단한 과정을 설명한 것이다.

1. [웹브라우저] SSL로 암호화된 페이지를 요청하게 된다. (일반적으로 https://가 사용된다)
2. [웹서버] Public Key를 인증서와 함께 전송한다.
3. [웹브라우저] 인증서가 자신이 신용있다고 판단한 CA(일반적으로 trusted root CA라고 불림)로부터 서명된 것인지 확인한다. (역주:Internet Explorer나 Netscape와 같은 웹브라우저에는 이미 Verisign, Thawte와 같은 널리 알려진 root CA의 인증서가 설치되어 있다) 또한 날짜가 유효한지, 그리고 인증서가 접속하려는 사이트와 관련되어 있는지 확인한다.
4. [웹브라우저] Public Key를 사용해서 랜덤 대칭 암호화키(Random symmetric encryption key)를 비롯한 URL, http 데이터들을 암호화해서 전송한다.
5. [웹서버] Private Key를 이용해서 랜덤 대칭 암호화키와 URL, http 데이터를 복호화한다.
6. [웹서버] 요청받은 URL에 대한 응답을 웹브라우저로부터 받은 랜덤 대칭 암호화키를 이용하여 암호화해서 브라우저로 전송한다.

7. [웹브라우저] 대칭 키를 이용해서 http 데이터와 html문서를 복호화하고, 화면에 정보를 뿌려준다.

여기서 간단한 개념 정도는 알아두어야 할 것이다.

1.3.1. 개인키/공개키(Private Key/Public Key):

Private key/Public Key를 이용한 암호화는 하나의 키로 암호화하고 나머지 다른 하나로 복호화할 수 있도록 되어있다. 이해하기 어렵겠지만 필자를 믿어라. (역주:독자가 PKI(Public Key Infrastructure)에 대해서 잘 모른다면 이에 대한 간단한 문서를 읽어보기를 권한다) 앞에서 암호화한 키로만 암호화를 할 수 있는 것이 아니라 반대 방향으로 복호화한 키로도 암호화할 수도 있다.(당연히 앞에서 암호화한 키가 이번엔 복호화하는 키가 되는 것이다) 이러한 키쌍은 소수(prime number)로부터 생성되며, 그 길이(Bit 단위)는 암호화의 강도를 나타낸다.

Private Key/Public Key는 이러한 키쌍을 관리하는 방법이다. 한개의 키는 안전한 장소에 자기만 알 수 있도록 보관하고(Private Key) 다른 하나는 모든 사람에게 퍼뜨리는 것(Public Key)이다. 그렇게 하면 그 사람들이 당신에게 메일을 보낼 때 암호화해서 보낼 수 있으며, 당신만이 그 암호를 풀 수 있다. 반대로 다른 사람에게 메일을 보낼 일이 있으면 Private Key를 이용해 암호화 할 수 있다. 그러면 그 사람들은 복호화해서 볼 수 있다. 그러나 이 방법은 모든 사람이 Public Key를 가지고 있어야 하기 때문에 그리 안전한 방법이 아니다. (역주:오히려 역효과를 일으킬 수 있다. 단순하게 메시지, 파일에 암호화, 서명을 할 생각이면 PGP나 GnuPG를 권한다)

1.3.2. 인증서(Certificate):

당신과 접속해있는 사람이나 웹 사이트가 믿을 수 있는지 어떻게 판단할 수 있을까? 한 웹사이트 관리자가 있다고 가정하자. 그 사람이 당신에게 이 사이트가 믿을만하다고 (심각할 정도로) 열심히 설명했다. 당신이 그 사이트의 인증서를 설치해 주기를 바라면서 말이다. 한두번도 아니고 매번 이렇게 해야한다면 귀찮지 않겠는가?

인증서는 여러 부분으로 이루어져있다. 아래는 인증서 속에 들어있는 정보의 종류를 나타낸 것이다.

1. 인증서 소유자의 e-mail 주소
2. 소유자의 이름
3. 인증서의 용도
4. 인증서 유효기간
5. 발행 장소
6. Distinguished Name (DN)

   - Common Name (CN)

   - 인증서 정보에 대해 서명한 사람의 디지털 ID

7. Public Key
8. 해쉬(Hash)

SSL의 기본 구조는 당신이 인증서를 서명한 사람을 신뢰한다면, 서명된 인증서도 신뢰할 수 있다는 것이다. 이것은 마치 트리(Tree)와 같은 구조를 이루면서 인증서끼리 서명하게 된다. 그러면 최상위 인증서는? 이 인증서를 발행한 기관을 Root Certification Authority(줄여서 Root CA)라고 부르며, 유명한 인증 기관(역주:Verisign, Thawte, Entrust, etc)의 Root CA 인증서는 웹 브라우저에 기본적으로 설치되어 있다. 이러한 인증 기관은 자신들이 서명한 인증서들을 관리할 뿐만 아니라 철회 인증서(Revoked Certificate)들도 관리하고 있다. 그러면 Root CA의 인증서는 누가 서명을 했을까? 모든 Root CA 인증서는 자체 서명(Self Signed)되어 있다.

1.3.3. 대칭키(The Symmetric key):

Private Key/Public Key 알고리즘은 정말 대단한 알고리즘이지만, 비실용적이라는 단점이 있다. 비대칭(Asymmetric)이란 하나의 키로 암호화를 하면 다른 키가 있어야 복호화를 할 수 있는 것을 뜻한다. 즉, 하나의 키로 암호화/복호화를 할 수 없다는 말이다. 대칭키(Symmetric Key)는 하나의 키로 암호화/복호화를 하게 된다. 대칭키를 사용하면 비대칭키보다 훨씬 빠르게 암호화/복호화를 할 수 있다. 그렇지만 속도 때문에 대칭키를 이용한다는 것은 너무 위험하다.

만약 당신의 적이 키를 입수해 버리면 여태까지 암호화된 정보가 모두 무용지물이 되어버리게 된다. 그래서 대칭키 알고리즘을 사용한 키를 상대방에게 전송하려면 인터넷과 같은 통로는 너무나도 위험하기 때문에 직접 손으로 전달해야만 한다. 귀찮지 않은가? 해결책은 대칭키를 비대칭키로 암호화시켜서 전송하면 된다. (역주:위에서 살펴본 웹서버와 브라우저의 관계에서 볼 수 있었다) 자신의 Private Key만 안전하게 관리하면 Public Key로 암호화되어 안전하게 전송할 수 있다.(여기서 보안은 죽음이나 협박 등을 제외한다) 또한 대칭키는 매번 랜덤으로 선택되는데, 이렇게되면 만약 대칭키가 누출되어도 다음번에는 다른 키가 사용되기 때문에 안전하다.

1.3.4. 암호화 알고리즘(Encryption Algorithm):

암호화 알고리즘은 대칭이든 비대칭이든 간에 상당히 많은 종류가 있다. 일반적으로 암호화 알고리즘으로 특허를 낼 수 없다. 만약 Henri Paincare가 암호화 알고리즘으로 특허를 낸다면 Albert Einstein으로부터 고소당할 수 있는 것이다. 단, USA에서는 암호화 알고리즘으로 특허를 낼 수 있다.(역주:또한 군사법으로 보호받게 된다.) OpenSSL의 경우에는 암호화 알고리즘으로 특허를 낼 수 없고, 군사법이나 보안법에 저촉되지 않는 국가에서 개발되고 있다.

실제로 알고리즘이 어떻게 사용되는지 살펴보자. 웹서버와 브라우저는 서로 통신을 하는 동안 서로 어떤 알고리즘을 사용할 수 있는지 확인하게 된다. 다음에 서로 이해할 수 있는 일반적인 알고리즘을 선택한 후 통신이 이루어지게 된다. OpenSSL은 컴파일해서 삽입할 알고리즘을 택할 수 있다. 그렇게 하면 암호화 알고리즘에 제한을 걸고 있는 국가에서도 사용할 수 있게 된다.

1.3.5. 해쉬(Hash):

해쉬는 해쉬 함수에 의해 만들어지는 숫자다. 이 함수는 단방향으로 연산되는 함수다. 즉, 한번 해쉬함수로 원본으로부터 해쉬값을 생성했다면, 해쉬값으로부터 원본 메시지를 알아내는 것이 불가능하다. 해쉬의 용도는 원본 메시지가 손상이 되었는지를 알아내는데 있으며, 해쉬값을 변경시키지 않고 원본을 조작하는 것은 극히 어렵다. 그래서 해쉬 함수는 패스워드 메커니즘을 비롯한 소프트웨어의 손상 유무(MD5 sum)를 가리는데도 이용할 수 있으며, 메시지의 손상을 방지하기 위해 널리 쓰이고 있다.

1.3.6. 서명(Signing):

서명은 특정 메시지를 내가 작성했다는 것을 인증하는 역할을 한다. Text가 될 수도 있고, 인증서 등등이 될 수 있다. 메시지에 서명하기 위해서는 아래의 순서를 따라야 한다.

1. 해쉬 생성
2. Private Key로 해쉬 암호화
3. 암호화된 해쉬와 서명된 인증서를 메시지에 추가
4. 받는 사람은 따로 해쉬를 생성
5. 받은 메시지에 포함된 해쉬를 Public Key를 이용해서 복호화
6. 4, 5번 과정에서 생성된 해쉬를 비교

1.3.7. 암호문(Pass Phrase):

암호문(Pass Phrase)은 기존의 패스워드(Password)를 확장한 시스템이다. 예전의 Unix 시스템의 암호(Password)는 8자가 한계였다. 암호문이라는 것은 단순히 암호의 한계가 더 길어졌다는 것을 뜻한다. 당연히 8자가 한계인 것보다 보안이 강력하다. 최근의 Unix 시스템은 MD5를 사용하고 있기 때문에 암호의 길이 제한이 없어졌다.

(역주:암호문은 SSL에서 키가 누출되었을 경우, 최종 보안 장치이기 때문에, 중요한 키에는 반드시 새겨두어야 한다.)

1.4. S/Mime은 무엇을 뜻하며, 그 외 다른 프로토콜은 무엇이 있는가?

SSL이 웹의 기능을 확장하기 위해 생겨났다면, 다른 프로토콜들도 SSL을 사용하여(Encapsulate) 암호화된 프로토콜로 만들 수 있을 것이다. 현재 이렇게 만들어진 프로토콜들이 IMAPS, POPS, SMAPS 등등이 있다. (역주:끝에 S만 추가되었다) 이러한 보안 프로토콜은 기존의 프로토콜과는 다른 포트(Port)를 사용하게 된다. SSL은 이런 프로토콜뿐만 아니라 어

떠한 데이터의 전송에도 접목해서(Encapsulate) 사용할 수 있다. S/Mime은 바로 이러한 역할을 하는 프로토콜이다. 그것을 사용하면 일반 e-mail 메시지를 받는 사람의 Public Key를 이용해서 암호화할 수 있다. 그러므로 S/Mime을 사용하기 전에 반드시 받는 사람의 Public Key를 웹이나 저장소(Repository), e-mail을 통해서 받아야 한다. (e-mail을 사용할 때는 그 사람이 정확한 사람인지 확인하기 위해 인증서와 함께 받는 것이 좋다.)

또한 반대방향으로 클라이언트가 서명된 인증서를 서버쪽으로 보낼 수도 있다. 예컨데 웹브라우저가 웹서버로 인증서를 보낼 수 있다. 이렇게 하려면 모든 사람이 인증 기관 웹사이트에서 브라우저 인증서를 받아야 한다. 그리고 나서 서명된 인증서가 Private Key로 암호화되어 전송되고, Public Key로 복호화된다.

2장. 인증서 관리

2.1. 설치

오늘날 OpenSSL의 설치는 매우 쉽게 되어있다.. 패키지 관리 시스템(역주:Debian은 deb, RedHat은 rpm, FreeBSD는 포트 시스템)을 사용하면 간편하게 설치할 수 있다. 만약 이러한 여건이 안된다면 OpenSSL tarball 파일 속에 있는 README와 INSTALL 파일을 읽어보기를 권한다.

혹시 독자들의 시스템과 필자의 시스템이 틀릴 수 있으므로, 여기서 간단히 필자의 OpenSSL 설치 옵션을 설명하겠다.

모든 OpenSSL 인증서가 있는 디렉토리는 /var/ssl/로 설정하였다. 이 문서의 모든 명령과 경로는 이 디렉토리를 의미한다고 보면 된다. 절대적인 것은 아니지만, 예제를 이해하는데 도움이 될 것이다.

OpenSSL을 가장 기본적인 옵션으로 설치하면 설정 파일이 /usr/lib/ssl/openssl.cnf에 위치하게 된다. 이 경로는 -config옵션으로 바꿀 수 있다. 필자는 /etc/openssl.cnf를 설정 파일로 사용하고 있다. 이 설정 파일은 openssl ca명령이나 openssl req명령에 영향을 미치게 된다.

유틸리티나 다른 라이브러리들은 /usr/lib/ssl에 위치하고 있다.

CA.pl이란 실행 스크립트가 /usr/sbin과 같은 실행 경로에 있는 것을 확인해야 한다. CA.pl은 기본적으로 /usr/lib/ssl 디렉토리에 있다. 이 스크립트는 복잡한 openssl명령을 간편화시켜주는 유틸리티다.

/usr/sbin/CA.pl파일에서 openssl ca와 openssl req가 실행될 때 -config /etc/openssl.cnf 옵션이 포함되도록 수정해야 한다.

```
#$SSLEAY_CONFIG=$ENV{"SSLEAY_CONFIG"}
$SSLEAY_CONFIG="-config /etc/openssl.cnf";
```

/etc/openssl.cnf 또한 수정해서 입력을 최소화해야 한다. (역주:openssl.cnf에는 기본적인 인증기관 Private Key위치, 인증서 위치, Serial Number 등등의 위치가 있다. 이러한 것이 제대로 되어있지 않다면 수정하고, 필요에 따라 policy부분, Netscape Comment 부분도 수정한다. 그 외에 인증서의 용도, 다른 이름(Alt Name)등등도 필요에 따라 수정한다. 아래의 예제는 아직 문서 버전 0.1에는 포함이 되어 있지 않으나, 추후 추가될 것이다)

```
/etc/openssl.cnf example
```

인증 기관(Certification Authority)을 생성하려면 아래와 같이 명령을 내리면 된다:

```
CA.pl -newca
```

2.2. 루트 인증 기관 인증서(Root Certification Authority Certificate) 만들기

```
CA.pl -newcert
(openssl req -config /etc/openssl.cnf-new -x509 -keyout newreq.pem -
out newreq.pem -days 365)
```

자체 서명된 루트 인증 기관의 인증서를 만든다. 인증서와 Private Key는 newreq.pem 에 저장된다. Common Name(CN)은 "ACME root Certificate"와 같은 것을 사용하면 된다. 이 파일에는 인증서와 Private Key가 섞여있기 때문에 쪼개야 한다. ---CERTIFICATE--- 부분은 인증서 부분으로 cacert.pem이란 파일에 저장하고, ---RSA PRIVATE KEY---부분은 Private Key로 cakey.pem에 저장한다. (역주:이 때 반드시 ---BEGIN...---부분과 ---END...--- 부분이 포함되도록 저장해야 한다.) 그 다음에 키 누출을 방지하기 위해 newreq.pem을 지우자.

openssl.cnf에 포함된 database경로에 "index.txt"라는 빈 파일을 만들고, serial경로에 1 이란 숫자로 된 "serial"파일을 생성한다. (역주:아래는 역자가 생성한 방법이다.)

```
# touch /etc/index.txt
# echo "01" > /etc/serial
```

기본적으로 365일 날짜 제한이 되어 있는데, 이렇게 하면 1년이 지나고 나서 더이상 루트 인증 기관의 키와 그것에 의해 서명된 키들을 사용할 수 없다. 이렇게 되는 것을 방지하려면 날짜 제한을 크게 해서 생성하면 된다. 기업 정도의 레벨에서는 약 5년 정도의 기간을 권고한다. (역주:아직 SSL에 대해 완벽히 알지 못한다면 짧은 기간만 설정할 것을 권고한다.)

```
openssl req -config /etc/openssl.cnf-new -x509 -keyout newreq.pem -
out newreq.pem -days 1825
```

이렇게 생성된 루트 인증 기관 키는 반드시 다른 인증서들을 서명하는 곳에만 사용해야 한다. Private Key는 위험에 노출되지 않도록 반드시 안전한 곳에 보관하고, 절대로 암호문(Passphrase)을 제거해서는 안된다.

자. 이제 루트 인증 기관을 만들었다. 다른 사람들이 당신의 루트 인증 기관 인증서가 필요할 수 있다. 그러면 그들에게 다운로드받아서 그들의 브라우저에 설치하라고 알려주면 된다.

다른 인증서를 서명할 때마다 암호문을 입력해야 하는 번거로움이 있지만, Private Key가 누출될 것을 생각해서 귀찮게 여기지 말자.

2.3. 하부 인증 기관 인증서(Non-root Certification Authority Certificate) 만들기

FIXME 필자는 이 작업에 대해 잘 알지 못한다.

루트 인증 기관에 의해 서명된 인증서를, 또다시 다른 인증서를 서명하는데 사용할 수도 있다. 이 때 서명 권한을 넘겨주어야 한다. 만약 이러한 인증서를 사용하려면 인증 요구서(Certificate Request)와 Private Key를 만든 후, 서드 파티에게 인증 요구서를 보내면 된다. 그러면 서드 파티가 서명한 인증서를 받아서 Private Key와 함께 설치하면 된다. 이때도 역시 ---RSA PRIVATE KEY---부분은 cakey.pem과 같은 파일에 저장하고, ---CERTIFICATE---부분은 cacert.pem으로 저장해서 사용하면 된다.

(역주:인증 권한 설정은 openssl.cnf를 이용하면 되며, 이러한 인증서를 만들고 사용하려면 CA.pl과 같은 자동 스크립트가 아닌, openssl.cnf를 비롯한 openssl 명령을 이용한 수동 사용법에 대해 잘 알고 있어야 한다. Verisign과 Thawte등등의 상업 인증 기관에서 서명된 키를 사용하려면 인증 요청서(Certificate Request)를 만든 후, 인증 요청서를 요금과 함께 보내주면 인증서를 받을 수 있다. 조만한 HOWTO에도 이 부분이 자세히 추가될 것으로 예상된다.)

2.4. Root CA Certificate를 Trusted Root Certificate로 설치하기

우선 ---CERTIFICATE---로 된 인증서 부분을 잘라야 한다.

```
openssl x509 -in cacert.pem -out cacert.crt
```

이 파일을 http://mysite.com/ssl/cacert.crt와 같은 경로에 저장한다. 웹서버에 .crt 파일에 대한 MIME을 추가하고, 브라우저로 하여금 하드디스크에 저장하도록 하면 된다.

2.4.1. 넷스케이프(Netscape)

FIXME

2.4.2. 갈레온(Galeon)

FIXME

2.4.3. 인터넷 익스플로어(Internet Explorer)

브라우저에서 인증서가 있는 주소를 치고 디스크에 저장한다. 파일을 더블 클릭하고 설치를 누르면 인증서 설치 마법사(Certificate Installation Wizard)가 시작될 것이다. 루트 인증 기관 인증서는 자체 서명되어 있기 때문에, 인터넷 익스플로어에서 자동적으로 Trusted root Certificate Authority 리스트에 저장하게 된다. 그렇게 설치하면 다음부터 그 루트 인증 기관 인증서로 서명되어 있는 인증서는 자동적으로 신용하게 된다.

2.5. 인증서 관리

2.5.1. 인증 요구서 생성과 서명하기

```
CA.pl -newreq
(openssl req -config /etc/openssl.cnf -new -keyout newreq.pem -
out newreq.pem -days 365)
```

새로운 Private Key와 인증 요구서를 newreq.pem에 저장한다.

```
CA.pl -sign
(openssl ca -config /etc/openssl.cnf -policy policy_anything -
out newcert.pem -infiles newreq.pem)
```

인증 요구서를 아까 만든 루트 인증 기관의 키로 서명한 후, newcert.pem이란 새로운 인증서를 생성한다. 이 때 인증 기관의 암호문(Passphrase)을 넣어야 서명이 될 것이다. 그리고 난 후 newcerts/xx.pem이란 파일이 openssl.cnf에서 지정한 경로에 생기며, index.txt, serial이 자동으로 갱신된다.

Private Key는 newreq.pem 파일에 ---PRIVATE KEY--- 라고 되어 있으며, 인증서는 newcert.pem에 ---CERTIFICATE---라고 되어 있다.

방금 말했다시피 newcerts/란 디렉토리에 newcert.pem파일이 생기며 index.txt가 갱신된다. 클라이언트는 이 정보를 바탕으로 웹서버에게 인증 기관에 대한 정보를 요청할 수 있다.

newreq.pem파일을 조심해서 다루어야 한다. 그 파일에는 인증 요구서 뿐만 아니라, Private Key까지 들어있기 때문이다. Private Key는 서명할 시에 필요가 없으며, 같이 보낼 경우 인증 기관에게 보안이 노출될 수 있다. 그래서 만약 다른 인증 기관에게 인증 요구서를 보낼 때는 반드시 ---PRIVATE KEY--- 부분은 잘라내야 한다. 반대로 당신이 서명을 할 때는 ---CERTIFICATE REQUEST---부분만 보내주도록 요구해야한다.

2.5.2. 인증서 철회하기

인증서를 철회하기 위해서는 아래와 같이 명령하면 된다:

```
openssl ca -revoke newcert.pem
```

그러면 데이터베이스가 업데이트되고, 인증서가 철회되었다고 표시된다. 다음은 인증서의 인증 철회 리스트를 작성해야 한다.

```
openssl ca -gencrl -out crl/crl.pem
```

(역주:MS Windows에서는 .crl이란 확장자를 인증 철회 리스트로 인식한다. 위 pem확장자를 crl으로 바꾸면 MS Windows에서 사용할 수 있다.)

2.5.3. 인증서 갱신하기

날짜 제한이 지나거나 Private Key가 누출되면 더 이상 인증서를 사용할 수 없으므로 새로운 인증서를 발행해야 한다. 이 때 유저들이 인증서를 분실해서 인증 요구서나 Private Key를 보내주면서 철회 요구를 할 수도 있다.

우선 예전의 인증서를 철회하고 인증 요구서를 다시 서명해야 한다.

예전의 인증서를 찾기 위해서는 index.txt 파일에 Distinguished Name (DN) 부분, serial 파일의 Serial Number 부분을 인증 요구서와 함께 비교하면 된다. 그리고 그에 맞는 cert/<xx>.pem 에 저장된 인증서를 인증 철회시에 사용하면 된다.

시작과 만기일을 확실히 하기 위해 수동으로 서명할 수도 있다.

```
openssl ca -config /etc/openssl.cnf -policy policy_anything -
out newcert.pem -infiles newreq.pem -startdate [now] -
enddate [previous enddate+365days]
```

[now] 부분과 [previous enddate+365days] 부분을 정확한 값으로 수정해서 실행하면 된다.

2.5.4. 웹 기반 Certificate Authority 만들기

아래는 인증 기관을 운영할 때 필요한 것이다.

1. 루트 인증 기관의 인증서를 발행해서 응용 프로그램에 설치되도록 해야 한다.
2. 인증 철회 리스트를 발행해야 한다.
3. 인증서의 정보를 Serial Number를 포함해서 자세히 출력되도록 해야 한다.
4. 유저들로 하여금 인증 요구서를 받을 수 있도록 폼(form)을 제공해야 한다.

웹서버와 몇몇 스크립트 언어를 이용하면 위와 같은 것들을 할 수 있다.

2.6. 인터넷 프로토콜 보안

2.6.1. Apache에서 mod_ssl을 이용하여 인증서 사용하기

절대로 루트 인증 기관 인증서를 Apache에서 사용하면 안된다. 왜냐하면 Apache의 Private Key는 암호문(Passphrase)을 제거해야만 사용할 수 있기 때문이다.

우선 인증 요구서를 발행해야 한다. 이 때 반드시 Common Name(CN)을 www.mysite.com과 같은 웹 도메인으로 해야 한다.

다음에 Apache가 정상적으로 작동하기 위해서 암호문을 제거해야 한다. 아래는 Private Key를 포함하고 있는 newreq.pem파일에서 암호문을 제거하는 방법이다.

```
openssl rsa -in newreq.pem -out wwwkeyunsecure.pem
```

Private Key가 제거된 상태이기 때문에 반드시 권한(Permission)등을 철저하게 체크해야 한다. 만약 다른 사람이 Private Key를 얻게 되면 그 사이트는 보안이 망가지게 되며, 그 사람이 악의로 키를 사용할 수 있다. 이 작업이 끝나면 newcert와 wwwkeyunsecure.pem을 Apache에서 사용하면 된다.

wwwkeyunsecure.pem과 newcert.pem을 /etc/httpd/conf/ssl/에 wwwkeyunsecure.pem 과 wwwcert.crt라는 이름으로 복사한다. (역주:이 때 wwwkeyunsecure.pem이 서버 유저에게 노출되지 않도록 퍼미션을 잘 관리한다.)

/etc/httpd/conf/ssl/ssl.default-vhost.conf 파일을 수정한다.

```
----
# Server Certificate:
# Point SSLCertificateFile at a PEM encoded certificate. If
# the certificate is encrypted, then you will be prompted for a
# pass phrase. Note that a kill -HUP will prompt again. A test
# certificate can be generated with 'make certificate' under
# built time.
#SSLCertificateFile conf/ssl/ca.crt
SSLCertificateFile wwwcert.crt
# Server Private Key:
# If the key is not combined with the certificate, use this
# directive to point at the key file.
#SSLCertificateKeyFile conf/ssl/ca.key.unsecure
SSLCertificateKeyFile wwwkeyunsecure.pem
----
```

httpd 프로세스를 죽이고(/etc/rc.d/init.d/httpd stop), 모든 프로세스가 죽었는지 확인한 후(killall httpd), 재시작한다. (etc/rc.d/init.d/httpd start)

(역주:Apache-SSL의 경우 Apache와 별도의 프로세스에서 돌아가며, httpd.conf에 SSL관련 설정이 내장되어 있다.)

2.6.2. IMAPS에서 인증서 사용하기

FIXME

2.6.3. POP에서 인증서 사용하기

FIXME

2.6.4. Postfix에서 인증서 사용하기

FIXME

2.6.5. Microsoft Key Manager로 키 생성과 서명하기

마이크로소프트 키 매니저(Microsoft Key Manager)에서 키를 만들기 위해서는 사용할 서비스(예컨데 IMAP이나 WWW)를 선택해야 한다. 새로운 키를 생성하려면 마법사(Wizard)를 사용하면 된다. 앞에서는 Common Name(CN)을 사이트 주소와 동일하게 적었으나, 여기서는 Distinguished Name을 다르게 적어도 된다. 마법사는 C:\NewKeyRq.txt 이란 파일에 인증 요구서를 저장하게 된다. 키 매니저는 그 키가 서명되지 않았다고 경고할 것이다.

/var/ssl 디렉토리에 그 파일을 받아와서 newreq.pem으로 이름을 바꾸고 서명한다.

```
CA.pl -sign
```

다음에 newcert.pem을 확인해보면 ---CERTIFICATE---부분과 다른 텍스트들이 있는 것을 볼 수 있다. ---CERTIFICATE---부분을 제외한 텍스트를 제거해야 비로소 키 매니저가 인식할 수 있게 된다. 아래와 같이 쉽게 제거할 수 있다.

```
openssl x509 -in newcert.pem -out newcertx509.pem
```

간단하게 텍스트 에디터를 이용해서 지워도 된다. (이 때 ---CERTIFICATE---부분이 지워지지 않도록 주의하자)

이런 작업을 거치면 newcertx509.pem에는 ---CERTIFICATE---부분만 남게 된다.

newcertx509.pem 파일을 키 매니저가 돌아가고 있는 컴퓨터로 전송하고, 마우스 오른쪽 버튼을 누르면 설치할 수 있다. 다음에 암호문을 입력하면 그 키는 완벽하게 작동하게 된다.

2.7. E-mail 보안

2.7.1. S/MIME 인증서의 생성과 사용

단순하게 인증 요구서를 생성해서 서명하면 된다. 그러나 반드시 Common Name (CN)을 사용자의 e-mail 주소로 적어야 한다.

다음에 test.txt라는 메시지를 인증서(newcert.pem)와 Private Key(newreq.pem)로 서명한다. 결과는 test.msg에 저장된다.

```
openssl smime -sign -in test.tx -text -out test.msg -signer newcert.pem -
inkey newreq.pem
```

2.7.2. MS Outlook에서 이러한 인증서 사용하기

우선 Outlook에서 PKCS12 형태의 파일을 수입(Import)해야 한다. PKCS12 파일을 newcert.pem과 newreq.pem으로부터 생성하기 위해서는 아래와 같이 명령하면 된다.

```
CA.pl -pkcs12 "Franck Martin"
(openssl pkcs12 -export -in newcert.pem -inkey newreq.pem -
out newcert.p12 -name "Franck Martin")
```

이 인증서에는 Public Key뿐만 아니라 Private Key도 저장되어 있다는 점을 상기해야 한다.(암호문도 뚫린다면 그야말로 끝장이다) 이 파일이 타인의 손에 넘어가지 않도록 주의하자.

MS Outlook에서 도구(Tools), 옵션(Options), 보안(Security)부분에 가서 수입/수출(import/export) 버튼을 누르고 newcert.p12 파일을 수입한다. 다음에 패스워드와 디지털 ID "Franck Martin"를 입력한다. (디지털 ID는 위의 예에서 입력한 것을 그대로 입력하면 된다.) 그리고 OK를 누르면 된다.

다음에 설정(Setting) 버튼을 누르고 MS Outlook으로 하여금 기본 설정으로 사용할 수 있게끔 New를 클릭한다. 그리고 마지막으로 이 설정을 기본 설정으로 유지하고 싶으면 OK를 누르면 된다. 이렇게 하면 비로소 e-mail을 보낼 준비가 끝나게 된다. 그리고면 다른 사람에게 메일을 보내면 Public Key와 함께 전송되어 그 사람은 당신에게 e-mail을 보낼 때 암호화할 수 있다.

만약 직접 생성한 루트 인증 기관에서 서명한 인증서라면 응용 프로그램에서 루트 인증 기관을 신용할 수 없다고 경고가 나올 수 있다. 루트 인증 기관 인증서를 다운로드받아서 설치하면 이러한 경고가 나오지 않는다. 루트 인증 기관 인증서는 "Root CA Certificate를 Trusted Root Certificate로 설치하기"부분의 "인터넷 익스플로어"부분을 보면 된다.

2.7.3. Outlook Express에서 이러한 인증서 사용하기

FIXME

2.7.4. Galeon에서 이러한 인증서 사용하기

FIXME