# Certificate Server의 설치 와 Client인증

국승수 sskuk@sskuk.cataegu.ac.kr V0.1 2000년 11월 11일

이 문서는 제가 applet사인과 관련하여 이것저것 조사하다가, OpenSSL이라는 것을 알게되서, OpenSSL을 이용한 Certificate Server(rootCA)를 설치하기 까지의 과정을 정리한 것이다. OpenSSL의 설치와, apache, mod_ssl, php, mysql등을 이용하여 Certificate Server를 운용하기 위한 방법론과, Netscape, MS Internet Explorer, JDK등의 Client에 대한 인증서를 발행하는 방법론을 담고 있다.

## 목 차



## 제 1 절 저작권

## 제 2 절 준비물

1. *OpenSSL* <http://www.openssl.org/>
2. *Apache* <http://www.apache.org/>
3. *Perl* <http://www.perl.com/>
4. *mod_ssl* <http://www.modssl.org/>
5. *PHP* <http://www.php.net/>
6. Client Browser(Netscape나 Explorer)
7. Java2 SDK
8. ...
9. 담배 한두갑 과 찐한 커피(필수!)
10. 담배안피면 껌이나 사탕(^^)

OpenSSL만 있어도 Certificate Service를 처리하는데에는 문제가 없지만, 좀더 사용자를 편하게 할려면 나머지 들도 있는것이 좋다.

## 제 3 절 Certificate Server란?

Certificate Server는 인터넷에서 메일이나 프로그램(applet, script같은것)등의 송신자를 믿을만한 사람이 보냈는지 인증해주는 서버이다. 쉽게 말하면 철수가 은행에 개인적인 문서를 Sign을 하여 보냈다고 하면, 은행의 입장에서는 이 메세지가 정말로 철수에게서 온것인지를 확인할 필요가 있다. 이때 PGP나 GPG처럼 비밀키와 공개키를 이용하여 철수는 비밀키로 사인하고 은행에서는 철수의 공개키를 이용하여, 수신된 메세지가 철수가 보낸것이 맞는지 확인할수 있다. 또한 은행에서 철수 한테 보내는 메세지인 경우에도 마찬가지 이지만, 서로간의 공개키가 정확하게 전달이 되었는지 확인할 필요가 있다. (보통 메일이나, 홈페이지등을 통하여 확인할수도 있고, 직접 만나서 finger print등을 적어와서 비교해볼수도 있다.)

만약 서로 상대방의 공개키가 정말로 믿을만한 인증기관에서 인증해준것 이라면, 상대방의 공개키가 맞다는것을 확인하는 수고를 덜수가 있는데, 이 인증기관을 Certificate Authority(CA)라고 부르며, CA에의해서 상대방의 공개키가 인증되어 있으면, 서로간에 직접 확인할 필요없이 공개키를 믿을 수 있게 되는것이다.

(OpenSSL에는 이것뿐만이 아니고 상당히 많은 기능을 가지고 있으나 제가 아는게 별로없네요...^^;) 또 Applet을 작성해 보신분은 아시겠지만 인증서에의한 사인이 되어있는 경우에는 일부 시스템의 제한된 자원(프린터, 디스크, 사운드등)에 접근할수 있는 Applet을 생성할수 있습니다. (다이얼패드를 사용할 때에 나타나는 창을 생각해보세요...^^)

위에서 나타난것 처럼, 인터넷에서 특정개인이나 기관에대한 인증을 해주는 곳이 CA이며, 이런 역활을 수행하는 서버가 Certificate Server이다. *VeriSign* <http://www.verisign.com/>이나 *Thawte Consulting* <http://www.thawte.com/>등에서 이런 서비스를 하고 있고, 기본적으로 Netscape나 MS Internet Explorer등의 신뢰된 인증기관에 나타난다. 본 문서는 VeriSign이나 Thawte Consulting등과 같은 서비스를 OpenSSL을 이용하여 실행할 수 있는 방법에 대하여 기술하고 있다.

OpenSSL은 Secure Sockets Layer와 Transport Layer Security프로토콜을 만족하는 Open Source 툴킷이다. OpenSSL은 Eric A. Young과 Tim J. Hudson에의해 만들어진 SSLeay 라이브러리를 기초로하여 만들어 졌고, 현재 OpenSSL-0.9.6이 사용가능하며, Unix와 Windows, OpenVMS, MacOS등의 플랫폼에서 실행된다.

# 제 4 절 Certificate Server의 설치

Certificate Server의 구성은 *OpenSSL* <http://www.openssl.org/>과 *Apache* <http://www.apache.org/>가 필요하며, 개인의 신상정보가 Web을 이용하여 전송됨으로 Apache 서버에 *mod_ssl* <http://www.modssl.org/>을 설치하거나 Apache + SSL의 형태로 Apache서버를 설치하여 Browser와의 통신시 sniffing되지 않도록 하여야 할 것이다. 또한 자동화된 서비스를 위해서 *Perl* <http://www.perl.com/>이나 *php* <http://www.php.net/>등이 필요하다. (Perl이나 PHP는 module로 설치해도 좋으며 CGI형태로 실행해도 상관이 없으나 보안을 위한 부분을 생각해보고 설치하는것이 좋을것 같다.) 인증을 요구하는 Client등의 관리를 위하여 *MySQL* <http://www.mysql.org/>등을 설치하는 것도 좋을것 같다. 그리고 mod_ssl의 INSTALL화일을 살펴보면 선택적으로 설치하는 화일이 있는데(mm같은것) 이것들도 필요하면 같이 설치할 수 있다. Perl은 5.6버젼을 필요로 한다고 되어있는것 같은데, 5.6을 굳이 설치할 필요는 없었던것 같다.

## 4.1 OpenSSL의 설치

OpenSSL을 설치하기 위해서는 *Perl 5* <http://www.perl.com>와 ANSI C컴파일러가 필요하다. 기본적인 설치 방법은 인터넷상의 다른 패키지들처럼 다음과 같이하여 설치할수 있다.

```
$ ./config
$ make
$ make test
$ make install
```

config에서 prefix를 주지 않았을 때에는 /usr/local/ssl디랙토리에 설치가 되니 다른 디랙토리에 설치를 하고자 한다면 다음과 같이 한다. **$ ./config –prefix=/usr/local –openssldir=/usr/local/openssl** 이렇게 하면 OpenSSL의 실행화일은 /usr/local/bin에 설치되고 Certificate Service를 위한 화일들은 /usr/local/openssl아래의 디랙토리에 생성된다. 또한 rsaref를 선택사항으로 줄수가 있는데 이것은 librsaref.a라는 라이브러리가 라이브러리 Search경로에 있어야 한다. (법적으로 문제가 되는부분이 아닌가 모르겠네요..어디서 읽은것 같아서..-.-; ftp search에서 rsaref20으로 검색하시면 찾을수 있습니다...그런데, 넣고 컴파일하면 에러가 나더라구요..)

**make**를 실행하면 OpenSSL라이브러리(libcrypto.a와 libssl.a)와 실행화일 (openssl)이 생성되고, 라이브러리는 make를 실행한 디랙토리에, OpenSSL실행화일은 apps디랙토리에 생성된다. make를 실행하는 도중 에러가 생기면, 에러메세지를 살펴보아야 한다. 대부분 OpenSSL의 문제가 아니라고 하며, 정말로 OpenSSL의 문제라고 판단된다면 make report명령을 이용하여 출력된 내용을 openssl-bugs@openssl.org에 송신한다.

다음은 **make test**를 실행하여 OpenSSL이 정상적으로 동작하는지 확인한다. 실패한 내용이 있다면, CFLAGS등을 수정하고 "make clean; make" 명령을 이용하여 다시 컴파일한후 make test를 실행해 본다.

모든 시험이 정상적으로 성공한다면, **make install**명령어를 이용하여 설치디랙토리(/usr/local/ssl이나 /usr/local/bin, /usr/local/openssl등)에 실행화일과 도움말 화일, 환경설정화일등을 설치한다. 여기까지 별 문제 없이 실행되었다면, Certificate Server를 실행하기 위한 기본적인 준비는 완료된 것이다. 다음은 OpenSSL을 이용한 Certificate Server운용을 위한 키생성과 환경설정을 실시한다.

## 4.2 rootCA의 생성

Certificate Server를 운용하는 곳은 Tree구조의 형태로 이루어져 있다. 제일 처음에 나타나는 CA를 rootCA라고 하고 자기 자신이 생성한 인증요구서를 자기자신이 sign하여 개인이나, 기관에게 배포한다. rootCA로부터 인증을 받은 인증기관은 중계인증기관이라고 하며, 인증요구서를 rootCA로 부터 인증받아서 다른사람이나 기관의 인증요구서를 인증해주는 역활을 한다. 지금부터는 앞에서 생성된 OpenSSL의 실행화일들을 이용해서 자신만의 rootCA를 생성한다.

(아래에서 설명하는 내용은 /path/to/openssldir/misc에 보시면 CA.pl과 CA.sh 가 있습니다. CA.xx를 적절히 수정하고 ./CA.xx -newca로 하여 rootCA를 생성하셔도 됩니다. Perl과 shell의 script임으로 크게 어려운 것은 없을 겁니다...^^; "CATOP"변수를 적절히 수정하시면 됩니다. 또한 Client들의 Sign Request가 있을 때에도 사용하실수 있으므로 자세히 살펴보시는것이 좋습니다.)

1. 비밀키의 생성$ **openssl genrsa -des3 -out ca.key 1024**

   1024비트의 보안도를 가지는 RSA키를 생성한다. Certificate Server자체의 비밀키가 됨으로 받드시 다른사람이 볼수 없는곳에 보관하고 빽업을 받은후에 사용하는것이 좋다.

2. 생성된 비밀키를 이용하여 Self Sign된 인증서 화일을 생성한다.$ **openssl req -new -x509 -days 365 -key ca.key -out ca.crt**

   위의 명령어를 이용하면 Certificate Server의 Distinguished Name을 물어온다. 질문에 대한 답을 정확하게 입력하고나면 rootCA의 Certificate Server의 인증서가 생성된다. 그리고 필요에 따라 days같은 항목은 수정해서 사용하도록 한다. (비밀번호 잊어먹지 않도록 주의하세요..T.T)

3. openssl디랙토리의 openssl.cnf화일을 수정한다.openssl.cnf화일은 Certificate Service를 운용하기 위한 환경설정을 담고 있는 화일로 make install을 했을때에는 sample 자료가 들어있다. 이 화일은 편집하여 여러분의 Certificate Service를 할 수 있도록 수정 하여야 한다.

   (a) CA_default 부분에 "dir"을 여러분이 사용할 디랙토리를 적는다.(보통은 "/path/to/openssldir/자기CA"의 형태)

   (b) usr_cert부분의 nsCertType이 모두 #으로 막혀있으므로 필요한 내용을 골라서 풀어준다.(Netscape Client의 인증종류를 결정하는 부분인것 같다.)

   (c) Ca_default의 dir에 필요한 디랙토리와 화일을 생성한다. 디랙토리는 certs, crl, newcerts, private이고 화일은 "01"의 내용을 담고있는 serial과 내용이 없는 index.txt이다. (CA.xx의 내용과 openssl.cnf를 참고하시면 됩니다.)

   ```
   $ mkdir /usr/local/openssl/MyCA
   $ mkdir /usr/local/openssl/MyCA/certs
   $ mkdir /usr/local/openssl/MyCA/crl
   $ mkdir /usr/local/openssl/MyCA/newcerts
   $ mkdir /usr/local/openssl/MyCA/private
   $ echo "01" > /usr/local/openssl/MyCA/serial
   $ touch /usr/local/openssl/MyCA/index.txt
   ```

   (d) keyUsage부분의 #을 풀어준다.

   (e) 필요하다면 nsComment를 원하는것으로 바꾼다.

4. 생성된 비밀키와 공개키를 openssl.cnf에서 지정한 디랙토리에 복사한다.ca.key는 /path/to/openssldir/자기CA/private/cakey.pem으로 복사하고 ca.crt는 /path/to/openssldir/자기CA/cacert.pem정도로 복사하면 됩니다.

5. 공개키는 application/x-x509-ca-cert의 MIME형태로 Client Browser에 전송할 수 있으며, Client에서는 "신뢰된 루트인증기관"으로 import할 수 있다.

여기까지 하셨으면, 불편하지만 수동(?)으로 Certificate Service를 할 수 있습니다. 다음은 조금이나마 편안하게 Service를 하기 위해서 Apache에서 몇가지 작업을 합니다.(나중에 Client와 연결해서 CSR을 생성할때 사용합니다.)

## 4.3 Apache + mod_ssl/OpenSSL + PHP4/MySQL의 설치

일반적인 Apache 와 php의 설치 방법에 큰 차이는 없으니, 여기에서는 순서만 기술하고 넘어 가도록 한다. 또한 이 내용은 mod_ssl의 INSTALL화일에 설명이 되어 있으므로 그것을 읽어보는 것이 정확할 것이다. mod_ssl은 버젼별로 Apache와 연결이 됨으로 자신이 설치하는 Apache버젼과 일치하는 mod_ssl을 설치하여야 한다. 이는 mod_ssl의 화일명을 보면 알 수가 있는데, "mod_ssl-자신버젼-아파치버젼"의 형태로 되어있다.

```
$ tar xvfz apache_1.3.x.tar.gz
$ tar xvfz mod_ssl-2.7.x-1.3.x.tar.gz
$ tar xvfz php-4.0.x.tar.gz
$ cd apache_1.3.x
$ ./configure
$ cd ../mod_ssl-2.7.x-1.3.x
$ ./configure --with-apache=../apache_1.3.x
$ cd ../apache_1.3.x
$ .configure --prefix=/path/to/apache
$ cd ../php-4.0.x
$ CFLAGS='-O2 -I/path/to/openssl/include' \
  ./configure \
        --with-apache=../apache_1.3.x \
        --with-mysql \
        --enable-memory-limit=yes \
        --enable-debug=no
$ make
$ make install
$ cd ../apache_1.3.x
$ SSL_BASE=/path/to/openssl \
  ./configure \
        --prefix=/path/to/apache \
        --enable-module=ssl \
        --activate-module=src/modules/php4/libphp4.a \
        --enable-module=php4
$ make
$ make certificate
$ make install
```

컴파일 과정중 주의할 점은 마지막의 Apache의 환경설정을 할때에 SSL_BASE는 OpenSSL을 설치할때에 prefix에 주어진 디렉토리를 주어야 한다. (**openssldir이 아니다.**)

상기의 컴파일 과정을 거쳐서 생성된 Apache서버는 **apachectl start**명령을 이용하여 실행을 하게 되면 일반 Apache서버와 동일한 기능을 하게 되고 **apachectl startssl** 명령을 이용하면 mod_ssl을 이용한 https프로토콜을 사용할수 있게 된다. Client에서 사용법은 url에 "https://server/path/"라고 적게되며, 이때 Server와 Client간의 암호화 방법에 대한 협상이 이루어 지게 되고 이후에 Server와 Client간의 자료는 암호화 되어 전송이 됨으로, sniffer를 이용한 방법으로는 자료를 훔쳐볼수 없게 된다. 자세한것은 *kldp.org* `<http://kldp.org/>`에서 mod_ssl을 검색어로 찾아보길 바란다.

## 4.4 Apache Server의 키 생성과 인증

Apache + mod_ssl을 설치하면 기본적으로 30일간 사용할수 있는 Certificate가 /path/to/apache/conf/ssl.crt 디렉토리에 설치되어 있다. 이것을 여러분의 CA에서 인증한다면 여러분이 원하는 기간동안 사용할

수 있는 Certificate가 생성된다.(1년이든 10년이든...^^;) (아래의 과정을 진행하다 보면 기존에 설치된 OpenSSL과 충돌하는 경우가 있습니다. 이때에는 기존에 설치된 OpenSSL을 제거 하시거나, "-config /path/to/openssldir/openssl.cnf"를 추가로 입력해서 사용하십시요.) 설치과정중 make certificate TYPE=custom으로 실행하는 경우에는 openssl에서 생성시킨 rootCA의 key와 certificate화일을 다시 설치해야 한다. 이때 생성된 rootCA에는 object sign(S/W에 사인하는것(?))이 빠져 있으므로 rootCA를 운영하는 데에는 부적합 할 수도 있다.

1. openssl명령어를 이용하여 Web Server의 RSA키를 생성한다.$ **openssl genrsa -des3 -out server.key 1024**

   이때 생성되는 키는 Web Server를 운용하기위한 비밀키가 됨으로 반드시 백업을 받아놓고 사용하는 것이 좋다. 생성된 키는 다음 명령을 이용하여 내용을 확인할수 있다.

   $ **openssl rsa -noout -text -in server.key**

2. 생성된 RSA키를 이용하여 Certificate Signing Request(CSR)을 생성한다.$ **openssl req -new -days 365 -key server.key -out server.csr**

   위의 명령어를 수행하는 도중 CommonName을 물어볼때에 Server의 FQDN(Fully Qualified Domain Name -> host.domain.com의 형태)로 적어야 한다. 생성된 화일은 다음 명령으로 내용을 확인 할 수 있다.

   $ **openssl req -noout -text -in server.csr**

3. 생성된 CSR화일은 유료서비스를 하는 VeriSign이나 Thawte등의 rootCA로 보내면 server.crt화일을 받을수 있다. OpenSSL을 이용할 때에는 위에서 Self Sign할 때처럼 "openssl ca"명령을 이용한다. 이때 반드시 /path/to/openssldir/openssl.cnf의 내용중 nsCertType을 Server로 해놓아야 한다.(**nsCertType = Server**) (요거 빼먹으면 Netscape에서 https로 접속 할 수가 없어요.. -.-) $ **openssl ca -in server.csr -out server.crt**

   (CA.pl이나 CA.sh를 적절히 수정하였다면 "CA.xx -sign"을 이용해도 됩니다. 그리고 기존의 OpenSSL과 충돌하지 않는지 확인하시고 사용하십시요.)

4. 생성된 server.crt와 server.key화일을 적절한 위치에 복사하고 빽업한다.Apache의 conf/httpd.conf에 보면, SSLCertificateFile 항목이 지시하는곳이 server.crt화일을 나타내는 곳이고 SSLCertificateKey-File 항목이 server.key화일을 지시한다.

## 제 5 절 Client의 인증

인증을 받기위해서는 DN(Distinguished Name)을 포함하고 있는 인증요구서(CSR)를 생성하여 CA에 송신해야 한다. DN은 개인이나, 기관, 서버등을 구분하기위한 것으로 이름, 부서, 직장, 주소, 나라등의 필드로 이루어져 있고, CN, OU, O, L, ST, C등의 필드로 표현된다. 인증을 받기위한 Client들의 인증요구서(CSR) 화일이 각각 다른형태로 생성이 되고, Client들이 인증서를 수령하는 방법이 제각기 다름으로 Client별로 각각의 사용방법을 설명한다.

Client들이 자신의 인증서를 수령할 때나, 다른사람이 사인한 코드를 실행할 때에는 rootCA의 인증서가 미리 설치되어 있어야 한다. 따라서 앞에서 생성한 rootCA의 Certificate를 미리 수신해서 설치해야 한다. Netscape와 MS Explorer는 application/x-x509-ca-cert MIME을 이용해서 서버에서 서비스하는곳에 접속하면 설치할수 있고, JAVA의 SDK인 경우에는 미리 다운로드 한다음 "keytool -import"명령을 이용해서 설치해야 한다. 또한 다음의 Netscape나 MS Interne Explorer에서 인증요구서(Certificate Signing Request:CSR)을 생성할 때에는 개인의 정보가 Network을 이용하여 전송됨으로 반드시 **https** 프로토콜을 이용하도록 한다.

## 5.1 Netscape 사용자인증

Netscape에서는 CSR화일을 위해서 SPKAC형태의 자료를 생성한다. CSR화일을 생성하기 위해서는 Web Server에서 KEYGEN TAG를 사용한 Form을 생성하여야 한다. 다음은 KEYGEN TAG를 사용한 FORM의 예이다.

```
<FORM name="CSR_request" method="post" action="user_cert.cgi">
<table border=1>
<tr><td> 성명   </td><td><input name="CN" type=input></td>
<tr><td> E-mail </td><td><input name="Email" type=input></td>
<tr><td> 부서   </td><td><input name="OU" type=input></td>
<tr><td> 회사   </td><td><input name="O" type=input></td>
<tr><td> 시     </td><td><input name="L" type=input></td>
<tr><td> 도     </td><td><input name="ST" type=input></td>
<tr><td> 국가   </td><td><input name="C" type=input></td>
<tr><td> 키생성 </td><td><keygen name="SPKAC" type="RSA"></td>
</table>
</FORM>
```

KEYGEN TAG의 기본문법은 다음과 같다.

**<KEYGEN NAME="name" CHALENGE="challenge string" KEYTYPE="type" PQG="pqg-params" >**

NAME항목은 필수이고(cgi에서 처리하여야 함) KEYTYPE은 RSA와 DSA중에 하나를 선택할수 있다. PQG는 DSA에서만 사용한다. keygen tag는 form tag안에서만 유효하며, 특정한 길이의 key크기를 선택할 수 있다. 미국내에서는 512, 768, 1024비트의 키를 사용할 수 있고, 그외 국가에서는 512비트의 키를 사용할 수 있다. Form을 전송하면 512비트의 보안도를 가지는 Key Pair가 생성되면 비밀키는 로컬DB에 암호화 되어서 저장되고 공개키와 challenge스트링은 DER encode되고 비밀키로 sign된다음 base64로 encode되고 서버로 전송된다. 서버측에서는 DN과 SPKAC스트링을 cgi를 이용하여 화일로 만들고 다음과 같은 명령을 이용하여 Sign한다. (이때 SPKAC스트링은 1줄로 되어 있어야 한다.)

**openssl ca -spkac 생성한화일 -out 인증서.crt**

생성된 인증서는 application/x-x509-user-cert의 MIME형태로 Netscape Browser로 전송하여 import할수 있도록 한다. 이때 Netscape의 로컬DB에 인증서에 해당하는 비밀키가 존재하지 않으면 import할수 없다.

## 5.2 MS Internet Explorer 사용자인증

MS Internet Explorer는 PKCS#10형태의 인증요구서를 발행하며, PKCS#7형태의 인증서를 수령할수 있다. 따라서 Server측 입장에서 볼때에는 약간 귀찮은 작업을 하나더 해줘야 한다. 그리고 Explorer에서는 Netscape처럼 KEYGEN TAG를 사용하지 않고 Microsoft에서 정의한 CryptoAPI(CAPI)를 이용한다.(좀더 유연하다고 하는데...정말일까요?..^^;) CAPI는 기본적인 암호화 작업을 위한 함수를 제공한다.(키 생성, 자료의 암호화, 문서의 서명, 인증서의 요구등)

CAPI는 다음의 4단계를 이용하여 인증요구서를 발행하고 인증서를 수령한다.

1. CAPI함수가 당신을 위한 Key Pair를 생성하고 인증서 창고에 저장한다.
2. DN(Distinguished Name)을 생성한다.(?)
3. CA로 전송을 하기위한 PKCS#10형태의 자료를 생성한다.
4. CA에서 인증한 자료를 인증서 창고에 저장한다.

Explorer의 인증서는 "HKEY_CURRENT_USER/Software/Microsoft/Cryptography/"에 저장된다. CAPI함수는 OBJECT TAG를 이용하여 로컬디스크에 있는 DLL을 읽어들인후에 사용 할 수 있다. 다음은 CAPI를 위한 xenroll.dll을 읽어들이는 예이다. (예전에는 certenr3.dll이라는 화일에서 지원했었는데 지금은 사용되지 않고 있고, 사용했을경우에 OpenSSL에서 사인방법이 틀려진다.)

```
<OBJECT classid="clsid:43F8F289-7A20-11D0-8F06-00C04FC295E1"
          codebase="xenroll.dll"
          id=Enroll>
</OBJECT>
```

정상적으로 xenroll.dll이 불려졌으면 DN스트링을 만들고 xenroll의 createPKCS10함수를 불러서 PKCS#10형태의 인증 요구서를 만든다. DN스트링의 문자열의 구성과 createPKCS10함수콜은 다음과 같이 한다.

**$szStr="CN=SeungSu KUK; C=KR; ST=KyungBook; L=Kyungsan; O=Catholic University of DAEGU; OU=Computer Center";**

**$result=Enroll.createPKCS10(szStr, "1.3.6.1.4.1.311.2.1.21");**

생성된 $result에는 PKCS#10형태의 문자열이 들어 있고 이것을 CA로 전송한다. CA측에서는 이것에 사인하고 PKCS#7의 형태로 인증서를 돌려준다.

OpenSSL에서는 다음과 같은 방법을 이용하여 사인하고 인증서를 생성한다. 먼저 $result에서 들어오는 자료에는 "BEGIN CERTIFICATE REQUEST"라는 문자열이 없는 상태임으로 $result를 이용하여 다음과 같은 형태의 화일을 생성한다.

**——BEGIN CERTIFICATE REQUEST——**

**$result에서 ^M을 삭제한 내용**

**——END CERTIFICATE REQUEST——**

생성된 화일은 다음과 같이 사인하고 PKCS#7형태의 화일을 생성한다.

**openssl ca -in 생성된화일 -out cert결과화일**

**openssl crl2pkcs7 -certfile cert결과화일 -in /path/to/crl.pem -out pkcs7화일**

(crl.pem화일은 미리 생성되어 있어야 하는데 "openssl ca -gencrl"명령을 이용하여 생성한다.) pkcs7화일을 Explorer로 전송할 때에는 "BEGIN PKCS7"와 "END PKCS7"이라는 문자열(첫줄과 끝줄)을 지운상태로 전송해야 한다. (않그러면 에러가 발생한다.) 전송된 PKCS7화일(VBScript나 JavaScript의 변수로 전송)은 AcceptPKCS7함수에 의해서 import되며 다음과 같이 한다.

**$szStr="전송된 PKCS7화일의 내용"**

**$result=Enroll.AcceptPKCS7(szStr);**

## 5.3 JDK2 사용자인증

(여기는 1년전의 내용입니다..-.- 따라서 간단한 개념만 적어보겠습니다. 그리고 빠른시간내에 전제적인 내용을 정리하겠습니다.)

JDK2에서는 applet이나 javascript등을 사인하기 위해서 인증서를 사용한다. 비밀키와 공개키의 생성은 keytool을 통해서 이루어지며 keytool을 통하여 import/export기능을 수행한다. keytool을 이용한 공개키/비밀키의 생성은 다음과 같이 한다.

**keytool -genkey -alias 사용자명 -keypass 해당키의비밀번호**

keytool에서 "-genkey"를 이용하면 Distinguished-Name을 요구하고 시험용으로 사용하는 Self-Signed Certificate를 생성한다. 이것은 나중에 "import"에의해 CA에서 인증한 키로 대체된다. keytool에서 사용할수 있는 옵션은 다음과 같다.

- -certreq : CSR화일을 생성한다.
- -delete : keystore에서 해당키를 삭제한다.
- -export : 공개키를 추출한다.
- -genkey : keystore entry를 생성한다.
- -import : CA에서 인증한 인증서를 import한다.
- -list : keystore의 내용을 보여준다.
- -keypasswd: 비밀키에 암호를 설정한다.
- -storepasswd: (요건모지?)

인증요구서를 만들기 위해서는 "-certreq"를 사용한다. (이때 genkey에서 입력한 비밀번호를 넣어야 한다.)

**keytool -certreq -keystore keystore이름 -alias 사용자명 -file CSR화일명**

생성된 CSR화일은 CA에서 인증하고 "-import"를 이용해서 keystore에 저장한다.

**keytool -import -alias 사용자명 -file CERT화일명 -keystore keystore명**

내용을 확인하고 싶으면 **keytool -printcert -file CERT화일명**이렇게 입력한다.

OpenSSL에서 J2sdk에서 요구한 인증서를 처리할 때에는 다음과 같이하여 인증서를 생성하고 E-mail등을 이용하여 돌려주면 된다.

**openssl ca -in certreq화일 -out 결과cert화일**

# 제 6 절 기타

## 6.1 참고자료

1. *www.openssl.org* <http://www.openssl.org/>
2. *www.modssl.org* <http://www.modssl.org/>
3. *www.apache.org* <http://www.apache.org/>
4. Frederick J. Hirsch의 *cookbook* <http://www.ultranet.com/~fhirsch/Papers/cook/ssl_cook.html> (cookbook은 *SSLeay* <http://www.ssleay.org/>에 관한내용이 많습니다만 OpenSSL이 ssleay를 기초로하기 때문에 참고가 많이 될것 같습니다. 저는 이런게 있는지도 모르고 맨땅에 헤딩함서 만들었는디...정리하면서 찾아보니 좋은자료가 있네요..)

## 6.2 헛소리..^^

옛날에(1년전쯤..^^) 자료를 여기저기서 참고해서 만들었던 것인데... 참고 자료를 찾을려니, 어디에 있는지 잘 모르겠군요. 원고 마감할때까지 찾아놔야 할 텐데...-.-;

저도 아직 OpenSSL을 이용한 Certificate Server운용방법에는 익숙하지 않습니다. CRL(Certificate Revoke List(?))나 사용자의 CSR화일을 자동화하여 처리하는 방법론등 여러가지를 아직 제대로 하지 못하고 있습니다. 이 글을 쓰게되면서 더이상 썩히지말고 써먹어야 되겠다는 생각이 드는군요...아직 덜 된 부분

들(Client등에서의 사용법)은 제가 시험해보구 정리가 되면 이문서를 버전업(^^) 해서 다시 올리도록 하겠습니다. 본문에 틀린내용이 있거나 건의사항이 있으면 언제든지 환영합니다. 메일주시면 착실히 답은 못해드려도 반드시 수정하도록 하겠습니다.

## 6.3 감사의 말

업무와 관계없는 일을 하는 저를 옆에서 지켜봐주고 여러가지 시험을 해주는 박상수님께 감사하며, 볼품없는 제글을 끝까지 읽어주신 여러분께 감사드립니다..^^;