anywhere, you're safe
anycert

# Apache 웹서버 (2.0.xx) 인증서 설치

Apache 웹서버는 Apache 웹서버 설정 파일(2.0.xx 버젼은 ssl.conf 파일)에서 [SSL Virtual Host Context] 영역에 웹서버 인증서를 설치하게 됩니다. Apache 웹서버에 SSL 설정되는 인증서는 발급된 인증서(공개키)와 고객님의 개인키 파일을 설치하게 됩니다.

---

## ※ 아파치 웹서버의 인증서 설치 순서

1. 발급된 인증서 확인
2. 웹서버로 발급된 인증서 올리기
3. Apache 웹서버에 인증서 설치하기
4. Apache SSL restart
5. SSL 구동 확인
6. 인증서 백업해 두기

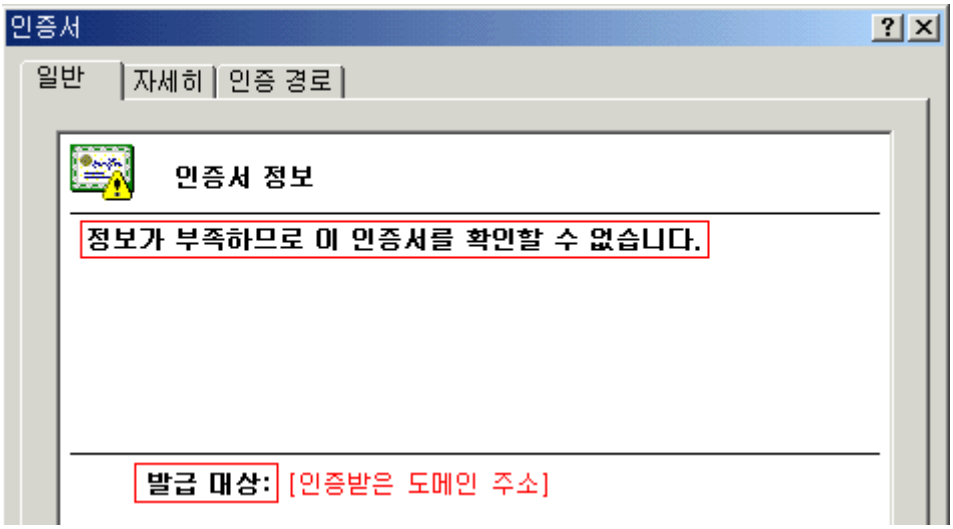
---

## 1. 발급된 인증서 확인

먼저 발급된 인증서를 확인합니다. 웹서버 인증서와 체인인증서, 두개의 파일이 발급됩니다.

웹서버인증서 : [인증받은 도메인 이름으로 된].crt
체인인증서 : Bundle.crt
위의 발급된 인증서는 애니서트에서 고객님께 전자 메일로 발급해 드립니다. 그러므로 전자 메일에 발급된 인증서 항목을 알려드리며, 발급된 인증서는 첨부파일로 첨부되어있습니다.

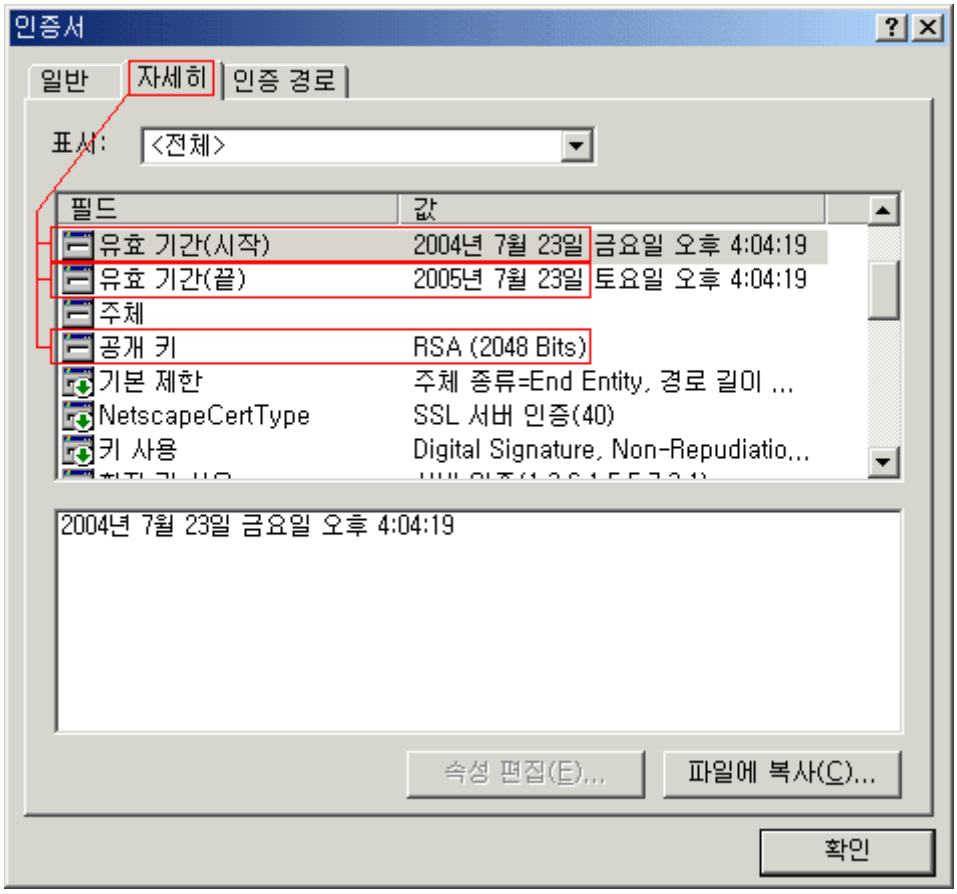
윈도우 PC 에서 발급 받으신 웹서버 인증서([인증받은 도메인 이름으로 된].crt 파일)를 열어서(double click) 내용을 확인합니다.(메모장이나 울트라 에디터에서 열지 않습니다.)
웹서버 인증서를 열게 되면 다음과 같은 내용을 확인하실 수 있습니다.




[인증서 보기 - 일반]

웹서버 인증서 정보의 [일반] 탭 부분에서 인증서 정보에서 "정보가 부족하므로 이 인증서를 확인할 수 없습니다." 메지시를 확인 할 수 있습니다. 현재 발급된 인증서는 함께 드린 체인인증서와 합쳐져서 사용되는 웹서버 인증서이므로, 위의 같은 메시지를 확인 하는 것이 맞습니다.
그리고 발급 대상에서 고객님이 인증받을 도메인 주소로 신청하신 도메인 주소가 맞는지 확인합니다.

다음으로는 [자세히] 탭 부분에서



[인증서 보기 - 자세히]

발급한 웹서버 인증서의 유효기간을 확인합니다. (유효 기간(시작)과 유효 기간(끝) 정보를 확인합니다.)
그리고 공개 키 부분에서 RSA (2048 Bits) 를 확인합니다. 애니서트에서 발행되는 인증서는 2048 Bits 키를 권고합니다. 꼭 확인해 주시기 바랍니다.

그리고 먼저 CSR(Certificate Signing Request) 파일을 생성하시면서, Apache 웹서버에 개인키도 이미 생성했었습니다. 개인키도 확인해 주시기 바랍니다. (CSR 생성때에 개인키를 체크했으므로, 어디에 개인키가 저장되어있는지 확인만 해 주시면 됩니다.)

## 2. 웹서버로 발급된 인증서 올리기

발급받으신 인증서를 Apache 웹서버에 복사합니다.(보통 Ftp 로 웹서버로 올려 주시면 됩니다.)

## 3. Apache 웹서버에 인증서 설치하기

$HTTPD/conf/ssl.conf 웹서버 설정 파일이 있습니다. ($HTTPD 변수는 아파치 설치 디렉토리를 가르킵니다. 설치시에 다른 디렉토리에 웹서버 설정파일 ssl.conf 을

설치할 수도 있으므로 ssl.conf 파일을 확인 바랍니다.)
다음은 ssl.conf 파일의 HTTPS port, [SSL Global Context], [SSL Virtual Host Context] 부분에 인증서 설치 영역의 원본 내용입니다.

```
#
# This is the Apache server configuration file providing SSL
support.
# It contains the configuration directives to instruct the
server how to
# serve pages over an https connection. For detailing
information about these
# directives see <URL:http://httpd.apache.org/docs-
2.0/mod/mod_ssl.html>
#
# Do NOT simply read the instructions in here without
understanding
# what they do.  They're here only as hints or reminders.  If
you are unsure
# consult the online docs. You have been warned.
#

#
# Pseudo Random Number Generator (PRNG):
# Configure one or more sources to seed the PRNG of the SSL
library.
# The seed data should be of good random quality.
# WARNING! On some platforms /dev/random blocks if not enough
entropy
# is available. This means you then cannot use the /dev/random
device
# because it would lead to very long connection times (as long
as
# it requires to make more entropy available). But usually those
# platforms additionally provide a /dev/urandom device which
doesn't
# block. So, if available, use this one instead. Read the
mod_ssl User
# Manual for more details.
#
# Note: This must come before the <IfDefine SSL> container to
support
#       starting without SSL on platforms with no /dev/random
```

```
equivalent
#       but a statically compiled-in mod_ssl.
#
SSLRandomSeed startup builtin
SSLRandomSeed connect builtin
#SSLRandomSeed startup file:/dev/random  512
#SSLRandomSeed startup file:/dev/urandom 512
#SSLRandomSeed connect file:/dev/random  512
#SSLRandomSeed connect file:/dev/urandom 512

<IfDefine SSL>

#
# When we also provide SSL we have to listen to the
# standard HTTP port (see above) and to the HTTPS port
#
# Note: Configurations that use IPv6 but not IPv4-mapped
addresses need two
#       Listen directives: "Listen [::]:443" and "Listen
0.0.0.0:443"
#
Listen 443

##
##  SSL Global Context
##
##  All SSL configuration in this context applies both to
##  the main server and all SSL-enabled virtual hosts.
##

#
#   Some MIME-types for downloading Certificates and CRLs
#
AddType application/x-x509-ca-cert .crt
AddType application/x-pkcs7-crl    .crl

#   Pass Phrase Dialog:
#   Configure the pass phrase gathering process.
#   The filtering dialog program (`builtin' is a internal
#   terminal dialog) has to provide the pass phrase on stdout.
SSLPassPhraseDialog  builtin
```

```
#   Inter-Process Session Cache:
#   Configure the SSL Session Cache: First the mechanism
#   to use and second the expiring timeout (in seconds).
#SSLSessionCache        none
#SSLSessionCache        shmht:logs/ssl_scache(512000)
#SSLSessionCache        shmcb:logs/ssl_scache(512000)
SSLSessionCache         dbm:logs/ssl_scache
SSLSessionCacheTimeout  300

#   Semaphore:
#   Configure the path to the mutual exclusion semaphore the
#   SSL engine uses internally for inter-process
synchronization.
SSLMutex  file:logs/ssl_mutex

##
## SSL Virtual Host Context
##

<VirtualHost _default_:443>

#   General setup for the virtual host
DocumentRoot "htdocs"
ServerName www.example.com:443
ServerAdmin you@example.com
ErrorLog logs/error_log
TransferLog logs/access_log

#   SSL Engine Switch:
#   Enable/Disable SSL for this virtual host.
SSLEngine on

#   SSL Cipher Suite:
#   List the ciphers that the client is permitted to negotiate.
#   See the mod_ssl documentation for a complete list.
SSLCipherSuite
ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL

#   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.  Keep
#   in mind that if you have both an RSA and a DSA certificate
you
#   can configure both in parallel (to also allow the use of DSA
#   ciphers, etc.)
SSLCertificateFile conf/ssl.crt/server.crt
#SSLCertificateFile conf/ssl.crt/server-dsa.crt

#   Server Private Key:
#   If the key is not combined with the certificate, use this
#   directive to point at the key file.  Keep in mind that if
#   you've both a RSA and a DSA private key you can configure
#   both in parallel (to also allow the use of DSA ciphers,
etc.)
SSLCertificateKeyFile conf/ssl.key/server.key
#SSLCertificateKeyFile conf/ssl.key/server-dsa.key

#   Server Certificate Chain:
#   Point SSLCertificateChainFile at a file containing the
#   concatenation of PEM encoded CA certificates which form the
#   certificate chain for the server certificate. Alternatively
#   the referenced file can be the same as SSLCertificateFile
#   when the CA certificates are directly appended to the server
#   certificate for convinience.
#SSLCertificateChainFile conf/ssl.crt/ca.crt

#   Certificate Authority (CA):
#   Set the CA certificate verification path where to find CA
#   certificates for client authentication or alternatively one
#   huge file containing all of them (file must be PEM encoded)
#   Note: Inside SSLCACertificatePath you need hash symlinks
#         to point to the certificate files. Use the provided
#         Makefile to update the hash symlinks after changes.
#SSLCACertificatePath conf/ssl.crt
#SSLCACertificateFile conf/ssl.crt/ca-bundle.crt

#   Certificate Revocation Lists (CRL):
#   Set the CA revocation path where to find CA CRLs for client
#   authentication or alternatively one huge file containing all
```

```
#   of them (file must be PEM encoded)
#   Note: Inside SSLCARevocationPath you need hash symlinks
#         to point to the certificate files. Use the provided
#         Makefile to update the hash symlinks after changes.
#SSLCARevocationPath conf/ssl.crl
#SSLCARevocationFile conf/ssl.crl/ca-bundle.crl

#   Client Authentication (Type):
#   Client certificate verification type and depth.  Types are
#   none, optional, require and optional_no_ca.  Depth is a
#   number which specifies how deeply to verify the certificate
#   issuer chain before deciding the certificate is not valid.
#SSLVerifyClient require
#SSLVerifyDepth  10

#   Access Control:
#   With SSLRequire you can do per-directory access control
based
#   on arbitrary complex boolean expressions containing server
#   variable checks and other lookup directives.  The syntax is
a
#   mixture between C and Perl.  See the mod_ssl documentation
#   for more details.
#<Location />
#SSLRequire (    %{SSL_CIPHER} !~ m/^(EXP|NULL)/ ₩
#            and %{SSL_CLIENT_S_DN_O} eq "Snake Oil, Ltd." ₩
#            and %{SSL_CLIENT_S_DN_OU} in {"Staff", "CA", "Dev"}
₩
#            and %{TIME_WDAY} >= 1 and %{TIME_WDAY} <= 5 ₩
#            and %{TIME_HOUR} >= 8 and %{TIME_HOUR} <=
20       ) ₩
#           or %{REMOTE_ADDR} =~ m/^192₩.76₩.162₩.[0-9]+$/
#</Location>

#   SSL Engine Options:
#   Set various options for the SSL engine.
#   o FakeBasicAuth:
#     Translate the client X.509 into a Basic Authorisation.
This means that
#     the standard Auth/DBMAuth methods can be used for access
control.  The
```

```
#     user name is the `one line' version of the client's X.509
certificate.
#     Note that no password is obtained from the user. Every
entry in the user
#     file needs this password: `xxj31ZMTZzkVA'.
#   o ExportCertData:
#     This exports two additional environment variables:
SSL_CLIENT_CERT and
#     SSL_SERVER_CERT. These contain the PEM-encoded
certificates of the
#     server (always existing) and the client (only existing
when client
#     authentication is used). This can be used to import the
certificates
#     into CGI scripts.
#   o StdEnvVars:
#     This exports the standard SSL/TLS related `SSL_*'
environment variables.
#     Per default this exportation is switched off for
performance reasons,
#     because the extraction step is an expensive operation and
is usually
#     useless for serving static content. So one usually enables
the
#     exportation for CGI and SSI requests only.
#   o CompatEnvVars:
#     This exports obsolete environment variables for backward
compatibility
#     to Apache-SSL 1.x, mod_ssl 2.0.x, Sioux 1.0 and Stronghold
2.x. Use this
#     to provide compatibility to existing CGI scripts.
#   o StrictRequire:
#     This denies access when "SSLRequireSSL" or "SSLRequire"
applied even
#     under a "Satisfy any" situation, i.e. when it applies
access is denied
#     and no other module can change it.
#   o OptRenegotiate:
#     This enables optimized SSL connection renegotiation
handling when SSL
#     directives are used in per-directory context.
```

```
#SSLOptions +FakeBasicAuth +ExportCertData +CompatEnvVars
+StrictRequire
<Files ~ "₩.(cgi|shtml|phtml|php3?)$">
    SSLOptions +StdEnvVars
</Files>
<Directory "cgi-bin">
    SSLOptions +StdEnvVars
</Directory>

#   SSL Protocol Adjustments:
#   The safe and default but still SSL/TLS standard compliant
shutdown
#   approach is that mod_ssl sends the close notify alert but
doesn't wait for
#   the close notify alert from client. When you need a
different shutdown
#   approach you can use one of the following variables:
#   o ssl-unclean-shutdown:
#     This forces an unclean shutdown when the connection is
closed, i.e. no
#     SSL close notify alert is send or allowed to received.
This violates
#     the SSL/TLS standard but is needed for some brain-dead
browsers. Use
#     this when you receive I/O errors because of the standard
approach where
#     mod_ssl sends the close notify alert.
#   o ssl-accurate-shutdown:
#     This forces an accurate shutdown when the connection is
closed, i.e. a
#     SSL close notify alert is send and mod_ssl waits for the
close notify
#     alert of the client. This is 100% SSL/TLS standard
compliant, but in
#     practice often causes hanging connections with brain-dead
browsers. Use
#     this only for browsers where you know that their SSL
implementation
#     works correctly.
#   Notice: Most problems of broken clients are also related to
the HTTP
```

```
#   keep-alive facility, so you usually additionally want to
disable
#   keep-alive for those clients, too. Use variable
"nokeepalive" for this.
#   Similarly, one has to force some clients to use HTTP/1.0 to
workaround
#   their broken HTTP/1.1 implementation. Use variables
"downgrade-1.0" and
#   "force-response-1.0" for this.
SetEnvIf User-Agent ".*MSIE.*" ₩
         nokeepalive ssl-unclean-shutdown ₩
         downgrade-1.0 force-response-1.0

#   Per-Server Logging:
#   The home of a custom SSL log file. Use this when you want a
#   compact non-error SSL logfile on a virtual host basis.
CustomLog logs/ssl_request_log ₩
          "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x ₩"%r₩" %b"

</VirtualHost>

</IfDefine>
```

HTTPS port 설정에서는 기본적으로 설정된 사항입니다. 확인만 해 주시면 되겠습니다.
웹서버를 SSL 모드로 기동했을 경우에 443 포트(https 통신)를 서비스합니다.

[SSL Global Context] 설정에서는 기본적인 SSL 환경을 설정합니다. 잘 모르시는 부분은 기본값으로 설정해 놓습니다.

[SSL Virtual Host Context] 설정부분이 직접적으로 Apache 웹서버에 SSL 인증서를 설정하는 부분입니다.
다음 설정 예시를 참고하시고, 설정해 주시기 바랍니다.

```
<IfDefine SSL>

## SSL Virtual Host Context

<VirtualHost 1.2.3.4:443>  ##이름기반(name based)
SSL 가상호스트 설정합니다.
```

```
DocumentRoot "htdocs" ## 웹문서 루트 경로
ServerName [인증받은 도메인 주소]
ErrorLog logs/error_log ## 에러 로그 설정
TransferLog logs/access_log ## 접속 로그 설정

SSLEngine on  ## SSLEngine on 설정되어야 SSL
동작합니다.

#   Server Certificate:
SSLCertificateFile conf/ssl/[인증받은 도메인
이름으로 된].crt
## 웹서버 인증서 [인증받은 도메인 이름으로 된].crt
경로를 설정합니다.

#   Server Private Key:
SSLCertificateKeyFile conf/ssl/server.key
## CSR 파일생성할 때 먼저 생성한 개인키 경로를
설정합니다.

#   Server Certificate Chain:
SLCertificateChainFile conf/ssl/Bundle.crt
## 체인 인증서 Bundle.crt 경로를 설정합니다.
## 위의 부분은 기존에 주석처리되어있으므로, 주석을
풀고 설정합니다.

## 나머지 설정을 기본 설정으로 둡니다.

</VirtualHost>

</IfDefine>
```

### 4. Apache SSL restart

$HTTPD/bin/apachectl 웹서버 설정 파일이 있습니다. ($HTTPD 변수는 아파치 설치 디렉토리를 가르킵니다.)
$HTTPD/bin/apachectl -D SSL -k start 옵션으로 기동합니다.

```
[root@web1 root]#
$HTTPD/bin/apachectl -D SSL -k start
Apache/2.0.52 mod_ssl/2.0.52 (Pass
Phrase Dialog)
Some of your private key files are
```

```
encrypted for security reasons.
In order to read them you have to
provide us with the pass phrases.

Server web1:443 (RSA)
Enter pass phrase: [개인키 비밀번호
입력]

Ok: Pass Phrase Dialog successful.
[root@web1 root]#
```

이 부분에서 에러로 인해서 $HTTPD/bin/apachectl -D SSL -k start 실행되지 않는다면,
error_log 파일의 에러 상태를 파악하시고 해결하시기 바랍니다.
애니서트로 문의를 주실 경우에는 간단한 설명과 함께 현재 에러 로그를 애니서트 메일로 통보해 주시기 바랍니다.

## 5. SSL 구동 확인

Apache 2.0.xx 웹서버 ssl 모드로 구동된 것을 확인합니다.
($HTTPD 변수는 아파치 설치 디렉토리를 가르킵니다.)

1. 프로세스 로딩 확인

```
▣
[root@web1 root]# ps -ax
  PID TTY      STAT   TIME COMMAND
... ...
11197 ?        S      0:00 $HTTPD/bin/httpd -k
start -D SSL
... ...
11203 pts/1    R      0:00 ps -ax
[root@web1 root]#
```

이 부분에서 에러로 인해서 $HTTPD/bin/httpd -k start -D SSL 가 시작되지 않았다면,
error_log 파일의 에러 상태를 파악하시고 해결하시기 바랍니다.
애니서트로 문의를 주실 경우에는 간단한 설명과 함께 현재 에러 로그를 애니서트 메일로 통보해 주시기 바랍니다.

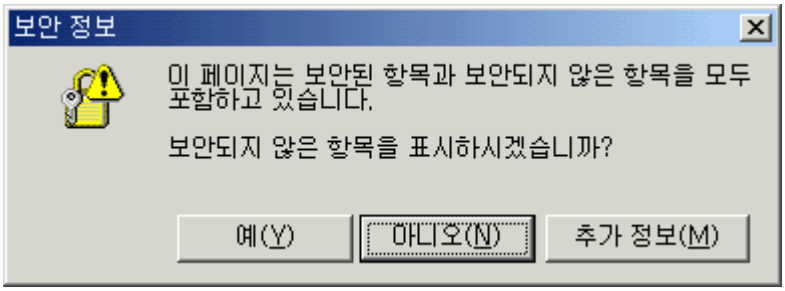
## 2. 443 포트(HTTPS 통신) 네트웍 활성 확인

```
▣
[root@web1 root]# netstat -ln
```

```
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address
State
tcp        0      0 0.0.0.0:80              0.0.0.0:*
LISTEN
tcp        0      0 0.0.0.0:443             0.0.0.0:*
LISTEN
... ...
[root@web1 root]#
```

3. 방화벽과 L4 Switch 장비 설정 확인
고객님의 웹서버와 연계되어 설정된 방화벽 장비와 L4 Switch 장비의 설정을 80 포트 설정된 것 같이 443 포트에도 설정되어야 합니다.

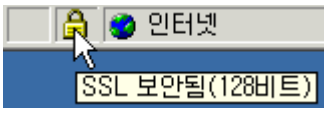
4. HTTPS 보안 통신으로 페이지를 확인
웹 브라우져를 통해서 https://[인증받은 도메인]/[테스트페이지].html 페이지를 확인해 봅니다.



[보안정보 확인창]

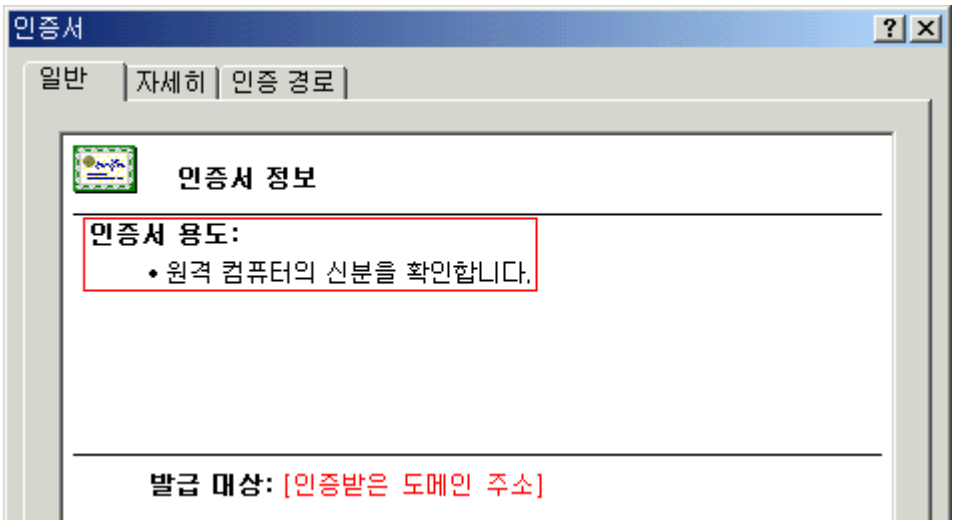
접속시에 위와 같은 보안정보 확인 창을 보실 수도 있으나, 이것은 전체 페이지를 암호화 처리했을 경우에 몇몇 이미지나 object 태그의 codebase 부분에 절대경로가 설정된 경우에 보안 정보를 나타내는 정보 창이므로, [아니오]를 선택합니다.

그러면, 웹 브라우져 하단 상태표시줄에 HTTPS 암호화 상태를 나타내는 노란 자물쇠를 확인해 보실 수 있습니다.



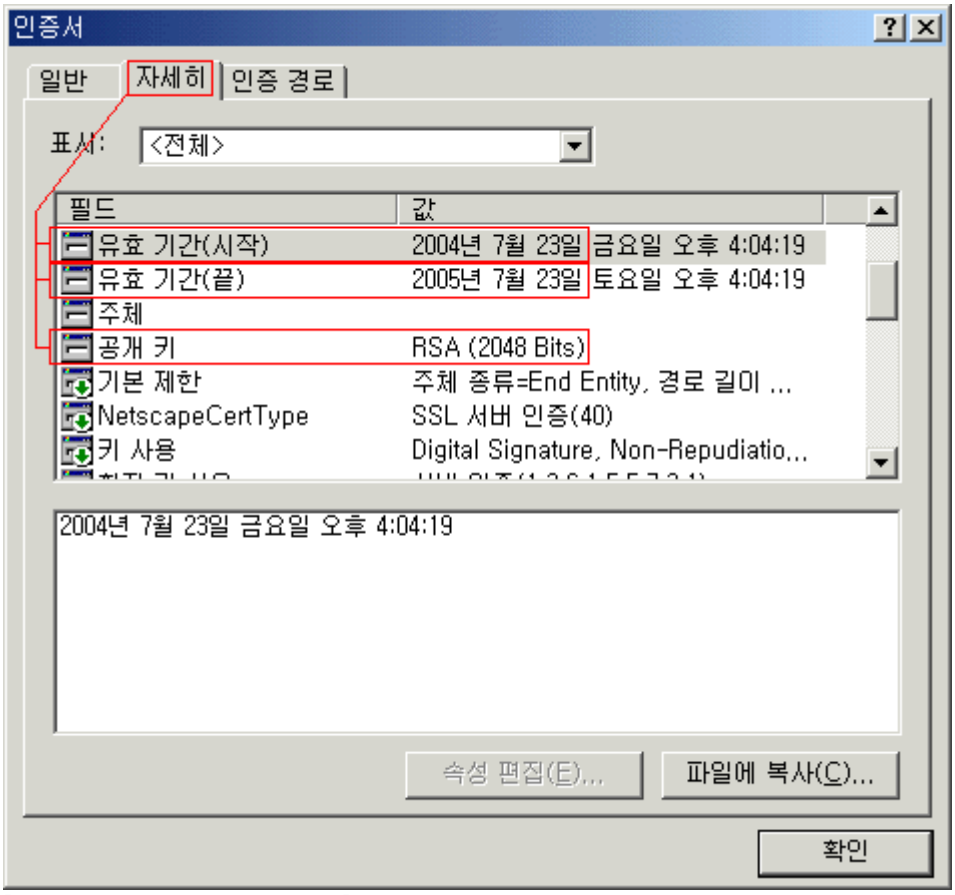
[SSL 128 bit 확인]

노란 자물쇠에 마우스 포인터를 가르키고 잠시 기다리면 128 bit 암호화 처리 상태를 확인해 보실 수 있습니다.
그리고 노란 자물쇠를 double click 하면, 서버에 설치된 인증서를 확인해 보실 수 있습니다.



[인증서 보기 - 일반]

인증서 용도가 [● 원격 컴퓨터의 신분을 확인합니다.] 라는 웹서버 인증서로 설정된 것을 확인할 수 있습니다.
다음으로 인증서 자세히 보기 부분에서 설치된 웹서버 인증서의 유효기간을 확인합니다. (유효 기간(시작)과 유효 기간(끝) 정보를 확인합니다.) 그리고 공개키 부분에서 RSA (2048 Bits) 를 확인합니다.



[인증서 보기 - 자세히]

## 6. 인증서 백업해 두기

개인키 파일, 웹서버 인증서, 체인 인증서를 백업해 둡니다.
<인증서는 개인키와 함께 꼭 백업을 해두셔야 하며, 백업을 하지 않아 발생하는 문제에 대해서는 재발급 비용이 추가될 수 있습니다.>