Building a Secure RedHat Apache Server HOWTO

Sigle Richard

richard.sigle@equifax.com

도운이

서정룡

s_ryong@hotmail.com

영문 버전 **: 0.1 2001-02-6**

최종수정일 **: 0.1 2001**년 **3**월 **19**일

이 지침은 PKI와 SSL이 함께 작동하는 방법을 설명하기 위한 것으로 보안 서버를 성공적으로 설치하기 위해서는 SSL 프로토콜의 작동 원리를 이해하는 것이 필수적이다.

1. 지침의 목적/범위

이 지침의 목적은 레드햇 리눅스 사용자들에게 아파치 웹서버를 사용해 서버 (SSL) 인증서를 설치하는데 있어 도움을 주기 위한 것으로 시간뿐만아니라 많은 경우 비용을 절약할 수 있는 명백한 절차를 제공하는 것이다.

우선 SSL 프로토콜과 디지털 인증서(digital certificate)에 관해 알아야 할 사항을 다룰 것인데 저자의 경험에 비추면 ModSSL 및 OpenSSL과 함께 아파치 웹서버를 구축하는 것이 가장 유익하다. OpenSSL은 SSL v2/v3와 TLS v1 프로토콜을 지원하는 범용 암호법 라이브러리이고 ModSSL은 아파치와 OpenSSL사이의 인터페이스로 작용하도록 설계된 아파치 API 모듈이다. 물론 가장 큰 장점은 세가지 소프트웨어 패키지 모두 ’free"라는 것이다.

4.1절부터 시작하여 ModSSL과 OpenSSL과 함께 컴파일된 레드햇 아파치 서버에 키 생성 및 인증서 설치의 단계적 절차를 자세히 검토할 것이다. 4절의 절차는 아파치와 밀접하게 관련된 Stronghold와 Raven과 같은 상용 SSL-서버 패키지에서도 또한 작용할 것이다.

Disclaimer: I am a technical support engineer for Equifax Secure Inc., a Certificate Authority. Therefore, I use Equifax Secure certificates and examples geared towards installing Equifax Secure certificates. However, the instructions will also work with certificates issued by other Certificate

Authorities. Since this document was written at my own initiative, Equifax Secure Inc. is neither liable nor accountable for any consequences resulting from the use of these procedures.

My comments to the reader is in this style (emphasized).

Example lines are in plain roman style.

Note that extra comments and advice is found in comments within the SGML source.

1.1. Secure Sockets Layer (SSL)에 대해

SSL은 TCP와 애플리케이션 계층 사이에 존재하는 presentation 계층 서비스 (OSI 7 계층)로 플랫폼과 애플리케이션에 독립적이다. SSL은 클라이언트와 서버사이의 안전한 통신 채널 관리를 담당하며 이들 사이에 전달되는 데이터를 암호하는데 있어 강력한 기구를 제공한다.

1.2. 피드백

이 지침에 대한 의견을 저자에게 보내주기 바란다 (richard.sigle@equifax.com).

1.3. Copyrights and Trademarks

Please freely copy and distribute this document in any format. It's requested that corrections and/or comments be forwarded to the document maintainer. You may create a derivative work and distribute it provided that you:

- Send your derivative work (in the most suitable format such as sgml) to the LDP (http://www.linuxdoc.org) (Linux Documentation Project) or the like for posting on the Internet. If not the LDP, then let the LDP know where it is available.
- License the derivative work with this same license or use GPL. Include a copyright notice and at least a pointer to the license used.
- 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.

1.4. Acknowledgements and Thanks

I would like to thank Tony Villasenor for tirelessly reading my drafts and offering his input and advice. Without Tony, this document would never have been finished.

2. Secure Sockets Layer/Private Key Infrastructure 소개

PKI는 클라이언트들에게 보내지는 공개키와 서버에 지역적으로 존재하는 비밀키로 구성되는 비대칭 키 시스템(**asymmetric key system**)으로 클라이언트와 서버 모두 암호화/복호화에 동일한 키를 사용하는 대칭 키 시스템(**symmetric key system**)과는 다르다.

2.1. SSL/PKI의 책임

SSL은 신용카드 정보, 의료 기록, 법률 문서와 전자 상거래 애플리케이션과 같은 가장 기밀을 다루는 트랜잭션(transaction)들의 전송에 이용되는 것을 허용할 수 있도록 하는 요건을 실현하기 위한 것이다. 각각의 애플리케이션은 처리될 트랜잭션의 중요도와 가치에 따라 다음 기준중 모두 또는 일부를 이용하려고 할 것이다.

기밀성 (Privacy)

가령 A로부터 B로의 전송을 목적으로 메세지가 암호화되었다고 가정하자. 메세지를 암호화하기 위해 A가 B의 공개키를 사용한다면 B는 자신의 비밀키를 이용하여 이 메세지를 복호화해서 해독할 수 있는 유일한 사람일 것이다. 그러나 A가 자신이 주장하는 누구라는 것을 확신할 수는 없다.

신뢰성 (Authenticity)

A가 자신이 주장하는 누구라는 것을 확신하기 위해 보증된 신뢰성을 원하는데 이는 약간은 더욱 복잡한 코딩 프로세스를 필요로 한다. 우선 B로 송신되는 A의 메세지는 A의 비밀키로 암호화된 후 B의 공개키로 암호화된다. B는 이제 우선 자신의 비밀키로 메세지를 복호화한 후 A의 공개키로 복호화해야 한다. 그래서 B는 어느 누구도 A의 비밀키로 암호화된 메세지를 생성할 수 없기 때문에 A가 자신이 주장하는 누구라는 것을 확신할 수 있다. SSL은 인증서를 사용하여 이를 달성하는데(PKI) 인증서는 인증서 발급기관(Certificate Authority, CA)과 같은 중립적인 제 삼자에 의해 발급되며 인증된 기관의 공개키외에 디지털 서명(Digital Signature)과/또는 time stamp를 포함한다. 자필 서명(Self-signed) 인증서는 SSL 도구를 사용하여 어느 누구라도 생성할 수 있지만 이는 공통적으로 존중되는 삼자에 의해 수행되는 인가로서의 영향력은 부족하다.

무결성 (Integrity)

SSL에서 자료 무결성은 필요한 해쉬 테이블 함수를 갖는 MAC(Message Authentication Code)를 이용하여 보장된다. 메세지 생성후 해쉬 함수를 이용하여 MAC이 얻어지며 이것이 메세지에 첨가된다. 메세지가 수신된 후 그 유효성은 수신 메세지로부터 계산된 새로운 MAC와 메세지에 덧붙여진 MAC와 비교하여 검사된다. 이러한 방법을 통해 제 삼자에 의해 메세지가 변경되었는지의 여부를 즉각적으로 알 수 있다.

부인 방지 (Non-repudiation)

부인 방지는 온라인 트랜잭션 중에 송수신자 서로를 보호하는데 특정 정보의 송신 사실을 부정하지 못하게 한다. 또한 트랜잭션이 이루어진 후 이의 변경을 허용하지 않으며 디지털 부인 방지는 일반적인 의미로 계약 체결과 동일하다.

2.2. 어떻게 SSL이 작동하는가

SSL 프로토콜은 SSL 레코드 프로토콜과 SSL 핸드쉐이크 프로토콜 두개의 하위 프로토콜을 포함한다. SSL 레코트 프로토콜은 데이터를 전송하는데 사용되는 포맷을 정의하며 SSL 핸드쉐이크 프로토콜은 SSL이 동작하는 서버와 클라이언트가 처음 SSL 연결을 맺을때 이들 사이에 일련의 메세지들을 교환하기 위해 SSL 레코드 프로토콜을 사용하는 것을 포함한다. 메세지 교환은 다음 기능들을 수월하게 하기 위해 설계되어 있다:

- 클라이언트에 서버를 인증한다. 서버 인증이 손상되지 않았고 신뢰 사슬(**chain of trust**)이 확립되었음을 보증하기 위해 서버 인증서는 CA에 의해 서명된다.
- 클라이언트와 서버 둘 모두가 지원하는 암호화 알고리듬 또는 암호(cipher) 선택을 허용한다.
- 임의로 서버에 클라이언트를 인증한다.
- 공유 비밀을 생성하기 위해 공개키 암호화 기법을 사용한다.
- 암호화된 SSL 연결을 확립한다.

SSL 핸드쉐이크 프로토콜

핸드쉐이크 프로토콜은 클라이언트와 서버의 상태를 통합하기 위해 사용되는데, 핸드쉐이크 중 다음 이벤트가 발생한다:

- 클라이언트와 서버 사이에 인증서가 교환된다(비대칭 키들). 서버가 클라이언트에 자신의 공개키를 보내는데 서버가 인증서를 통해 클라이언트 인증을 검증하도록 설정되어 있다면 클라이언트는 서버에 자신의 공개키를 보낸다. 인증서의 유효 날짜가 검증되며 신뢰받는 CA의 디지털 서명인지 검사되는데 유효 날짜와/또는 디지털 서명이 옳지 않다면 브라우저가 사용자에게 경고 메세지를 나타낼 것이다. 그리고 나서 인증서 보유자임을 확신하기 위해 사용자에게 옵션을 준다.
- 곧이어 클라이언트가 랜덤키(대칭키)를 생성하는데 랜덤키는 암호화와 MAC 계산을 위해 사용될 것이다. 그것들은 서버의 공개키를 사용하여 암호화되어 서버에 보내지는데 단지 서버만이 새로운 랜텀키를 복호화할 수 있다. 새로운 대칭키는 클라이언트와 서버사이에 보내지는 데이터를 암호화하는데 사용된다.

  Note: 서버-브라우저 인증 후의 대칭키 사용으로 인해 효율 성능은 대단히 향상된다.
- 데이터 무결성을 위해 메시지 암호화 알고리듬과 해쉬 함수를 협의해서 결정한다. 이 협의(negotiation) 프로세스가 수행되어 클라이언트는 지원되는 알고리듬 목록을 서버에 건네주며 다음에 양쪽 모두에 이용할 수 있는 가장 강력한 암호를 선택한다. 선택된 암호화 알고리듬과 해쉬 함수 식별자는 레코드 프로토콜이 사용하는 현재 상태의 암호 스펙 필드에 저장된다.
- 프로토콜 버전, 세션 ID, Cipher Suite, 압축 방법과 두개의 임의 값인 ClientHello.random과 ServerHello.random 들과 같은 필드들은 핸드쉐이킹 동안에 설정된다.

Note: 각각의 SSL 연결을 위해 IP 주소가 필요한데 가상 호스트 이름이 애플리케이션 계층에서 분석된다. SSL이 애플리케이션 계층 아래에 존재함을 기억해라.

세션 키 (대칭 코드)

- 40 비트, 원래 export를 위해서만 사용
- 56 비트, DES 에서 사용
- 64 비트, CAST 에서 사용, 56 비트보다 256배 강력
- 80 비트, CAST에서 사용, 현재 기술로 해독할 수 없으며 56 비트보다 160만배 강력
- 128 비트, CAST 또는 RC2 에서 사용, 현재 및 가까운 미래에 완전한 키 검색이 불가능

공개/개인 키 쌍(비대칭 코드)

- 512 비트
- 768 비트
- 1024 비트
- 2048 비트

2.3. 어떻게 PKI가 작동하는가

클라이언트와 서버는 각자 공개키와 비밀키를 갖는다 (클라이언트가 인증서를 갖고 있지 않고 서버가 인증서를 요청하지 않는다면 클라이언트의 브라우저가 SSL 세션을 위해 임의로 한쌍의 키를 생성한다).

송신자는 메세지를 암호화하기 위해 자신의 비밀키를 사용하는데 이것이 메세지의 출처를 인증한다. 결과적으로 생긴 암호문은 수신자의 공개키를 이용해 한 번 더 암호화되는데 단지 수신자만이 자신의 비밀키를 사용하여 메세지의 최초 복호화를 할 수 있기때문에 기밀성을 제공한다. 수신자는 암호화된 메세지를 더욱 복호화하기 위해 송신자의 공개키를 사용한다. 송신자만이 그 비밀키를 액세스하기 때문에 수신자는 암호화된 메세지가 송신자가 보냈다는 것을 확신한다.

메세지 다이제스트(digest)는 쌍방 또는 제 삼자가 어떤 방식으로든 메세지에 손을 대거나 변경하지 않았다는 것을 보증하기 위해 사용된다. 메세지 다이제스트는 메세지에 해쉬 함수(지문로 알려진 비밀키의 일부)를 적용함으로써 얻어지며 다이제스트(이제 서명으로 알려진)가 메세지에 첨부 또는 첨가된다. 서명의 길이는 일정(파일 크기에 무관하게)하며 비밀키가 함유하는 메세지 다이제스트의 유형에 의존한다(md5-128비트, sha1- 160 비트 등등). 메세지 중 단 한개의 비트라도 변경된다면 서명의 길이는 변경될 것이고 결국 메세지가 변경되었음을 입증한다.

2.4. 인증서(x509 Standard)

디지털 인증서는 인터넷상에서 참여자(entity)를 신뢰할 수 있게 하는데 이는 중립적인 제 삼의 CA에 의해 입증된 사용자의 credential 을 포함한다.

데이터를 해독할 수 없는 형태로 암호화하기 위해 수학적 알고리듬과 값(키)이 사용되며 두 번째 키가 보충(complementary) 알고리듬과 그 관련 값을 이용하여 데이터를 복호화하기 위해 사용된다. 이 두키는 관련된 값을 포함해야 하는데 키쌍(**key pair**)으로 알려져 있다.

Note: ITU-T 권고 X.509 [CCI88c]는 X.509 인증서 구문뿐만 아니라 X.500<65533> 디렉토리 대한 인증 서비스를 지정한다. 인증서는 사용자(subject) 이름과 공개키간의 바인딩을 인증하기 위해 발급자에 의해 서명된다. SSLv3은 1994년에 채택되었는데 버전 2와 3의 주요 차이점은 확장(extension) 필드가 추가되었다는 것이다. 이 필드는 키와 이름 바인딩외에 부수적인 정보를 전달할 수 있기 때문에 더욱 융통성을 준다. 표준 확장은 사용자와 발급자 속성, 인가 정책 정보와 키 사용 제한을 포함한다.

X.509 인증서는 다음 필드로 구성된다:

- 버전
- 시리얼 넘버
- 서명 알고리듬 ID
- 발급자 이름
- 유효 기간
- 사용자(subject) 이름
- 사용자 공개키 정보
- 발급자 고유 식별자 (버전 2와 3에 해당)
- 사용자 고유 식별자 (버전 2와 3에 해당)
- 확장(extension, 버전 3에 해당)
- 위 필드에 대한 서명

2.5. 디지털 인증서 비밀키

비밀키는 디지털 인증서내에 덧붙여지지 않으며 어떤 서버 정보도 포함하지 않는다. 비밀키는 암호화 정보와 지문을 포함하는데 시스템내에 지역적으로 생성되며 안전한 환경내에 유지되어야 한다. 비밀키가 손상된다면 범죄자가 반드시 보안 시스템에 대한 코드를 갖는다. 클라이언트와 서버사이의 전송이 도청 및 복호화될 수 있다. 이러한 유형의 침입 가능성때문에 삼중 DES 기법을 이용하여 암호화되는 비밀키 생성이 추천되는데 파일은 정확한 pass phrase 없이는 거의 사용이 불가능하도록 암호화되고 패스워드가 보호된다.

트랜잭션의 보안은 비밀키에 의존하는데 이 비밀키가 잘못된 사람에게 누출된다면 누구라도 이를 쉽게 복제해서 보안을 손상시키기 위해 사용할 수 있다. 키의 손상은 서버가 비양심적인 해커에 의해 도청 및 조작되었음을 의미하는 메세지를 생성할 것이다. 완벽한 보안 시스템은 사칭자 탐지 및 키 복제 방지를 할 수 있어야 한다.

2.6. 디지털 인증서 공개키

공개키는 디지털 인증서내에 덧붙여지는데 이는 보안 연결이 요청될 때 서버에서 클라이언트로 보내진다. 이 프로세스는 인증서를 사용하는 서버를 식별한다. 공개키는 무결성, 신뢰성을 인가하며 비밀스런 데이터 전송을 생성하기 위해 데이터를 암호화하는데 사용된다.

2.7. 인증서 서명 요청(Certificate Signing Request,CSR)

CSR(인증서 서명 요청)은 인증서를 생성하기 위한 CA가 필요로 하는 정보를 포함하는데 비밀키의 보충 알고리듬, 공통값 및 서버를 식별하는 정보 들의 암호화된 버전을 포함한다. 이 정보는 국가, 주, 조직, 공통 이름(도메인 이름)과 연락 정보를 포함하며 이에 국한되어 있지 않다.

3. 인증서 관련 작업

다음 절은 비밀키 파일, CSR 및 자필 서명 인증서를 생성하는데 포함된 단계들을 다룬다. CA가 서명한 인증서를 얻으려면 **CSR**을 생성할 필요가 있으며 그렇지 않은 경우 자필 서명 인증서를 생성할 수 있다.

3.1. 비밀키 생성하기

비밀키를 만들기 위해서는 OpenSSL 툴킷을 아파치와 함께 설치 및 설정해야 한다. 다음 예는 디폴트로 /usr/local/ssl/bin 디렉토리내에 설치된 OpenSSL command line 도구를 사용하는데 이 도구를 포함하는 디렉토리가 $PATH 변수에 추가되어 있다고 가정한다.

삼중 des 암호화 표준(추천된다)을 사용해 비밀키를 생성하려면 다음 명령을 실행시킨다:

```
openssl genrsa -des3 -out filename.key 1024
```

pass phrase를 입력 및 재입력하라는 지시 메세지를 볼 것이다. 삼중 des 암호화를 사용한다고 선택한다면 cold start로 SSL 서버를 시작할 때마다 패스워드를 묻는 지시 메세지를 볼 것이다 (restart 명령을 사용할 때는 이러한 메세지를 보지 못할 것이다). 어떤 사람은 패스워드 프롬프트을 귀찮게 생각할 수 있는데 특히 휴식 시간에 시스템을 시동할 필요가 있는 경우가 그렇다. 또는 시스템이 이미 충분히 안전하다고 믿을 수 있기 때문에 패스워드 프롬프트가 나타나지 않도록 한다면(따라서 삼중 des 암호화가 아니다) 아래의 명령을 사용해라. 오히려 단지 512 비트 키를 생성하려고 한다면 명령 끝부분의 1024를 생략해라. OpenSSL은 디폴트로 512 비트가 될 것이다. 더욱 작은 키를 사용한다면 약간 빠르겠지만 더욱 보안에 취약하다.

삼중 des 암호화를 사용하지 않고 비밀키를 생성하려면 다음 명령을 실행시킨다:

```
openssl genrsa -out filename.key 1024
```

기존 비밀키에 패스워드를 추가하려면 다음 명령을 실행시킨다:

```
openssl -in out filename.key -des3 -out newfilename.key
```

기존 비밀키로부터 패스워드를 제거하려면 다음 명령을 실행시킨다:

```
openssl -in filename.key -out newfilename.key
```

Note: 특별히 지정되지 않는다면 비밀키는 현재 디렉토리내에 생성될 것이다. 이를 다루는 손쉬운 세가지 방법이 있는데 OpenSSL이 경로에 있다면 키 파일을 저장하도록 명시한 디렉토리(RPM 또는 소스 파일을 사용해 아파치를 설치했다면 각각 /etc/httpd/conf/ssl.key 또는 /usr/local/apache/conf/ssl.key 가 디폴트이다)에서 이를 실행시킬 수 있다. 다른 방법은 생성된 디렉토리에서 정확한 디렉토리로 파일을 복사하는 것이다. 마지막으로 특히 명령을 실행시킬 때 (예를들면 openssl genrsa -out /etc/httpd/conf/ssl.key/filename.key 1024) 경로를 지정할 수도 있다. 어떤 방법을 사용하든 별 문제는 없다.

OpenSSL 툴킷에 대해 더 많은 정보를 얻기 위해서 OpenSSL WebSite (http://www.openssl.org)를 참조해라.

3.2. CSR 생성하기

CA가 서명한 인증서를 얻기 위해서는 CSR을 생성할 필요가 있다. 이 목적은 전체 비밀키를 보내거나 모든 기밀 정보를 손상시키지 않고 인증서를 생성할 수 있을만큼 충분한 정보를 CA에 보내려는 것인데 CSR은 도메인 이름, 소재지 정보 등과 같은 인증서에 포함될 수 있는 정보를 포함한다.

- CSR을 생성하려는 비밀키 위치를 결정하고 다음 명령을 실행시킨다:

  ```
  openssl req -new -key filename.key -out filename.csr
  ```

- 소재지 정보, 공통 이름(도메인 네임), 조직 정보 등에 대한 지시 메세지를 볼 것이다. 필수 필드와 무효한 엔트리에 관한 정보에 대해 신청하려는 CA에 문의해라.
- CSR을 지시에 따라 CA에 보내라.
- 새로운 인증서를 기다리거나 자필 서명 인증서를 생성해라. CA로 부터 인증서를 받을 때까지 자필 서명 인증서를 사용할 수 있다.

Note: 비밀키 생성과 요청을 동시에 하기 위해 다음 명령을 실행시킨다:

```
openssl genrsa -des3 -put filename.key 1024
```

3.3. 자필 서명 인증서 생성하기

CA가 서명한 인증서를 얻으려 한다면 자필 서명 인증서를 생성하는 것은 필요하지 않지만 이는 매우 간단하다. 필요한 것은 비밀키와 보호하려고 하는 서버 이름(fully qualified domain name)이다. 소재지 정보, 공통 이름(도메인 네임), 조직 정보 등에 대한 지시 메세지를 볼 수 있는데 OpenSSL은 여기서 많은 자유를 준다. 인증서가 정확히 작동되기 위해 필요한 필드는 도

메인 네임 필드로 이 필드가 없거나 부정확하다면 브라우저로부터 **Certificate Name Check**이라는 경고 메세지를 받을 것이다.

자필 서명 인증서를 생성하기 위해서는 다음 명령을 실행시킨다:

```
openssl req -new -key filename.key -x509 -out filename.crt
```

3.4. 웹서버 인증서 설치하기

지금까지 지시들을 잘 따랐다면 이 시점에서 아무 문제도 없어야 한다. CSR을 CA에 보내고 인증서를 아직까지 받지 못했다면 잠시 쉴 수 있을 것이다! 자필 서명 인증서를 사용하거나 인증서를 받았다면 다음을 계속할 수 있다.

- 사용하기로 결정한 비밀키 파일이 디렉토리내에 존재하는지 확인해라. 다음 예는 레드햇 배포판의 RPM 설치시의 디폴트 /etc/httpd/conf/ssl.key 에 기초할 것이다.
- CA가 서명한 또는 자필 서명 인증서가 명시한 위치에 존재하는지 확인해라. RPM 설치시의 디폴트 /etc/httpd/conf/ssl.crt를 사용할 것이다. 이 위치에 없다면 인증서를 이곳에 놓는다.
- 설치된 intermediate(root) 인증서가 있다면 이를 /etc/httpd/conf/ssl.crt 디렉토리에 복사한다.
- 이제 httpd.conf 파일을 편집해야 하는데 다음 단계, 4절로 가기 전에 이 파일을 백업한다.

4. 아파치 서버 설정하기

SSL을 지원하기 위해서 추가 API 모듈과 함께 아파치가 설정되어야 한다. 많은 SSL 소프트웨어 패키지를 이용할 수 있는데 이 문서는 ModSSL과 OpenSSL에 기초한다. 이 제품을 지원하는데 도움이 되는 무수히 많은 메일링 리스트와 뉴스그룹이 있는데 아파치 웹서버에 기초한 상용 SSL 소프트웨어 패키지에 대해서도 이 문서가 도움이 될 것이다.

명실해야 할 사항: 동일한 서버에 다중 가상 호스트를 만들 수 있는데 동일 IP 주소로 매우 많은 이름을 갖는 가상 호스트를 만들 수 있다. 그러나 동일한 IP 주소로 여러개의 보안 가상 호스트를 만들 수는 없으며 서로 다른 이름을 갖는 가상 호스트와 단 하나의 보안 가상 호스트를 만들 수 있다. 이렇게 많은 가상 호스트를 가질 수 있는 이유는 SSL이 애플리케이션 계층 아래서 작동하기 때문인데 애플리케이션 계층이 정의된 후 이름을 갖는 호스트가 정의된다.

구체적으로 동일한 소켓(IP 주소 + 포트)에 여러개의 보안 가상 호스트를 만들 수 없으며 보안 호스트는 포트 443을 사용할 것이다. 동일한 IP에서 다른 포트를 사용하기 위해, 따라서 다른 소켓을 만들기 위해 가상 호스트 설정을 변경할 수 있는데 이 접근 방법에는 많은 단점이 있다. 가장 명백한 단점은 디폴트 포트를 사용하지 않을 경우 보안 사이트에 액세스하기 위해 URL에 포트 넘버까지 포함시켜야 한다는 것이다.

예:

- 디폴트 포트를 사용하는 www.something.com 사이트는 https://www.something.com으로 접속할 수 있다.
- 포트 8888을 사용하는 사이트는 https://www.something.com:8888으로 접속할 수 있다.

다른 단점은 포트를 더 도입할 경우 포트를 탐지하는 해커에 더욱 많은 침입 기회를 제공할 수 있다는 것이다. 마지막으로 어떤 다른 서비스에 의해 사용되는 포트를 선택할 경우 충돌 문제가 생길 수 있다.

4.1. 보안 가상 호스트 정의하기

가상 호스트 설정은 상당히 수월한데 보안 가상 호스트 설정의 기초를 자세히 살펴볼 것이다.

다음 예에서 .crt 와 .key 파일 확장자를 사용하는데 다양한 파일들과 구별하기 위한 개인적인 방식이다. 아파치에서는 선택한 모든 확장자를 사용할 수 있으며 확장자가 없어도 무방하다.

모든 보안 가상 호스트들은 대개 httpd.conf 파일의 끝부분에 위치한 <IfDefineSSL>와 </IfDefineSSL> 사이에 포함되어야 한다.

```
<VirtualHost 172.18.116.42:443>
DocumentRoot /etc/httpd/htdocs
ServerName www.somewhere.com
ServerAdmin someone@somewhere.com
ErrorLog /etc/httpd/logs/error_log
TransferLog /etc/httpd/logs/access_log
SSLEngine on
SSLCertificateFile /etc/httpd/conf/ssl.crt/server.crt
SSLCertificateKeyFile /etc/httpd/conf/ssl.key/server.key
SSLCACertificateFile /etc/httpd/conf/ssl.crt/ca-bundle.crt
<Files ~ "\.(cgi|shtml)$">
      SSLOptions +StdEnvVars
</Files>
<Directory "/etc/httpd/cgi-bin">
      SSLOptions +StdEnvVars
</Directory>
SetEnvIf User-Agent ".*MSIE.*" nokeepalive ssl-unclean-shutdown
CustomLog /etc/httpd/logs/ssl_request_log \
         "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"
</VirtualHost>
```

SSL에 대한 가장 중요한 지시는 SSLEngine on, SSLCertiFficateFile, SSLCertificateKeyFile과 많은 경우에 있어 SSLCACertificateFile 지시이다.

SSL 엔진

"SSLEngine on" - 이 지시는 SSL을 구동하는 ModSSL의 명령이다.

SSLCertificateFile

SSLCertificateFile은 인증서 위치와 그 이름을 아파치에게 알려준다. 위 예에서 인증성 파일 이름은 "server.crt"로 ModSSL 설정시 추가되는 디폴트이다. 저자 개인적으로는 디폴트 이름 사용을 추천하지 않는데 얼마간의 낭패를 피하고 인증서 이름을 servername.crt(domainname.crt)로 하라. 또한 디폴트 /etc/httpd/conf/ssl.crt 또는 /usr/local/apache/conf/ssl.crt 가 아닌 다른 디렉토리를 사용할 수 있는데 경로 변경한 것을 꼭 기억해라.

SSLCertificateKeyFile

SSLCertificateKeyFile은 비밀키 이름가 그 위치를 아파치에게 알려주는데 여기서 정의된 디렉토리는 단지 루트에게만 읽기/쓰기 허가권이 주어져야 하며 다른 누구도 이 디렉토리에 액세스하지 못해야 한다.

SSLCACertificateFile

SSLCACertificateFile지시는 Intermediate(root) 인증서 위치를 아파치에게 말해주는데 사용하는 인증서에 따라 필요할 수도 있고 아닐 수도 있다. 이 인증서는 반드시 신뢰 고리(ring of trust)이다.

Intermdiate 인증서 - CA는 사용자와 동일한 방식으로 인증서를 얻는데 이것이 intermediate 인증서이다. 이는 기본적으로 intermediate 인증서 보유자가 그들이 말하는 CA이고 고객에게 인증서 발급이 인가된 기관임을 말한다. 웹브라우저는 각각의 릴리스와 함께 갱신된 신뢰받는 CA의 리스트를 갖고 있다. CA가 너무 신규 기관이라면 브라우저의 신뢰받는 CA 리스트에 없을 수 있다. 이를 대부분의 사람들이 자주 브라우저를 갱신하지 않는다는 사실과 결부시킨다면 CA가 자동적으로 신뢰받는 CA 라고 승인받을 때까지 수년이 걸릴 것이다. 이에 대한 해결 방안이 SSLCACertificateFile 지시를 사용하여 서버에 intermediate 인증서를 설치하는 것이다. 보통은 신뢰받는 CA가 intermediate 인증서를 발급하는데 그렇지 않다면 SSLCACertificateFile 지시를 사용할 필요가 있을 수 있다 (있을 법하지 않음에도 불구하고).

4.2. 인증서 예

서버 인증서 파일

```
-----BEGIN CERTIFICATE-----
MIIC8DCCAlmgAwIBAgIBEDANBgkqhkiG9w0BAQQFADCBxDELMAkGA1UEBhMCWkEx
FTATBgNVBAgTDFdlc3Rlcm4gQ2FwZTESMBAGA1UEBxMJQ2FwZSBUb3duMR0wGwYD
VQQKExRUaGF3dGUgQ29uc3VsdGluZyBjYzEoMCYGA1UECxMfQ2VydGlmaWNhdGlv
biBTZXJ2aWNlcyBEaXZpc2lvbjEZMBcGA1UEAxMQVGhhd3RlIFNlcnZlciBDQTEm
MCQGCSqGSIb3DQEJARYXc2VydmVyLWNlcnRzQHRoYXd0ZS5jb20wHhcNOTkwNTI1
MDMwMDAwWhcNMDIwNjEwMDMwMDAwWjBTMQswCQYDVQQGEwJVUzEbMBkGA1UEChMS
RXF1aWZheCBTZWN1cmUgSW5jMScwJQYDVQQDEx5FcXVpZmF4IFNlY3VyZSBFLUJ1
c2luZXNzIENBLTIwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMYna8GjS9mG
q4Cb8L0VwDBMZ+ztPI05urQb8F0t1Dp4I3gOFUs2WZJJv9Y1zCFwQbQbfJuBuXmZ
QKIZJOw3jwPbfcvoTyqQhM0Yyb1YzgM2ghuv8Zz/+LYrjBo2yrmf86zvMhDVOD7z
dhDzyTxCh5F6+K6Mcmmar+ncFMmIum2bAgMBAAGjYjBgMBIGA1UdEwEB/wQIMAYB
Af8CAQAwSgYDVR0lBEMwQQYIKwYBBQUHAwEGCCsGAQUFBwMDBgorBgEEAYI3CgMD
BglghkgBhvhCBAEGCCsGAQUFBwMIBgorBgEEAYI3CgMCMA0GCSqGSIb3DQEBBAUA
A4GBALIfbC0RQ9g4Zxf/Y8IA2jWm8Tt+jvFWPt5wT3n5k0orRAvbmTROVPHGSLw7
```

```
oMNeapH1eRG5yn+erwqYazcoFXJ6AsIC5WUjAnClsSrHBCAnEn6rDU080F38xIQ3
j1FBvwMOxAq/JR5eZZcBHlSpJad88Twfd7E+0fQcqgk+nnjH
-----END CERTIFICATE-----
```

인증서 파일 내용

```
   Certificate:
      Data:
        Version: 3 (0x2)
        Serial Number: 1516 (0x5ec)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=US, O=Equifax Secure Inc, CN=Equifax Secure E-
Business CA
        Validity
          Not Before: Jul 12 15:21:01 2000 GMT
          Not After : Jun  2 22:42:34 2001 GMT
        Sub-
ject: C=us, ST=ga, L=atlanta, O=Equifax, OU=Rick, CN=172.18.116.44/Email=richard.sigle@equif
        Subject Public Key Info:
          Public Key Algorithm: rsaEncryption
          RSA Public Key: (1024 bit)
              Modulus (1024 bit):
                00:c8:eb:93:26:97:ca:00:ce:4c:e4:f3:fd:43:31:
                cd:53:ed:b4:8a:ad:93:84:dc:7a:48:39:b5:28:57:
                03:7f:a9:ac:3e:58:6a:7a:e3:52:3e:1e:52:58:a2:
                6f:23:ad:bb:84:d8:88:ed:6d:a5:da:08:6b:c8:6c:
                a5:4c:34:67:d8:46:1c:ca:20:50:b0:e8:54:7f:ca:
                5e:ef:09:ff:6e:8d:a6:2b:02:f5:54:0f:c2:d0:45:
                12:ad:66:e7:8b:dd:68:be:64:a4:9b:69:bd:a4:1a:
                5e:ef:09:ff:6e:8d:a6:2b:02:f5:54:0f:c2:d0:45:
                12:ad:66:e7:8b:dd:68:be:64:a4:9b:69:bd:a4:1a:
                5a:2f:3b:6e:73:84:d8:d6:17:bd:12:39:34:fa:3d:
                d8:a9:e8:59:3c:c2:61:c5:b3
              Exponent: 65537 (0x10001)
        X509v3 extensions:
          X509v3 Key Usage: critical
             Digital Signature, Non Repudiation, Key Encipher-
ment, Data Encipherment
          Netscape Cert Type:
             SSL Server
          X509v3 Authority Key Identifier:
             keyid:5B:E0:A8:75:1C:78:02:47:71:AB:CE:27:32:E7:24:88:42:28:48:56
      Signature Algorithm: md5WithRSAEncryption
        87:53:74:e9:e1:a6:10:56:8c:fa:63:0e:7b:72:ff:76:4b:79:
        0e:49:2a:58:ed:71:7a:bf:77:61:fa:e8:74:04:37:8c:d3:6a:
        9a:3d:80:76:7a:c3:64:30:e7:1b:40:25:4e:2a:81:8b:e5:ac:
        76:a4:38:67:cc:3f:93:43:e1:1d:c3:8d:ba:ed:cc:d7:aa:a4:
        ab:d3:84:77:7c:8f:26:f6:dd:ba:3b:6a:99:81:e1:9e:7e:0f:
        ca:a6:ff:c0:c3:59:6e:dc:a6:03:23:bf:8f:24:ff:15:ad:ac:
```

```
      0d:85:fc:38:bf:d1:24:2d:1a:d3:72:55:12:95:5f:65:f0:60:
      df:b1
```

비밀키 파일

```
-----BEGIN RSA PRIVATE KEY----- Proc-Type: 4,ENCRYPTED DEK-Info:
DES-EDE3-CBC,124F61450D85A480

ELz64SV+tFSRybsHjY9NH7CP7yDHXP6xcd9FY6MVgQykTkq2h0n7j+tmpfUPbStT
6jCgm/dTYM9mpkQ3jYZBALiVD5JNJ9t1dWisxQXY/nsak8LSTN7LhUtZSfk5xSmV
Zsl4gwQS20UdBzFiJ+4qDajP/pzocSdSuQvxIHq7UzNwJsW8UYxR3I1qrDgyNXKS
db41BWH4QdNtE0p+pi9VndDzXktqZGHEvtrQTV+39DV/dwOdnGBpYBETljMO5X6t
D42xcVs0Doa1vZ6PiMCkwFNPXsPlKHZtHwEL4I3CQdiH4E0oYh3klBzlXBY4YldN
A+s4xU44FpXp5xwt9nnVPUKHPo+NpdaRK7dAcRNO3GN3+ek1ggzvEjjuWKes3RQh
PlHPuF7VWo4KeaTfTIwJWfGxz4nvwlVByPJ6Z73Mn0VcDXCkVm6+h3PLlYL0FMqM
baUyQPpw6bhfW71FO/IIQxz3R1EqkxW7OHv74uuYl8kjHXf3S6qRZEGUG/zOGLGr
mI5s2qnU69HlBObFkc6WQq0QxMq4PiUi7HhCLMkH8+wBsNNMnb75+7lQKkEhdOeE
iUMKe5kgQqfd9w8jsBH5nu+J/nCfvPdp0isQW+P3/Rrh6YMwdKnlVfNZWdGiTzpQ
ngThAGq5lit4uf4zdTIYYrs+T9I5ltjj0KgCUD4VL5/7OfnR3gcphpbHXQf0E2cz
Qwq7q7ppKwCf/x92pHi8oVevlV5Dx9NQbGhEOA5pooqD6S2xZBbPLzkUKWDEO2il
oBZ5L1jClR5jjdF2U61w7aRrL0t6luDU/aRv/fcoYes=
-----END RSA PRIVATE KEY-----
```

비밀키 내용

```
read RSA key
Enter PEM pass phrase:
Private-Key: (1024 bit)
modulus:
    00:c8:eb:93:26:97:ca:00:ce:4c:e4:f3:fd:43:31:
    cd:53:ed:b4:8a:ad:93:84:dc:7a:48:39:b5:28:57:
    03:7f:a9:ac:3e:58:6a:7a:e3:52:3e:1e:52:58:a2:
    6f:23:ad:bb:84:d8:88:ed:6d:a5:da:08:6b:c8:6c:
    a5:4c:34:67:d8:46:1c:ca:20:50:b0:e8:54:7f:ca:
    5e:ef:09:ff:6e:8d:a6:2b:02:f5:54:0f:c2:d0:45:
    12:ad:66:e7:8b:dd:68:be:64:a4:9b:69:bd:a4:1a:
    5a:2f:3b:6e:73:84:d8:d6:17:bd:12:39:34:fa:3d:
    d8:a9:e8:59:3c:c2:61:c5:b3
publicExponent: 65537 (0x10001)
privateExponent:
    00:b6:57:7d:3b:58:24:1e:a9:1b:85:e9:9c:9e:5f:
    d3:3d:69:0c:21:93:37:bf:2b:2c:da:e1:6c:74:48:
    cb:c7:0f:60:5f:50:74:8a:44:45:be:54:5c:5d:4e:
    45:58:f6:f1:a8:b5:af:46:f2:ec:c2:bc:43:bd:28:
    44:b7:ad:13:d3:ca:de:59:24:e8:fa:f8:e5:5f:45:
    38:2c:a0:a3:de:98:13:d8:80:38:e1:47:53:4c:ea:
    e4:66:c3:82:93:89:c3:90:83:44:e1:13:4f:74:76:
    e2:c0:89:97:77:5f:33:d8:7d:27:21:52:55:c2:d7:
    dc:01:f9:bc:21:8d:a3:f5:c1
```

```
prime1:
    00:e3:2d:6b:5e:05:6b:e1:46:e6:ab:ae:f3:8b:d0:
    5f:94:5c:6f:f5:47:46:1d:4e:66:d3:7e:98:18:e0:
    2c:0d:08:ca:b7:29:72:af:53:62:30:ec:be:26:1f:
    cc:5a:ed:65:62:65:70:1e:18:19:61:e3:77:00:a7:
    3a:9e:4e:12:93
prime2:
    00:e2:69:56:78:e8:39:ff:17:db:cc:39:d7:7f:70:
    41:dc:c5:59:43:16:c1:84:4c:ae:e7:5d:8a:c5:4b:
    da:88:8e:03:99:7c:88:f2:8a:13:31:57:44:e0:b5:
    c8:0a:60:b0:05:de:f6:9e:f2:00:ec:37:21:8d:3b:
    dc:8e:c9:d4:61
exponent1:
    1a:ad:6a:be:4f:c4:ab:5f:b8:16:d1:24:a8:76:7f:
    c2:dc:58:09:65:a5:46:2b:be:c7:77:46:45:25:8e:
    06:b9:d1:94:50:b9:b6:fd:03:ba:db:12:39:47:e2:
    a7:8a:d9:2d:04:dc:75:ac:3e:ce:cf:f7:59:8c:49:
    c5:ed:45:21
exponent2:
    2d:4e:fd:32:06:ef:0c:40:7f:08:d8:8e:6a:7f:51:
    7e:d7:b3:6c:3c:92:8f:62:35:22:31:d3:02:76:92:
    8d:ff:35:73:32:bb:c9:25:9e:7f:a2:42:33:61:cd:
    5d:5e:49:fb:72:ca:11:b6:c6:3e:7f:2d:e4:b0:95:
    0b:b2:12:21
coefficient:
    50:52:09:22:cb:fb:b2:b8:58:85:ab:1d:82:b9:6e:
    d0:f6:dc:e8:ce:a6:5d:a1:ff:c8:4d:3b:2b:1c:19:
    64:f0:c4:4a:bc:b2:1d:2b:2d:09:59:83:a3:9a:89:
    f8:db:2c:2c:8a:bd:fd:a3:16:51:76:aa:ce:ea:85:
    6b:1c:9f:f7
```

4.3. 웹 서버 재구동하기

웹서버를 재구동할 스크립트는 /usr/local/sbin, /usr/bin (httpd 스크립트인 경우) 또는 /usr/local/apache/bin (apachectl 스크립트인 경우) 디렉토리에 위치할 수 있는데 SSL 기능과 함께 서버를 구동하고 있지 않다면 서버를 중지시킨후 구동해야 한다. 서버 구동, 재구동 및 정지를 위한 자신만의 개별화된 스크립트를 작성할 수 있는데 SSL 엔진을 시동시킨다면 무방하다.

명령은 다음과 같다:

```
httpd stop
httpd startssl
httpd restart
```

또는

```
   apachectl stop
   apachectl startssl
   apachectl restart
```

5. 문제해결

제기될 수 있는 다소의 공통되는 문제가 있다.

5.1. 서버는 구동된 듯 한데, 보안 사이트에 액세스 할 수 없다(Server Appears to start, but you cannot access the secure site).

error_log 파일을 체크해라. 에러 로그를 작성하도록 가상 호스트를 설정하지 않았다면 이를 다시 고려하고 싶을 수 있다. 예제 SSL 가상 호스트는 에러 로그 파일을 작성하는데 아마도 대부분 로그 끝부분에 비밀키가 인증서와 일치하지 않는다는 것을 말하는 약간의 경고들과 에러가 있을 것이다.

예:

```
   [Tue Nov 21 09:09:02 2000] [notice] Apache/1.3.14 (Unix) mod_ssl/2.7.1
   OpenSSL/0.9.6 configured -- resuming normal operations
   [Tue Nov 21 09:09:16 2000] [notice] caught SIGTERM, shutting down
   [Tue Nov 21 14:39:54 2000] [notice] Apache/1.3.14 (Unix) mod_ssl/2.7.1
   OpenSSL/0.9.6 configured -- resuming normal operations
   [Tue Nov 21 14:40:31 2000] [notice] caught SIGTERM, shutting down
   [Tue Nov 21 14:43:53 2000] [er-
ror] mod_ssl: Init: (esi.fin.equifax.com:443)
   Unable to configure RSA server private key (OpenSSL library er-
ror follows)
   [Tue Nov 21 14:43:53 2000] [error] OpenSSL: er-
ror:0B080074:x509 certificate
   routines:X509_check_private_key:key values mismatch
```

위에서 에러 메세지를 얻는다면 키와 인증서가 일치하지 않는 경우인데 디폴트 server.key 파일을 사용하지 않았는지 확신해라. 또한 지시가 정확한 비밀키와 인증서를 가리키고 있는지 확신하기 위해 httpd.conf파일을 체크해야 한다.

비밀키와 인증서가 정확한 포맷이고 서로 일치하는지 확신하기 위해 체크할 수 있다. 이를 위해 각각의 터미널 윈도우에서 비밀키와 인증서를 복호화하기 위해 아래의 명령을 실행시켜라. 각 키의 모듈러스와 지수가 비교할 대상이다. 키와 인증서의 모듈러스와 지수가 일치한다면 인증서와 키가 정확한 쌍인지 확신해라.

모든 다른 것이 실패한다면 새로운 비밀키, CSR 또는 자필 서명 인증서를 생성해라. 이를 하기 전에 CA의 재발급 정책을 체크해라. 재발급시 비용이 들 수 있다.

인증서 내용을 보려면 다음 명령을 실행시킨다:

```
openssl x509 -noout -text -in filename.crt
```

비밀키 내용을 보려면 다음 명령을 실행시킨다:

```
openssl rsa -noout -text -in filename.key
```

5.2. 클라이언트 브라우저에서 인증서 이름 체크 경고가 나타난다(Certificate Name Check Warning is issued by the client's browser).

이는 대부분 CSR을 생성할 때 도메인 네임 시작부분에서 "www"를 생략했기 때문이다. 가상 호스트에 대해 "ServerName" 지시에 의해 정의된 이름은 인증서에 나타난 도메인 네임과 정확히 일치되야 하는데 그렇지 않다면 브라우저가 클라이언트에게 알려줄 것이다. 예외는 와일드 카드 인증서이다. 와일드 카드 인증서의 도메인 네임은 *.somedomain.com 같이 보일 것이다. 이는 somedomain.com 의 어떤 하위 도메인들에 대해 하나의 인증서를 사용할 수 있도록 할 것이다 (예를들면 host1.somedomain.com과 host2.somedomain.com).

5.3. 클라이언트 웹브라우저가 "인증서가 신뢰되지 않는 CA에 의해 서명되었다"라는 경고를 나타낸다(Certificate was Signed by an Untrusted Certificate Authority Warning is issued by the client's browser).

자필 서명 인증서를 사용하고 있다면 이 경고를 얻을 것이다. 클라이언트에 인증서 신뢰 여부를 선택할 수 있게 옵션을 줄 수 있다. CA가 서명한 인증서가 있고 untrusted 경고를 얻는다면 아마도 intermediate (root) 인증서를 설치할 필요가 있다.

5.4. 아파치를 구동할 때 SSLEngine on 이 인식되지 않는 명령어이다(SSLEngine on is an un-recognized command (when starting Apache)).

ModSSL이 아파치와 함께 컴파일되지 않은 경우 이 에러 메세지가 나타난다. 어떤 SSL 패키지는 가상 호스트내에서 SSL을 시동하기 위해 다른 지시를 사용하는데 이러한 패키지를 사용하고 있다면 이 에러 메세지를 받을 것이다.

5.5. PEM passphrase를 잊었는데 이를 재설정하는 방법을 알고 싶다(You have forgotten your "PEM Passphrase" and you would like

to know how to reset it).

이 passphrase를 재설정할 방법은 없으며 passphrase를 기억하고 있거나 새로운 비밀키를 생성하는 것이 유일한 해결책이다. 새로운 인증서를 얻거나 새로이 자신이 서명한 인증서를 생성할 필요가 있다.

6. 용어 해설

인증 (Authenticatoin)

서버, 클라이언트 또는 사용자와 같은 네트워크 참여자(entity)의 명백한 식별. SSL과 관련해서 인증은 서버와 클라이언트 인증서 확인 절차를 나타낸다.

액세스 제어 (Access Control)

네트워크 영역으로의 액세스 제한. 아파치와 관련해서 보통 어떤 URL로의 액세스 제한을 의미한다.

알고리듬 (Algorithm)

한정된 단계내에서 문제를 해결하기 위한 명백한 식 또는 일련의 규칙들. 암호화 알고리듬은 보통 Cipher로 불린다.

인증서 (Certificate)

서버 또는 클라이언트와 같은 네트워크 참여자를 인증하는데 사용되는 데이터 레코드. 인증서는 subject라 불리는 그 소유자 및 issuer라고 불리는 서명 인증서 발급기관(signing Certificate Authority)에 대한 X.509 정보와 소유자의 공개키와 CA 서명을 포함한다. 네트워크 참여자는 CA 인증서를 사용하여 이러한 서명을 확인한다.

인증서 발급 기관 (Certificate Authority)

신뢰받는 제삼자로 보안 방법을 사용하여 인증한 네트워크 참여자들에 대한 인증서에 서명하는 것을 목적으로 한다. 다른 네트워크 참여자들은 CA가 인증서 소지자를 인증했는지를 확인하기 위해 서명을 검사할 수 있다.

인증서 서명 요청 (Certificate Signing Request)

CA에 의뢰를 하기 위한 서명되지 않은 인증서. CA는 자신의 인증서의 비밀키로 이를 서명한다. 일단 CSR이 서명되면 진짜 인증서가 된다. 데이터 암호화를 위한 알고리듬 또는 시스템으로 DES, IDEA, RC4 등이 그 예이다.

암호문 (Ciphertext)

평문 (plaintext)을 암호화한 결과.

설정 지시 (Configuration Directive)

프로그램 동작의 한가지 이상의 측면을 제어하는 설정 명령. 아파치와 관련해서 설정 파일의 첫번째 열에 있는 모든 명령어 이름이다.

암호학 - 대칭 (Cryptography - Symmetric)

클라이언트와 서버가 데이터의 암호화와 복호화에 동일키를 사용한다.

암호학 - 비대칭 (Cryptography - Asymmetric)

공개키와 비밀키 쌍으로 구성되는데 PKI는 비대칭 암호이다.

디지털 서명 (Digital Signatures)

암호화된 메세지와 함께 송신자 식별 및 메세지가 변경되지 않았음을 확인하는 데이터.

HTTPS

하이퍼텍스트 전송 프로토콜 (Secure), 웹상의 표준 암호화된 통신 기구로 실제 단지 SSL을 통한 HTTP이다.

메세지 다이제스트 (Message Digest)

메세지 내용이 교신중에 변경되지 않았음을 보증하는데 사용될 수 있는 메세지의 해쉬

부인 방지 (Non-repudiation)

양측 모두 위조되지 않은 관계에서 언제 누구라도 확인할 수 있는 데이터 무결성 및 출처를 입증하는 서비스 또는 확신을 갖고 거짓이 없다고 주장될 수 있는 인증

개인 또는 참여자가 데이터와 관련해서 특별한 행동을 수행하지 못하도록 하는 암호화 방법을 통해 얻어진 성질(비거부 또는 인가(출처), 의무, 목적 또는 서약의 입증, 또는 소유권의 입증을 위한 기구)

OpenSSL

SSL/TLS에 대한 오픈 소스 툴킷; http://www.openssl.org를 참조

Pass Phrase

비밀키 파일을 보호하는 단어 또는 문구로 인가받지 않은 사용자가 비밀키 파일을 암호화하는 것을 방지한다. 대개 암호에 사용되는 비밀 암호화/복호화 키이다.

Plaintext

암호화되지 않은 평문

비밀키 (Private Key)

수신메세지 복호화 및 송신메세지 서명에 사용되는 공개키 암호법 시스템에서의 비밀키

공개키 (Public Key)

> 이 키 소유자에게 가는 메세지 암호화 및 이 키 소유자에 의해 만들어진 서명을 복호화 하는데 사용되는 공개키 암호법 시스템에서 공개적으로 알려진 키

공개키 암호학 (Public Key Cryptography)

> 암호화와 복호화에 다른 키를 사용하는 비대칭 암호학 시스템의 연구와 응용. 이러한 해당 키들이 키쌍을 구성하며 비대칭 암호학으로 불린다.

Secure Sockets Layer(SSL)

> TCP/IP 네트워크를 통한 일반 통신 인증과 암호화를 위해 넷스케이프사가 만든 프로토콜로 일반적으로 HTTPS(HyperText Transfer Protocol(HTTP) over SSL)로 불린다.

세션 (Session)

> SSL 통신 관련(context) 정보

SSLeay

> Eric A. Young eay aus.rsa.com 이 개발한 최초의 SSL/TLS 구현 라이브러리로 http://www.ssleay.org (http://www.ssleay.org/)를 참조

대칭 암호학 (Symmetric Cryptography)

> 암호화와 복호화 연산 무두에 하나의 비밀키를 사용하는 암호 연구 및 응용

전송 계층 보안(Transport Layer Security)

> TCP/IP 네트워크를 통한 일반적인 통신 인증과 암호화를 위해 IETF(Internet Engineering Task Force)가 만든 SSL의 대체 프로토콜. TLS 버전 1과 SSL 버전 3은 거의 동일하다.

Uniform Resource Locator(URL)

> 웹상의 다양한 자원들의 위치를 나타내는 공식 식별자. 대부분 대중적인 URL 스킴은 http로 SSL은 https 스킴을 사용한다.

X.509

> ITU-T(International Telecommunication Union)가 추천하는 인증 증서 스킴으로 SSL/TLS 인증에 사용된다.

ITU-T

> 권고 X.509 [CCI88c] 는 X.509 인증서 구문론뿐만 아니라 X.500 디렉토리에 대한 인증 서비스를 지정한다. X.509에서 디렉토리 인증은 비밀키 또는 공개키 기법을 사용하여 수행될 수 있는데 후자는 공개키 인증서에 기초한다. 표준의 유익한 부속문서가 RSA 알고리듬을 기술함에도 불구하고 표준은 특정 암호화 알고리듬을 지정하지 않는다.