안전한 세션 키 배포를 위한 TPM 기반의 OpenSSL RSA 키 관리 방법

이기웅 ○ 김병선 김지훈 조진성

경희대학교 컴퓨터공학과

lkw1023@khu.ac.kr, ykbs0903@khu.ac.kr, ppoi106@khu.ac.kr, chojs@khu.ac.kr

A RSA key management method for TPM-based OpenSSL to securely distribute session keys

Kiwoong Lee○ Byoungseon Kim Jihoon Kim Jinsung Cho

Department of Computer Science and Engineering

KyungHee University

요 약

IoT 서비스 구축에 있어 OpenSSL은 개체간 데이터 통신의 기밀성 보장을 위한 대표적 오픈소스 라이브러리이다. 이러한 OpenSSL은 기술적으로 RSA 키 쌍을 통해 암호화된 세션 키를 배포하고, 배포된 세션 키를 통해 데이터 암/복호화를 수행하여 데이터 기밀성을 보장한다. 하지만, 디바이스 내 파일 형태로 관리되는 OpenSSL RSA 암호화 키가 외부에 노출되었을 경우, 암호화된 세션 키 및 데이터의 복호화가 가능하여 보안 위협의 잠재적 원인이 될 수 있다. 이러한 문제를 해결하고자 본 논문에서는 TPM 기반의 OpenSSL RSA 키 관리 방법에 대해 제시한다. 제시하는 방법은 RSA 암호화 키를 TPM의 SRK 추가 암호화를 통해 유출된 RSA 암호화 키의 원본 키 데이터 추출을 차단하여 키 관리 보안성을 강화한다.

1. 서 론

최근 IoT 시장 확대에 따라 다양한 IoT 디바이스들이 출시되고 있으며, 이러한 디바이스의 IoT 서비스 제공을 위한 어플리케이션은 대부분 저비용 및 개발 효율성을 고려하여 오픈 소스 라이브러리를 기반으로 제작된다[1].

이러한 오픈 소스 라이브러리 중, OpenSSL은 IoT 개체간 안전한 데이터 통신을 위해 활용되며 클라이언트-서버 간 데이터의 무결성, 기밀성을 지원하기 위한 라이브러리이다. 본 기술은 OpenSSL 암호화 키를 중심으로 클라이언트-서버 간 세션 수립, 암호화 통신, 인증서 관리 등의 다양한 기능들을 지원하고 있으며[2], 이러한 암호화 키 정보는 외부에 노출되지 않도록 안전한 관리가 요구된다. 만약, OpenSSL 키 정보가 유출되었을 경우, 외부 공격자에 의한 데이터 변조, 도청 등의 문제로 안전한 데이터 통신을 보장할 수 없어 추가적인 인적 사고, 경제적 손실이 발생할 위험이 있다[3].

키 관리 실패의 대표적 사례로써, 2014년 발생한 OpenSSL의 Heartbeat 취약점 공격은 클라이언트-서버 간 연결 유지 기능을 수행하는 Heartbeat를 악용하여 서버에 저장된 민감한 데이터 (암호화 키)를 유출시키는 공격이다. 이 밖에 OpenSSL 키 관리에 대한 취약점들이 계속적으로 보고 및 보완되고 있지만, 나날이 발전하는 해킹 툴들로 인해 소프트웨어 기반 키 관리 방식의 신뢰도에는 여전히 한계가 존재한다[4].

한편, 이러한 소프트웨어적 키 관리 방식의 문제점을 하드웨어적으로 해결하고자 TCG (Trusted Computing Group)에서는 TPM (Trusted Platform Module)이라는 기술 규격을 개발하였다[5]. 본 규격의 주요 기능은 크게 외부에서 데이터를 관리하기 위한 RTS (Root of Trust for Storage), 데이터 변경 및 조작 여부를 증명하기 위한 RTM (Root of Trust for Measurement), 디바이스에 대한 무결성 인증 요청 시 RTM의 정보를 제공하는 RTR (Root of Trust for Report)로 구성된다.

본 논문에서는 기존 OpenSSL RSA 키 관리 방식의 잠재적인 취약점을 보완하기 위해 TPM 기반의 OpenSSL RSA 키 관리 방안을 제시한다. 제시하는 방안은 TPM의 SRK (Storage Root Key)를 통해 OpenSSL의 RSA 키를 암호화함으로써 노출된 RSA 키 파일에서의 키 데이터 추출을 차단하여 키 관리 측면에서의 보안성을 향상시킬 수 있다.

논문의 목차는 다음과 같다. 2장에서는 OpenSSL에

"이 논문은 2015년도 삼성전자 DS의 지원을 받아 수행된 연구결과임."

●TPM key 생성 $ create_tpm_key <*keyfilename*>
●OpenSSL 인증서 생성 $ openssl req -keyform engine -engine tpm -key <*keyfilename*> -new -x509 -days <days>-out <*certfilename*>
●OpenSSL 통신 $ openssl s_server -cert <*certfilename*> -accept <*port*> -keyform engine -engine tpm -key <*keyfilename*>

[표 1] TPM 기반의 OpenSSL 기능을 사용하기 위한 명령어

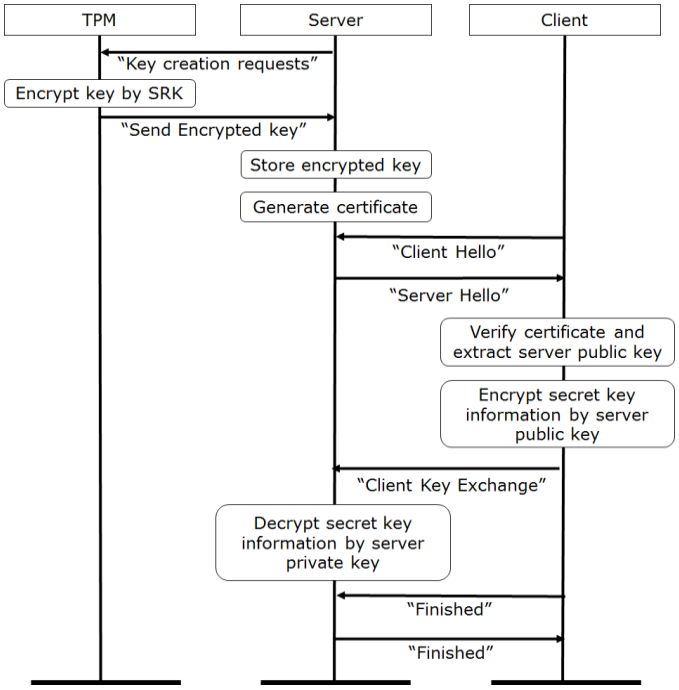
대해 설명하고, 3장은 TPM과 OpenSSL의 연동 방법 및 키 보안 향상 방안을 기술한다. 4장에서는 TPM 유무에 따른 OpenSSL의 성능 비교를 수행하며, 5장은 결론 및 향후 연구를 서술하였다.

2. OpenSSL

SSL은 Netscape에서 개발된 프로토콜로서 1995년 개발되었으며 현재 인터넷 사용자들에게 안전한 개인정보를 교환하기 위한 사실상의 표준 프로토콜로 인정되어 많은 온라인 상거래에 이용되고 있다. SSL은 실제로 다양한 장점을 지닌 암호화 기법들을 사용해 세계 각국에서 사용되는 대부분의 암호화 기법들을 지원할 수 있다. SSL은 버전 3.0 이후 IETF에서 표준화되어 TLS (Transport Layer Security)로 명명되었지만 실제 그 내용은 SSLv3와 TLSv1.0이 같으며, MS Explorer나 Netscape 등 대부분의 브라우저에서 지원하고 있다. 제공되는 보안 서비스로는 암호를 사용해서 메시지의 내용을 숨기는 기밀성 (Confidentiality), 데이터그램의 내용들이 이동 중 고의나 우연한 오류들에 의해서 변화되지 않았다는 것을 확인하는 무결성 (Integrity)이 있다. 또한, OpenSSL이 제공하는 기능 중 하나로 정보를 보낸 사람이 나중에 정보를 보낸 것을 부인하지 못하도록 하는 부인방지 (Nonrepudiation)가 있다.

OpenSSL을 통한 데이터 통신 시 서버 인증 및 세션 키 교환을 위해 Handshake 프로토콜을 사용한다. Handshake 프로토콜의 과정은 서버와 클라이언트가 사용할 암호화 기법을 선택하고 인증서를 통한 서버 인증 및 공개 키 방식의 암호화를 이용하여 세션 키를 안전하게 교환한 후 완료되었다는 신호를 보내는 것으로 종료된다[6].

3. 제안하는 방안

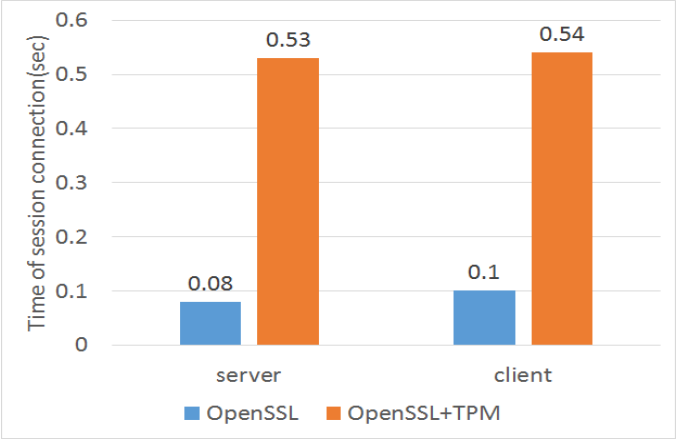


[그림 1] TPM-Server-Client 세션 연결 과정

OpenSSL은 Handshake 프로토콜을 이용하여 세션 키를 교환하며 이 때 RSA 키를 이용한다. 하지만 RSA 키 유출 시 세션 키 또한 유출 될 수 있다. 이에 본 장에서는 TPM 기반의 OpenSSL을 제안하며 TPM을 통해 RSA 키 정보를 보호함으로써 세션 키의 보호 및 안전한 교환을 가능하게 한다.

[표 1]은 TPM 기반의 OpenSSL 기능을 사용하기 위한 명령어를 정리한 것이다. "TPM key 생성"은 SRK로 암호화 된 RSA 키를 생성하기 위해 사용된다. 명령어에서 사용된 인자로 <keyfilename>에 생성할 TPM key의 이름을 입력한다. "OpenSSL 인증서 생성"은 TPM key를 기반으로 X.509 인증서를 생성한다. 인자는 engine에는 tpm, keyform 에는 engine, <keyfilename>에는 TPM key의 이름, <days>에는 인증서의 유효기간, <certfilename>에는 생성할 인증서 이름을 입력한다. "OpenSSL 통신"은 클라이언트-서버 간 데이터 통신을 하기 위해 서버를 실행시키는데 사용된다. 인자는 engine 에는 tpm, keyform 에는 engine, <keyfilename>에는 TPM key의 이름, <certfilename>에는 인증서의 이름, <port>에는 사용하고자 하는 PORT의 값을 입력한다.

[그림 1]은 TPM과 OpenSSL을 연동한 이후 세션 연결 과정을 나타낸다. 우선, 서버가 TPM의 보호를 받는 키 생성을 요청하면 (Key creation requests 단계), TPM은 키를 생성한 후 SRK를 이용해 암호화 하여 서버로 전달한다 (Encrypt key by SRK 단계). 이후 서버는 클라이언트의 세션 연결 요청을 대기한다. 클라이언트가 데이터 통신에서 사용할 암호화 정보와 함께 세션 연결을 요청하면 (Client Hello 단계), 서버는 클라이언트에게 받은 정보에서 선택한 값과 TPM key를 이용하여 생성한



[그림 2] TPM 사용 유무에 따른 세션 수립 시간

인증서를 전달한다 (Server Hello 단계). 인증서를 전달받은 클라이언트는 미리 내장되어 있는 CA의 정보와 인증서의 정보를 확인한 후, 신뢰할 수 있다면 인증서로부터 공개 키를 추출하여 클라이언트와 서버가 사용할 세션 키의 생성에 사용되는 임의의 데이터 값을 암호화하여 서버에 전달한다 (Client Key Exchange 단계). 데이터를 전달받은 서버는 비밀 키를 이용하여 암호화 된 데이터를 복호화한다. 데이터 전송을 완료한 클라이언트는 서버로 세션 연결 완료 신호를 전송하며 (Finished 단계), 서버 또한 클라이언트에게 세션 연결 완료 신호를 전송한다 (Finished 단계).

4. 성능측정

본 장에서는 TPM 유무에 따른 OpenSSL의 세션 요청 처리 및 완료 시간을 측정하였다. 측정방법은 [그림 1]의 클라이언트가 서버에게 Client Hello를 보내는 시점을 시작점, 클라이언트가 서버로부터 Finished를 받는 시점을 끝점으로 하여 현재 시간 출력 함수 clock_gettime()을 통해 차이를 측정하였다[7].

[그림 2]는 TPM 사용 유무에 따른 세션 수립 과정 간 소요되는 시간을 5회 측정하여 평균을 나타낸 결과이다. 기존 OpenSSL의 경우 서버는 0.08초, 클라이언트는 0.1초의 세션 수립 시간을 가지며 TPM기반의 OpenSSL의 경우 서버는 0.53초, 클라이언트는 0.54초의 세션 수립 시간을 갖는다.

측정 결과 TPM을 사용할 경우 서버와 클라이언트 모두 기존 OpenSSL보다 높은 지연시간을 보인다. 하지만 기존의 OpenSSL을 이용한 키 관리에 TPM을 이용한 키 관리가 추가되면서 보안성이 향상된 키 관리가 가능하다.

5. 결론 및 향후 연구

본 논문에서는 세션 키의 안전한 배포를 위해 사용되는 RSA 키의 보안성 향상을 위해 TPM과 OpenSSL의 연동을 통한 RSA 키 관리 방법에 대해 제시하였다. 제시하는 방안은 TPM을 통해 RSA 키를 암호화하여 보호하는 것이다. 이로 인해 키 유출 시에도 키 정보를 보호함으로써 키 악용을 방지할 수 있다. 이에 하드웨어를 추가함으로써 처리시간 증가의 오버헤드는 존재하지만, 하드웨어를 통한 보안성 향상을 기대할 수 있다.

TPM을 통해 RSA키를 암호화하여 보안성을 향상하였지만 소수를 이용하여 생성되는 RSA 키는 제한된 범위 안에서 사용할 소수를 선택함으로 인해 중복생성의 가능성이 있으며 중복된 키를 통해 세션 키 정보 유출이 가능하다[8]. 이에 향후 연구로는 RSA 키 생성방식에 의한 키 중복 생성 가능성에 대한 분석을 할 것이다.

참 고 문 헌

[1] 전종암, 김내수, 고정길, 박태준, 강호용, 표철식, “IoT디바이스 제품 및 기술 동향,” 한국통신학회지(정보와 통신), 31(4), 44-52, 2014.3.

[2] 정인성, 신용태, “OpenSSL을 이용한 보안 통신 API의 설계 및 구현,” 인터넷정보학회논문지, 4(5), 87-96, 2003.10.

[3] 김민정, 유진호, “S/W 취약점으로 인한 손실비용 추정,” 한국전자거래학회지, 19(4), 31-43, 2014.11.

[4] 안우현, 김형수, “코드 주입을 통한 OpenSSL 공유 라이브러리의 보안 취약점 공격,” 정보과학회논문지: 시스템 및 이론, 37(4), 226-238, 2010.8.

[5] 최종욱, 박우람, 박찬익, “TPM에 기반한 iSCSI 네트워크 스토리지의 안전한 접근방안,” 한국정보과학회 학술발표논문집, 36(1D), 5-9, 2009.6.

[6] 정상곤, 정전대, 신재호, 송유진, “SSL Handshake 프로토콜의 성능 개선,” 대한전자공학회 학술대회, 58-61, 1998.6.

[7] 김태환, 표창우, “제어 흐름 검증을 위한 해시 기반 고성능 주소 탐색 알고리즘,” 한국정보과학회 학술논문발표집, 112-114, 2013.6.

[8] 김선종, 권정옥, “금융 보안 서버의 개인키 유출 사고에 안전한 키 교환 프로토콜,” 정보보호학회논문지, 19(3), 119-131, 2009.6.