논문 17-42-05-12 The Journal of Korean Institute of Communications and Information Sciences '17-05 Vol.42 No.05
https://doi.org/10.7840/kics.2017.42.5.1027

CNG 암호 라이브러리에서의 SSL 통신과정 분석

이 경 률◆, 오 인 수*, 이 선 영*, 임 강 빈°

Analysis of SSL Communication Process in CNG Crypto Library

Kyungroul Lee◆, Insu Oh*, Sun-Young Lee*, Kangbin Yim°

요 약

CNG가 활용되는 환경이 증가함에 따라, CNG 암호 라이브러리에서의 보안 취약점 분석에 대한 연구가 요구되는 실정이다. 이에 본 논문에서는 CNG 암호 라이브러리에서의 SSL 통신과정을 분석함으로써 SSL 통신을 활용하는 응용에서 발생 가능한 보안 취약점을 도출하기 위한 자료 및 보안성을 향상시키는데 기여할 것으로 사료된다.

Key Words : Crypto library, CNG library, SSL communication process

ABSTRACT

By a spread of utilizing environment of the CNG library, it is required to analyze its vulnerability. For this reason, in this paper, we analyzed SSL communication process in CNG library. This study is expected to draw vulnerabilities and security threats and improve security criteria for various applications to fully take advantage of the CNG library.

Ⅰ. 서 론

암호를 이용한 데이터 보호 기술은 전자상거래, 개인정보보호 등의 목적을 위하여 필수적인 요소가 되었다. 많은 기관 및 기업으로부터 암호 시스템 구현을 지원하기 위하여 암호 라이브러리를 배포하였으며, 이들은 흔히 암호 API로 불린다. 국외 암호 라이브러리를 살펴보면 IETF의 GSS-API, The Open Group의 GCS-API, 마이크로소프트의 Cryptography API(CAPI), RSA의 PKCS#11, Intel의 CDSA CSSM API 등이 있으며, 국내에는 퓨쳐시스템의 Cryp Tool, 이니텍의 INI Crypto, STI의 J/LOCK, 장미디어의 CEAL98, 트러스컴의 trusCrypt 등이 있다.

상기의 국/내외 암호 라이브러리는 그 종류가 다양하지만, 현재 업계 표준으로 인식되는 라이브러리는 RSA의 PKCS#11과 마이크로소프트의 CAPI이며, 다양한 응용 시스템에 적용되고 있다. 특히, 하드웨어에 종속적인 제품, 즉, 스마트카드, USB 토큰, HSM (Hardware Security Module) 등의 하드웨어 보안 제품들은 암호 라이브러리와의 연동기능을 필수요소로 요구하고 있기 때문에 업계는 이러한 요구사항을 만족시킬 필요가 있다. 그러나 PKCS#11과 CAPI는 확장성에 대하여 고려하지 않고 설계되어 새로운 요구사항이 발생하는 경우 라이브러리를 새로 개발하여야 하는 문제점을 가지고 있다. PKCS#11의 경우 소스가 공개되어 있기 때문에 다양하게 확장할 수 있으나, 지나친 변형에 따른 문제가 제기되고 있으며, CAPI 역시 CSP 구조를 제공하고 있지만, 기존의 인프라에서 제공되는 플러그인 방식이 아니므로 비용과 신뢰성 측면에서 문제점을 가진다.

※ 본 연구는 한국연구재단 이공분야기초연구사업(No. NRF-2015R1D1A1A01057300), 순천향대학교 학술연구비 지원 및 순천향대학교 산학협력단 관리로 수행되었습니다.

◆ First Author : Soonchunhyang University R&BD Center for Security and Safety Industries (SSI), carpedm@sch.ac.kr, 정회원
° Corresponding Author : Soonchunhyang University Department of Information Security Engineering, yim@schac.kr, 정회원
* Soonchunhyang University Department of Information Security Engineering, {catalyst32, sunlee}@sch.ac.kr, 학생회원, 정회원

논문번호 : KICS2017-02-057, Received February 27, 2017; Revised May 3, 2017; Accepted May 4, 2017

CNG는 상기의 CAPI에서의 문제점을 보완하는 민첩성을 지원하기 위하여 플러그인 모델 기반의 공급자(CNG provider) 개념을 도입하였다. 기존 CAPI에서 새로운 알고리즘을 추가하기 위해서는 해당 Cryptographic Service Provider(CSP)를 새로이 개발하고, 이를 검증하기 위하여 마이크로소프트에 요청하여 서명을 받아야 하는 등의 환경변화에 민첩하지 못한 구조를 가지고 있었지만, 새로이 도입한 플러그인 모델 기반의 공급자는 새로운 암호 알고리즘이나 키 관리 방안 등의 새로운 요구가 있을 경우, 개발자가 이를 구현한 후, 기존의 라이브러리에 삽입하여 사용할 수 있도록 하였다. 이는 API를 매번 새로이 개발할 필요가 없기 때문에 개발자 입장에서는 매우 효율적이라 할 수 있다. 이와 같이 CNG는 윈도우즈 비스타부터 등장하였으며, 윈도우즈 비스타, 7, 8, 서버 2008, 서버 2008 R2, 서버 7, 스토리지 서버 2008, 스토리지 서버 2008 R2 등에 기본으로 제공하고 있다[1]. 특히, 윈도우즈 8부터는 CNG가 핵심적인 요소 중 하나로 제공되고 있으며, 윈도우즈 비스타 출시 이후로, 윈도우즈 XP의 지원이 종료되어 7, 8, 10의 점유율이 증가하고 있는 시점이다. 또한, 마이크로소프트 오피스 2010에서 디지털 서명을 사용한 정보보안 및 무결성을 향상시키기 위한 방법으로 CNG를 사용하고 있으며[2], 윈도우즈 8부터 클라우드 컴퓨팅과 관련된 일환으로 데이터 보호를 위하여 CNG의 DAPI(Data protection Application Programming Interface)가 새로이 등장하여 암호 시스템의 핵심기술로 소개되고 있다[3]. 이는 CNG가 활용되는 환경이 증가하고 있다는 것을 의미하므로 이에 대한 연구가 시급한 실정이다[4,15].

암호 라이브러리에서의 취약점을 살펴보면, 그 일례로, 보안 어플리케이션 산업분야에서 가장 기본적으로 사용되는 암호 라이브러리 표준인 PKCS#11을 활용하는 과정에서 구현상의 실수로 인한 문제가 지적된 바 있다. 즉, RSA 인증 클라이언트가 RSA 하이브리드 인증자에 대한 비밀키 객체를 저장하는데 PKCS#11 API를 이용하였는데, PKCS#11의 규격을 따르지 않아 문제가 된 사례가 있다. PKCS#11 사양에 따르면 "SENSITIVE" 및 "NON-EXTRACTABLE"로 태그된 비밀키 객체는 디바이스로부터 익스포트 될 수 없도록 하고 있다. 이는 어플리케이션 개발자가 의도하지 않은 방법으로 키가 사용될 가능성에 대비한 것이나 상기 구현에서의 RSA 인증 클라이언트가 PKCS#11의 비밀키 객체에 대한 중요태그를 무시하고 정상적인 사용자들에게는 이러한 객체들을 공개함으로써 문제가 발생하였다. 이와는 다르게 암호 라이브러리 및 암호 모듈을 활용하는 과정에서의 문제점에 대한 연구도 진행되었다. 그 일례로 OTP 환경에서 리버싱을 활용한 공격으로 OTP 값을 추출하는 연구가 진행된 바 있으며[5,6], OpenSSL 라이브러리에 코드를 삽입하여 키를 탈취하는 공격[7], CAPI 라이브러리의 API 후킹을 통한 암호 알고리즘, 키, 암호문 및 평문을 탈취하는 공격이 연구되었다[8].

고도의 기술발달로 인한 다양한 보안 어플리케이션의 등장과 이들을 지원하기 위하여 다변화하고 있는 플랫폼의 암호 라이브러리에 대한 절대적인 민첩성 요구는 마이크로소프트의 새로운 차세대 암호 라이브러리 CNG의 보급에 지대한 공헌을 하고 있으므로 이러한 상황에서 암호 라이브러리의 민첩성이 가지는 보안 취약점에 대한 분석과 공격 가능성 진단은 필수적이라 할 수 있다. 더구나, CNG가 가지는 커널모드 암호화 기능과 감사 기능은 상기의 취약점 및 공격 가능성에 대한 우려를 증대하고 있으므로 이에 대한 분석도 매우 시급한 상황이다. 따라서 본 논문에서는 이러한 분석에 대한 연구의 일환으로 안전한 통신 프로토콜로 알려진 SSL 프로토콜의 통신과정을 CNG 라이브러리를 대상으로 분석하였으며, SSL 통신 시, CNG 혹은 윈도우즈가 제공하는 라이브러리에서 호출되는 함수의 흐름을 분석함으로써 SSL 통신에서 사용되는 정보와 CNG에서 활용되는 실제 정보와의 연관성을 도출하였다.

Ⅱ. SSL 통신과정

SSL은 1994년 Netscape사에서 웹 브라우저를 통하여 안전하게 데이터를 전송하기 위한 목적으로 제안되었으며, 1996년 IETF(Internet Engineering Task Force)에서 SSL v3.0을 제안하였다. 이후 지속적으로 수정 및 보완되었으며, 1999년에는 TLS(Transport Layer Security)로 명칭이 바뀌어 RFC 2240(TLS v1.0)으로 표준화되었다. SSL/TLS는 전송계층과 응용계층 사이에 위치하여 응용계층 프로그램의 보안설정을 지원하며, 이를 위하여 내부적으로 레코드 레이어와 handshake, change cipher spec, alert, application data 프로토콜로 이루어진다.

SSL/TLS는 세션상태와 커넥션상태로 이루어지며, 하나의 세션 내에 여러 개의 커넥션이 포함되는 형태이다. 또한, 클라이언트와 서버가 통신을 수행하면서 설정하는 알고리즘과 키를 저장할 때는 예비상태, 레코드 레이어에서 데이터를 처리할 때는 현재상태로

변경하여 통신하며, 데이터 송/수신을 위하여 읽기상태와 쓰기상태를 준비하고 있다.

Handshake 프로토콜에서는 알고리즘과 같은 보안 파라미터를 설정하며, 설정된 파라미터는 레코드 레이어로 전달되어 키 블록을 생성한 후, 예비상태로 전환한다. 이후 change cipher spec 프로토콜에 의하여 현재상태로 전환되며, 기존의 예비상태는 초기화되고, 이 과정에서 change cipher spec 메시지가 전송되면 읽기상태와 쓰기상태를 통하여 데이터를 송/수신한다. Full handshake에 의하여 세션이 생성되면, 이후 통신은 abbreviated handshake 프로토콜에 의하여 세션상태는 공유하면서 커넥션상태만 재생성하여 통신이 이루어진다.

Handshake 프로토콜에서는 클라이언트가 서버에게 보안 파라미터를 요청하면, 서버는 필요한 파라미터를 설정하여 change cipher spec 프로토콜을 활성화하고, application data 프로토콜을 통하여 데이터를 안전하게 전송한다. 그리고 통신 과정에서 발생한 오류들은 alert 프로토콜을 통하여 처리된다. 이 프로토콜에서는 클라이언트와 서버가 상호인증을 수행하고, 암호 알고리즘과 암호키, MAC 알고리즘 등의 파라미터를 설정한다. Handshake 프로토콜은 full handshake 프로토콜과 abbreviated handshake 프로토콜로 구성되며, full handshake에 대한 동작과정을 그림 1에 나타내었다[9].

클라이언트는 client hello 메시지를 서버로 전송한 후, 서버의 응답을 기다리며, 서버가 server hello 메시지를 전송함으로써 프로토콜이 시작된다. 클라이언트와 서버는 hello 메시지에 프로토콜 버전과 세션 아이디, cipher suite, 압축방법 등을 포함하여 생성한 난수를 교환한다. 이후, certificate와 key exchange 메시지를 통하여 키 생성에 필요한 pre-master secret과 master secret을 서로 공유하며, 인증을 위하여 인증서를 교환할 수 있다. 이와 같은 과정이 완료되면, 설정된 파라미터들을 change cipher spec 메시지를 통하여 공유한 후, finished 메시지를 전송함으로써 세션이 연결된다.

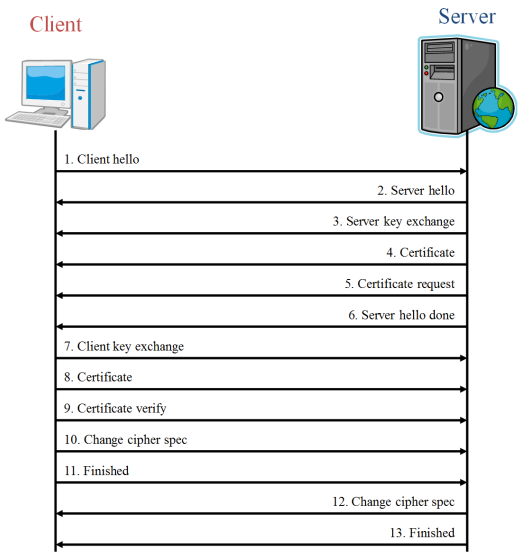

그림 1. SSL의 full handshake 프로토콜
Fig. 1. Full handshake protocol of SSL

클라이언트가 새로운 세션을 요구할 때는 empty 세션 아이디를 전송하며, 연결된 세션을 통하여 새로운 커넥션을 생성할 경우에는 client hello 메시지에 재사용을 위한 세션 아이디를 서버로 전송한다. 서버는 일치하는 세션 아이디를 확인하여 abbreviated handshake 프로토콜을 통하여 change cipher spec을 교환하며, 이에 대한 과정을 그림 2에 나타내었다[9].

Change cipher spec 메시지는 handshake 프로토콜 과정에서 설정된 예비상태를 현재상태로 전환하는 메시지이며, alert 프로토콜은 모든 통신과정에서 발생하는 에러 메시지를 전달한다. 에러 메시지는 warning level과 fatal level로 구분되며, fatal level일 경우에는 세션이 종료된다.

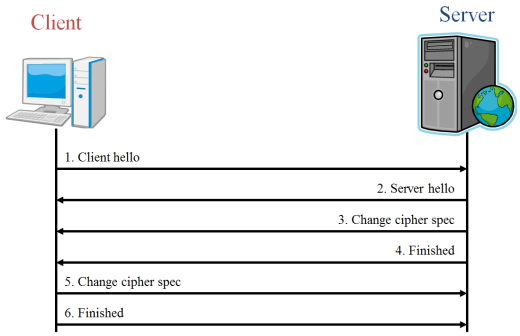

그림 2. SSL의 abbreviated handshake 프로토콜
Fig. 2. Abbreviated handshake protocol of SSL

Ⅲ. CNG에서의 SSL 통신과정 분석

본 논문에서는 상기와 같은 과정을 통하여 안전한 통신을 지원하는 SSL 프로토콜을 대상으로 암호 라이브러리인 CNG 라이브러리에서 활용할 때의 통신 및 동작과정을 분석하며, 이를 통하여 SSL 통신에서 사용되는 정보와의 연관성을 도출한다. 실험환경은

Intel(R) Core(TM) i5-2410M CPU 2.30 GHz, 4GB RAM, Microsoft Windows 7 Home Premium K 운영체제가 설치된 PC를 이용하였으며, CNG를 사용하기 위하여 CNG SDK, Windows SDK 7.1과 .NET Framework 4 설치, 개념검증도구를 구현하기 위하여 Visual Studio 2005 설치, CNG의 샘플 소스 참조를 위하여 CNG Demo Sample Code 설치, 통신과정 분석을 위하여 역공학 도구인 OllyDbg v1.10을 사용하였다.

SSL 통신과정 분석을 위하여 CNG 라이브러리, 즉, BCrypt와 NCrypt 라이브러리에서 사용하는 모든 함수를 후킹한 후[10,11], Google에서 사용자 아이디와 비밀번호를 암호화하는 함수인 SslEncryptPacket 함수의 인자를 확인하였으며, 그 결과를 그림 3에 나타내었다.

그림을 살펴보면 사용자 아이디, 비밀번호는 SslEncryptPacket 함수가 호출될 때의 입력 인자인 pbInput에 저장된 것을 확인할 수 있다. SslEncryptPacket 함수는 그림 4와 같이 총 8개의 입력 인자와 3개의 출력인자를 가진다[12].

인자를 상세히 살펴보면 hSslProvider는 SSL 프로토콜 공급자의 핸들, hKey는 패킷을 암호화하기 위하여 사용되는 키의 핸들, pbInput은 암호화하려는 패킷을 포함한 버퍼의 포인터, cbInput은 pbInput 버퍼의 바이트 크기, pbOutput은 암호화된 패킷을 저장할 버퍼의 포인터, cbOutput은 pbOutput 버퍼의 바이트 크기, pcbResult는 pbOutput 버퍼에 써진 바이트 크기, SequenceNumber는 암호화하는 패킷에 해당하는 순차 번호, dwContentType은 패킷 내용의 유형이다. 이러한 입력 인자로 유추해 볼 때, hSslProvider 인자에는 SSL 통신을 위한 공급자와 관련된 정보, hKey 인자에는 키와 관련된 정보, pbInput 인자에는 암호화하기 위한 정보가 저장되며, 이 정보를 기반으로 pbOutput 인자에 암호화된 정보가 저장될 것이라 판단된다. 본 논문에서는 이와 같은 정보들을 하나씩 확인함으로써 SSL 통신과정에 대한 분석을 진행하였으며, 크게 암호문 생성과정과 키 생성과정으로 분류하여 분석하였다.

그림 3. Google에서 SSL 통신 시 사용자 아이디와 비밀번호를 암호화하는 함수 인자 확인
Fig. 3. Parameters of encryption for ID and password of user when SSL communication in Google

```
SECURITY_STATUS WINAPI SslEncryptPacket(
  _In_     NCRYPT_PROV_HANDLE hSslProvider,
  _Inout_  NCRYPT_KEY_HANDLE hKey,
  _In_     PBYTE *pbInput,
  _In_     DWORD cbInput,
  _Out_    PBYTE pbOutput,
  _In_     DWORD cbOutput,
  _Out_    DWORD *pcbResult,
  _In_     ULONGLONG SequenceNumber,
  _In_     DWORD dwContentType,
  _In_     DWORD dwFlags
);
```

그림 4. SslEncryptPacket 함수 원형[12]
Fig. 4. SslEncryptPacket function prototype[12]

3.1 암호문 생성과정 분석

먼저 암호문 생성과정을 분석하였다. SSL 통신에 활용되는 공급자와 관련된 정보를 추출하기 위하여 후킹된 정보에서 공급자 핸들 주소(0x05E8BA40)를 검색하면, 해당 정보가 존재하지 않음을 확인하였다. 본 논문에서는 공급자 핸들을 여는 SslOpenProvider 함수가 후킹되기 이전에 호출되어 후킹 결과에 나타나지 않음을 역공학을 이용하여 확인하였다. 그림 5는 함수 호출의 결과이다.

따라서 단순히 후킹만 수행하는 것이 아닌, 모든 라이브러리가 로드될 때 이벤트가 발생하도록 분석과정을 수정한 후, NCrypt 라이브러리가 로드되는 시점을 찾고, NCrypt 라이브러리가 로드되면 SslOpenProvider 함수에 중지점을 설정하여 함수가 호출되지 않도록 한 다음 후킹을 수행하였다. 그 결과, 그림 6과 같이 SslOpenProvider 함수가 호출되는 것을 확인하였고, SslOpenProvider를 호출하여 얻어진 SSL 공급자 핸들의 주소가 SslEncryptPacket 함수를 호출할 때의 SSL 공급자 핸들의 주소와 동일한 것을 확인하였다. SslOpenProvider 함수를 호출할 때는 특별한 정보 없이 오픈하려고 하는 공급자에 대한 이름이 인자로 입력되며, 함수 원형을 그림 7에 나타내었다.

현재까지 SSL 공급자에 대한 정보, 암호화하려는 패킷의 평문에 대한 정보를 획득하였으므로, 암호문을 생성하기 위한 키 정보를 분석한다. SSL 통신에서

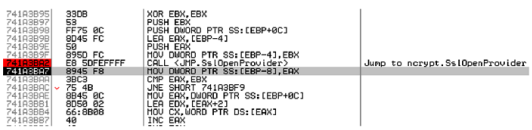

그림 5. SslOpenProvider 함수 호출 확인
Fig. 5. Calling routine of SslOpenProvider function

그림 6. SSL 공급자 핸들 주소 확인
Fig. 6. Verification of the SSL provider handle address

```
SECURITY_STATUS WINAPI SslOpenProvider(
  _Out_   NCRYPT_PROV_HANDLE *phSslProvider,
  _In_    LPCWSTR pszProviderName,
  _In_    DWORD dwFlags
);
```

그림 7. SslOpenProvider 함수 원형[13]
Fig. 7. SslOpenProvider function prototype[13]

SslImportKey 함수를 이용하여 키와 관련된 정보를 불러오고, SslExportKey 함수를 이용하여 키와 관련된 정보를 내보낸다. 따라서 SslEncryptPacket 함수를 호출할 때, hSslProvider 인자인 SSL 공급자 핸들의 주소와 hKey 인자인 키 핸들 주소가 일치하는 후킹된 SslImportKey 함수를 확인함으로써 그림 8과 같이 주소가 동일한 키의 정보를 도출하였다.

그림 8. SSL 통신 시 사용하는 키 핸들 주소 확인
Fig. 8. Verification of the kay handle address when SSL communication

키와 관련된 정보는 그림 9와 같이 SslImportKey 함수의 pbKeyBlob 인자를 통하여 전달되는 것을 확인하였으며, 현재까지 확인된 모든 정보, 암호화하려는 패킷의 평문에 대한 정보, 암호화에 필요한 키와 관련된 정보를 토대로 암호문의 생성이 가능하다. 따라서 상기 분석 결과가 실제 SSL 동작과정에 활용되는 정보인지 검증하기 위하여 SSL 통신과정에서 전송되는 암호문과 추출한 키를 이용하여 추출한 평문을 암호화한 결과를 비교하였다. 하지만 그림 10과 같이 암호문의 비교 결과가 다른 것을 확인하였다.

두 암호문이 다른 원인을 분석한 결과, SslEncryptPacket 함수의 SequenceNumber 인자가 3인 것을 확인하였다. SequenceNumber 인자는 암호화하는 패킷에 해당하는 순차 번호이므로 SequenceNumber 인자가 1인 암호문을 비교한 결과, 그림 11과 같이 두 암호문이 동일함을 확인하였다.

SequenceNumber 인자가 1인 암호문은 일치하였으나, SequenceNumber 인자가 2인 암호문에 대해서는 암호문이 일치하지 않았다. 이 결과로, 암호 설정을 확인한 결과, cipher suite 정보가 그림 12와 같이 TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA_P256을 사용하는 것을 확인하였다. 암호 설정으로 CBC 모드를 사용하기 때문에 SequenceNumber 인자

```
SECURITY_STATUS WINAPI SslImportKey(
  _In_    NCRYPT_PROV_HANDLE hSslProvider,
  _Out_   NCRYPT_KEY_HANDLE *phKey,
  _In_    LPCWSTR pszBlobType,
  _In_    PBYTE pbKeyBlob,
  _In_    DWORD cbKeyBlob,
  _In_    DWORD dwFlags
);
```

그림 9. SslImportKey 함수 원형[14]
Fig. 9. SslImportKey function prototype[14]

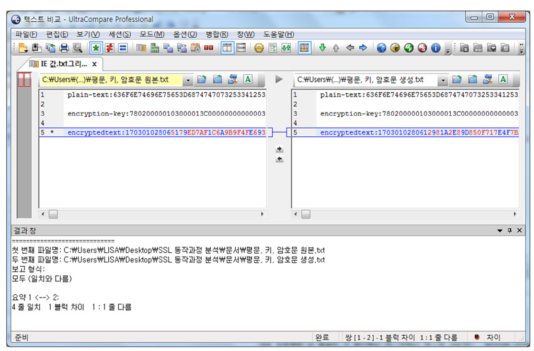

그림 10. 추출한 키를 이용하여 추출한 평문을 암호화한 결과와 암호문 비교
Fig. 10. Compared result of cipher text with encrypted result of extracted plain text based on extracted key

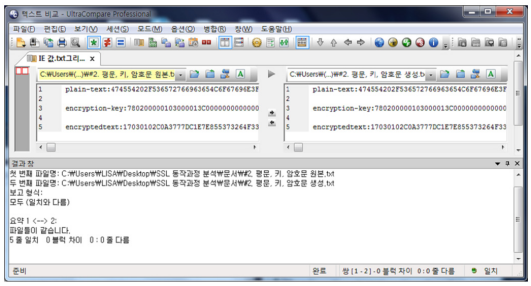

그림 11. SSL 통신 시 SequenceNumber 인자가 1인 암호문 비교
Fig. 11. Compared result of cipher texts as sequence number 1 in SSL communication

```
NewSslLookupCipherSuiteInfo [OUT]: hSslProvider(0664A400)
dwProtocol(00000301)
dwCipherSuite(0000C013)
dwKeyType(00000017)
pCipherSuite->dwBaseCipherSuite(0000C013)
pCipherSuite->dwCipherBlockLen(00000010)
pCipherSuite->dwCipherLen(00000080)
pCipherSuite->dwCipherSuite(0000C013)
pCipherSuite->dwHashLen(000000A0)
pCipherSuite->dwKeyType(00000017)
pCipherSuite->dwMaxExchangeLen(00000100)
pCipherSuite->dwMinExchangeLen(00000100)
pCipherSuite->dwProtocol(00000301)
pCipherSuite->szCertificate(HEX:525341000000000000000000000000000000000000000000000000
CHAR:RSA                                                                            )
pCipherSuite->szCipher(HEX:414553000000000000000000000000000000000000000000000000000000
CHAR:AES                                                                            )
pCipherSuite->szCipherSuite(HEX:544C535F4543444488455F5253415F574954485F4145535
CHAR:TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA_P256                                        )
pCipherSuite->szExchange(HEX:454434485F5032353600000000000000000000000000000000000000
CHAR:ECDH_P256                                                                      )
pCipherSuite->szHash(HEX:534841310000000000000000000000000000000000000000000000000000
CHAR:SHA1                                                                           )
dwFlags(00000000)
```

그림 12. SSL 통신과정에서 설정된 cipher suite
Fig. 12. Configured cipher suite during SSL communication

가 1인 평문부터 3인 평문까지 암호화한 후, 암호문을 비교하였으며, 그 결과, 그림 13과 같이 암호문이 동일한 것을 확인하였다. 따라서 상기 분석과정을 통하여 평문과 키에 관련된 정보를 확인함으로써 암호문 생성과정에 대한 분석을 완료하였다.

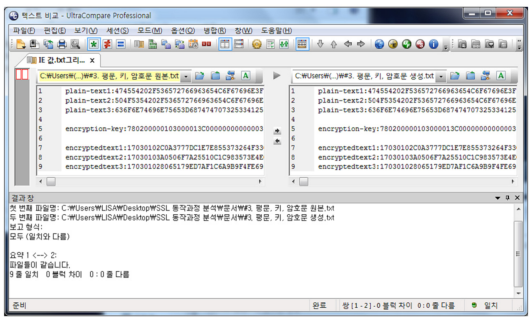

그림 13. 추출한 키를 이용하여 추출한 평문을 CBC 모드로 암호화한 결과와 암호문 비교
Fig. 13. Compared result of cipher text with encrypted result of extracted plain text based on extracted key in CBC mode

3.2 키 생성과정 분석

암호문 생성과정을 분석하였으므로, 그 다음 단계인 키 생성과정을 분석하였다. 암호문 생성과정의 분석에서 추출한 키와 관련된 정보는 그림 14에 나타낸 것과 같이 키 자체에 대한 정보가 아닌 CNG에서 활용하기 위한 구조를 기반으로 키가 포함된 것이라 판단된다.

따라서 본 논문에서는 SslImportKey 함수 호출 이전에 실제 키를 생성하고 생성한 키를 기반으로 CNG에서 SSL 통신과정에 활용 가능한 구조로 변환하는 것이라 가정하였다. 이에 후킹된 결과에서 CNG가 제공하는 키와 관련된 함수를 호출하는지 확인하였지만, 키와 관련된 함수는 호출되지 않았으며, 이는 CNG 라이브러리 이외의 윈도우즈 라이브러리에서 SSL 통신과 관련된 함수를 호출할 것이라 가정하였고, 실제로 SspiCli 라이브러리의 InitializeSecurityContext 함수가 호출되는 것을 확인하였다. InitializeSecurityContext 함수는 클라이언트와 서버 간 security context를 생성하는 기능을 수행하므로 이 함수를 후킹한 후, 입/출력 인자와 CNG 함수와의 호출과정을 분석하였다. 분석 결과, InitializeSecurityContext 함수 내부에서 SslImportKey 함수를 이용하여 암/복호키를 불러오는 것을 확인하였으며, 그 결과를 그림 15에 나타내었다.

InitializeSecurityContext 함수 내부 호출 과정에서 어떠한 버퍼가 입력 인자로 전달되는지를 확인하였으며, 해당 값을 확인한 결과, 네트워크로 전송되는 Change Cipher Spec과 Encrypted Handshake

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000   78 02 00 00 01 03 00 00 13 C0 00 00 00 00 00 00   x........À......
00000010   30 02 00 00 14 00 00 00 95 6A 57 99 62 1F 84 80   0.......•jW™b.„€
00000020   0C F6 FC 30 45 6E 07 63 70 A4 D6 24 00 00 00 00   .öü0En.cp¤Ö$....
00000030   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000040   00 00 00 00 00 00 00 00 30 02 00 00 4B 53 53 4D   ........0...KSSM
00000050   02 00 01 00 01 00 00 00 10 00 00 00 80 00 00 00   ............€...
00000060   10 00 00 00 9C 92 8C 56 CC AD 4C 23 18 0C 7B 49   ....œ'ŒVÌL#..{I
00000070   C7 3A 09 E2 00 00 00 00 EE 68 F6 93 18 F1 B4 E8   Ç:.â....îhö".ñ´è
00000080   FC 4B E8 B3 0B 21 39 4E 9C 92 8C 56 CC AD 4C 23   üKè³.!9Nœ'ŒVÌL#
00000090   18 0C 7B 49 C7 3A 09 E2 1D 93 14 90 D1 3E 58 B3   ..{IÇ:.â.“..Ñ>X³
000000A0   C9 32 23 FA 0E 08 2A 18 2F 76 B9 3B FE 48 E1 88   É2#ú..*./v¹;þHá^
000000B0   37 7A C2 72 39 72 E8 6A 6B ED BB 29 95 A5 5A A1   7zÂr9rèjkí»)•¥Z¡
000000C0   A2 DF 98 D3 9B AD 70 B9 F6 BC ED 3D 63 19 B7 9C   ¢ß˜Ó›p¹ö¼í=c.·œ
000000D0   C1 C6 2F 4F 5A 6B 5F F6 99 73 AF 83 FA 6A 18 1F   ÁÆ/OZk_ö™s¯ƒújˆ..
000000E0   3B AC 37 50 61 C7 68 A6 7F 36 8B 6C 85 5C 93 73   ;¬7PaÇh¦.6‹l…\"s
000000F0   BE F0 A4 23 DF 37 CC 85 A5 7D 1C F2 20 21 8F 81   ¾ð¤#ß7Ì…¥}.ò !..
00000100   9E D1 2B A2 41 E6 E7 27 AB E9 D0 71 8B C8 5F F0   žÑ+¢Aæç'«éÐq‹È_ð
00000110   15 19 74 52 54 FF 93 75 A6 35 4D 51 2D FD 12 A1   ..tRTÿ"u¦5MQ-ý.¡
00000120   38 E4 66 F3 6C 1B F5 86 3F D3 09 01 12 2E 1B A0   8äfól.õ†?Ó.....
00000130   2A CA 7D 53 46 D1 88 D5 81 D5 6F B4 3C 05 C5 9F   *Ê}SFÑˆÕ.Õo´<.Å Ÿ
00000140   CF 90 03 15 F4 BA 65 2F 62 77 C4 32 BD D0 AA 2B   Ï...ô°e/bwÄ2½Ð ª+
00000150   F3 95 C6 8A 3B 2A 66 3A 41 28 94 CB DF A7 6E 19   ó•ÆŠ;*f:A(”ËßSn.
00000160   4E 45 6C A1 C8 BF A0 B0 B9 8D 0A 90 9E 8F FA D2   NEl¡È¿ °¹...ž.úÒ
00000170   91 E2 02 B8 86 FA CC 11 22 91 83 F6 27 02 F0 42   'â.¸†úÌ."'ƒö'.ðB
00000180   0F 6D F8 6A 17 18 CE A9 28 0A 1C A4 05 93 73 B4   .møj..Î©(..¤.“s´
00000190   28 6F 08 28 18 75 36 C3 A5 C8 5D 24 2D 99 6F 10   (o.(.u6Å¥È]$-™o.
000001A0   2D FC 7B 9C 30 1A 3E EB AA 85 91 65 88 51 32 34   -ü{œ0.>ëª…‘e^Q24
000001B0   00 65 14 8C 1D E6 45 77 64 D0 79 C7 22 D4 A3 51   .e.Œ.æEwdÐyÇ"Ô£Q
000001C0   88 34 26 B8 1D 83 51 FB 9C 92 8C 56 CC AD 4C 23   ^4&¸.ƒQûœ'ŒVÌL#
000001D0   18 0C 7B 49 C7 3A 09 E2 00 00 00 00 00 00 00 00   ..{IÇ:.â........
000001E0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000001F0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000200   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000210   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000220   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
```

그림 14. SSL 통신과정에서 추출한 키와 관련된 정보
Fig. 14. Extracted key-related information in SSL communication

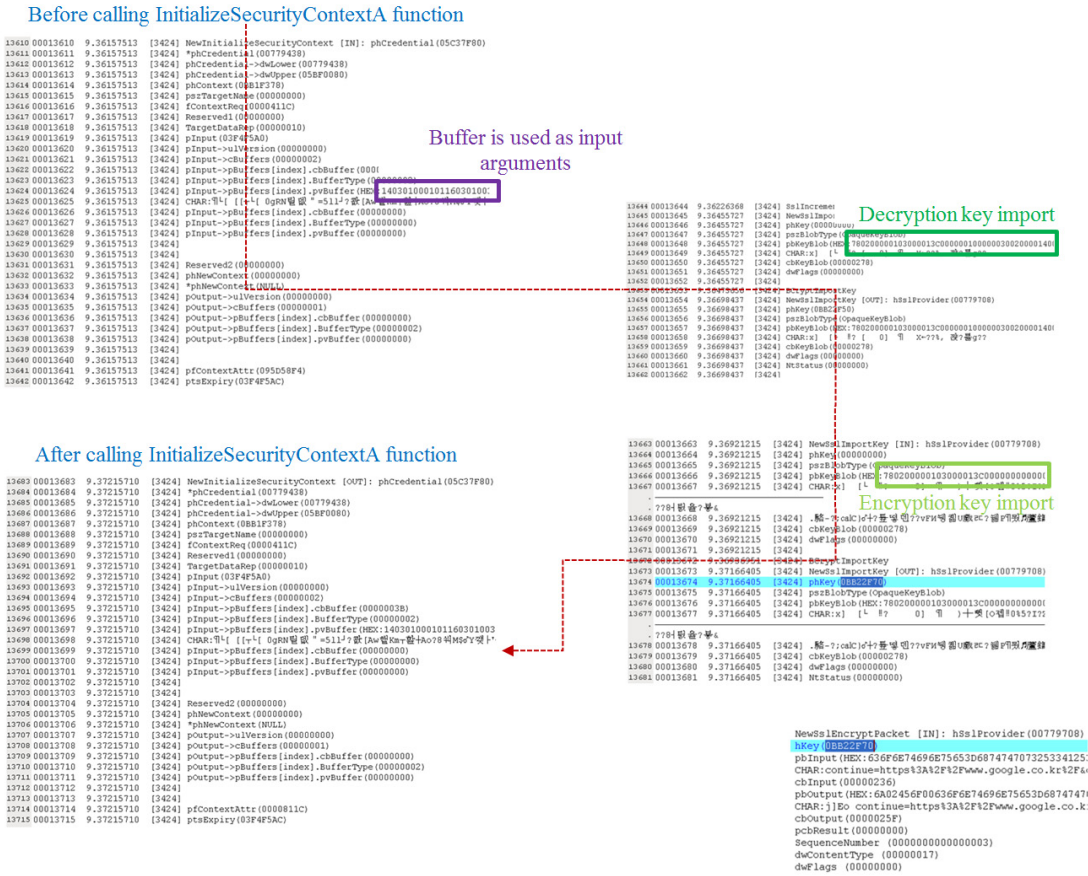

그림 15. InitializeSecurityContext 함수 내부 호출 과정
Fig. 15. Internal calling process of InitializeSecurityContext function

Message 정보가 일치하는 것을 확인하였다. 이 정보는 SSL 통신과정에서 활용되는 메시지 중 하나로서 연결의 마지막 메시지에 해당한다. 따라서 SSL 통신과정에 활용되는 메시지를 분석하기 위하여 InitializeSecurityContext 함수가 호출될 때의 입/출력 인자를 확인한 결과, SSL 통신과정의 연결에 활용되는 메시지인 ClientHello, ServerHello, Certificate, Server Key Exchange, Server Hello Done, Client Key Exchange, Change Cipher Spec 메시지들이 입/출력인자로 활용되는 것을 확인하였다. 각 메시지에 대한 확인 결과를 그림 16, 그림 17, 그림 18, 그림 19에 나타내었고, 전체 함수 호출 과정을 그림 20에 나타내었다.

CNG 이전의 CAPI에서도 InitializeSecurity Context 함수 내에서 암/복호키를 불러오며, SslImportKey 함수가 아닌 CryptImportKey 함수를 호출하고, 암/복호를 위하여 CryptEncrypt/CryptDecrypt 함수를 호출하며, 키를 불러오기 전까지의 과정은 CNG와 CAPI가 동일한 것을 확인하였다.

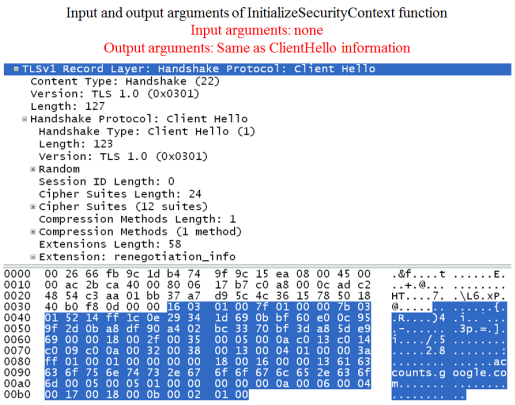

그림 16. InitializeSecurityContext 함수 입력 인자 확인 (ClientHello)
Fig. 16. Verification of input arguments of InitializeSecurityContext function(ClientHello)

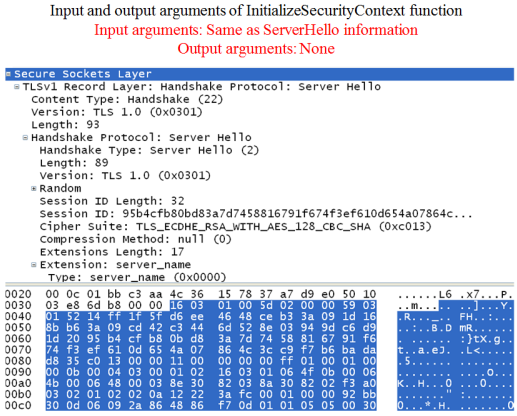

그림 17. InitializeSecurityContext 함수 입력 인자 확인 (ServerHello)
Fig. 17. Verification of input arguments of InitializeSecurityContext function(ServerHello)

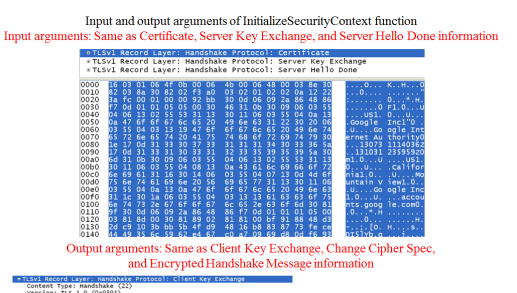

그림 18. InitializeSecurityContext 함수 입력 인자 확인 (Certificate, Server Key Exchange, Server Hello Done, Client Key Exchange, Change Cipher Spec)
Fig. 18. Verification of input arguments of InitializeSecurityContext function(Certificate, Server Key Exchange, Server Hello Done, Client Key Exchange, Change Cipher Spec)

3.3 보안 취약점 및 공격 가능성 고찰

본 절에서는 본 논문에서 분석한 결과를 기반으로 CNG 암호 라이브러리에서의 SSL 통신 과정에서 발생 가능한 보안 취약점 및 공격 가능성에 대한 고찰 결과를 서술하며, 암호와 관련된 정보 탈취 가능성, 안전성이 낮은 보안 파라미터 강제 설정 가능성을 서술한다.

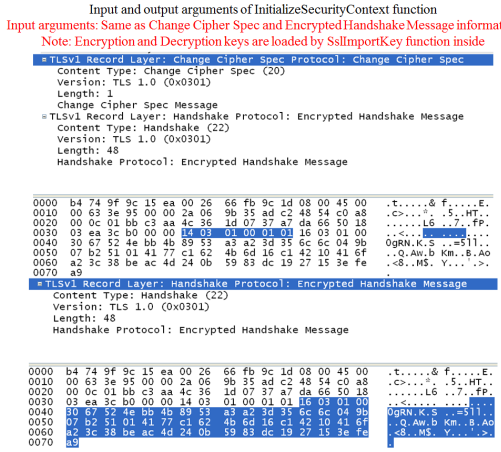

그림 19. InitializeSecurityContext 함수 입력 인자 확인 (Change Cipher Spec, Encrypted Handshake Message)
Fig. 19. Verification of input arguments of InitializeSecurityContext function(Change Cipher Spec, Encrypted Handshake Message)

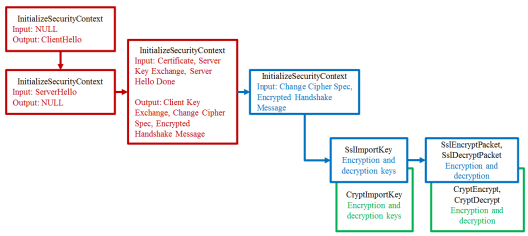

그림 20. SSL 통신의 전체 함수 호출 과정
Fig. 20. Calling process of whole functions in SSL communication

- 암호와 관련된 정보 탈취 가능성

본 논문에서 분석한 암호문 생성과정을 기반으로 공격 가능성을 살펴보면, SSL 통신에서 활용되는 정보는 SslEncryptPacket 함수 및 SslDecryptPacket 함수의 인자로 전달되며, 해당 함수를 후킹한 후, 이러한 인자를 추적한다면, 평문 및 암호문의 탈취가 가능하다. 다시 말하면, 공격자는 전달되는 정보를 모두 수집한 후, 암호문에 대응하는 평문을 확보할 수 있으며, 평문에 포함된 다양한 중요정보를 탈취함으로써 악의적인 공격에 활용한다. 예를 들어, SSL 통신으로 사용자를 인증하는 경우, 아이디와 비밀번호를 암호화하여 전달하지만, 공격자는 암호화되기 전의 평문을 확보한 후, 평문에 포함된 아이디와 비밀번호를 탈취하여 인증을 우회하는 것이 가능하며, 2차 및 3차 공격을 위한 수단으로도 활용이 가능하다. 또한, 공격자 및 공격환경에 따라 지속적인 정보유출이 어려운 경우에는 SslImportKey 함수에서 전달되는 암/복호키를 탈취한 후, 네트워크로 전달되는 암호문을 복호함으로

써 평문을 획득하는 것이 가능하며, 상기와 동일한 공격 가능성이 존재한다.

● 안전성이 낮은 보안 파라미터 강제 설정 가능성

본 논문에서 분석한 키 생성과정을 기반으로 공격 가능성을 살펴보면, CNG 라이브러리 내부의 함수가 아닌 SspiCli 라이브러리의 InitializeSecurityContext 함수를 통하여 ClientHello, ServerHello, Certificate, Server Key Exchange, Server Hello Done, Client Key Exchange, Change Cipher Spec 메시지들을 생성하므로, 전달되는 인자를 조작함으로써 악의적인 공격을 수행하는 것이 가능하다. 예를 들어, 전달되는 메시지를 강제로 삽입하거나 보안 파라미터를 강제로 설정하여 취약점이 존재하도록 환경을 구성한 후, 이를 기반으로 공격을 시도함으로써 공격자의 목적을 달성할 수 있다. 실제로 과거의 공격을 살펴보면, cipher suite 메시지의 목록을 수정하여 취약한 알고리즘을 선택하도록 유도한 후, 키를 유출하는 cipher suite rollback 공격이 있으며, 보안성이 좋은 높은 버전을 지원하더라도 상대적으로 취약한 낮은 버전으로 통신이 이루어지도록 유도하는 version rollback 공격, 키 교환 알고리즘을 서버와 클라이언트가 다르게 선택하도록 유도하여 통신을 방해하는 key exchange algorithm rollback 공격 등의 가능성이 존재한다.

상기와 같은 공격에 대응하기 위해서는 CNG 라이브러리의 위/변조, SSL 통신에 직접적으로 관여하는 함수의 위/변조 및 후킹 등을 지속적으로 관찰하여 공격을 탐지하고 방지하여야 한다. 이를 위해서는 본 논문의 결과와 같이 통신과정에서 함수의 호출과정 및 전달되는 인자 등에 대한 상세한 분석이 이루어져야 하며, 이러한 분석결과가 존재하지 않는다면, 다양한 지점에서 시도되는 공격을 탐지하고 방지하기 어렵다. 따라서 본 논문의 결과를 토대로 실제 호출되는 과정에서 감시하여야 할 부분을 분석하고, 그 결과를 기반으로 대응방안을 도출한다면 더욱 안전한 시스템을 구성할 것으로 사료된다.

Ⅳ. 결 론

본 논문은 CNG 암호 라이브러리에서의 SSL 통신과정에 대하여 분석하였다. 이를 위하여 SSL 통신과정을 조사하였으며, Google에서 사용자가 로그인할 때 활용되는 SSL 통신과정을 상세히 분석하였다. SSL 통신과정은 크게 암/복호문 생성과정과 키 생성과정으로 분류되며, 암/복호문 생성과정은 SSL 통신을 위한 공급자를 로드한 후, 로드한 공급자를 기반으로 암/복호 함수인 SslEncryptPacket/SslDecryptPacket 함수를 호출함으로써 데이터를 안전하게 전달한다. 암/복호에 활용되는 키는 SslImportKey 함수를 호출함으로써 임포트하여 활용하며, CBC 모드로 암/복호하는 것을 검증하였다. 키 생성과정은 SspiCli 라이브러리의 InitializeSecurityContext 함수를 통하여 SSL 통신에서 활용되는 메시지인 ClientHello, ServerHello, Certificate, Server Key Exchange, Server Hello Done, Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message를 생성하며, 이러한 과정에서 키를 생성하기 위한 정보를 교환한 후, 키를 임포트하여 사용하는 것을 확인하였다. 본 논문의 결과는 CNG를 활용하여 SSL 통신을 제공하는 다양한 응용의 보안성을 향상시키는데 기여할 것으로 사료된다.

References

[1] Microsoft, *Cryptography next generation*, Retrieved Jan., 23, 2017, from http://technet.microsoftcom/en-us/library/cc730763(v=ws.10).aspx

[2] Microsoft, *Microsoft Office 2010 and Microsoft SharePoint 2010 integration*, Retrieved Jan., 23, 2017, from https://www.google.co.kr/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=0ahUKEwjp_cbWpNjRAhXHgrwKHZdMBSkQFggeMAA&url=http%3A%2F%2Fdownload.microsoft.com%2Fdownload%2FF%2FA%2F9%2FFA934B21-600C-4BC2-95D2-DFC5DCE93BEA%2FBusiness%2520Productivity%2520at%2520Its%2520Best%2520-%2520Office%25202010%2520and%2520SharePoint%25202010%2520white%2520paper.docx&usg=AFQjCNHhAUvGhMADUKAb5JEwKCnXc5rO2Q&bvm=bv.144686652,d.dGc

[3] Microsoft, *CNG DPAPI*, Retrieved Jan., 23, 2017, from http://msdn.microsoft.com/ko-kr/library/windows/desktop/hh706794(v=vs.85).aspx

[4] H. J. Kwon and S. J. Kim, “RFID distance bounding protocol secure against mafia and terroist fraud,” *J. KICS*, vol. 39, no. 11, pp. 660-674, Nov. 2014.

[5] B.-T. Kang and H. K. Kim, “A study on the

vulnerability of OTP implementation by using MITM attack and reverse engineering," *J. KIISC*, vol. 21, no. 6, pp. 86-99, Dec. 2011.

[6] W. C. Hong, K. W. Lee, and S. J. Kim, "Vulnerabilities analysis of the OTP implemented on a PC," *J. IPS*, vol. 17-C, no. 4, pp. 361-370, Aug. 2010.

[7] W. H. Ahn and H. Kim, "Attacking OpenSSL shared library using code injection," *J. KIISE*, vol. 37, no. 4, pp. 226-238, Aug. 2010.

[8] J. Song and I. Hwang, "A study on neutralization malicious code using windows crypto API and an implementation of crypto API hooking tool," *J. KIISC*, vol. 21, no. 2, pp. 111-117, Apr. 2011.

[9] J. Lee, J. Nam, S. Kim, and D. Won, "Present and future of SSL/TLS, WTLS," *R. KIISC*, vol. 14, no. 4, pp. 27-36, Aug. 2004.

[10] K. Lee, Y. Lee, J. Park, I. You, and K. Yim, "Security issues on the CNG cryptography library(Cryptography API: Next Generation)," in *Proc. IMIS*, pp. 709-713, Jul. 2013.

[11] K. Lee, I. You, and K. Yim, "Vulnerability analysis on the CNG crypto library," in *Proc. IMIS*, pp. 221-224, Jul. 2015.

[12] Microsoft, *SslEncryptPacket function*, Retrieved Jan., 23, 2017, from http://msdn.microsoft.com/en-us/library/windows/desktop/ff468663(v=vs.85).aspx, 2013. 11.

[13] Microsoft, *SslOpenProvider function*, Retrieved Jan., 23, 2017, from http://msdn.microsoft.com/en-us/library/windows/desktop/ff468682(v=vs.85).aspx, 2013. 11.

[14] Microsoft, *SslImportKey function*, Retrieved Jan., 23, 2017, from http://msdn.microsoft.com/en-us/library/ff468676.ASPX, 2013. 11.

[15] Y.-H. Goo, S.-O. Choi, S.-K. Lee, S.-M. Kim, and M.-S. Kim, "Tracking the source of cascading cyber attack traffic using network traffic analysis," *J. KICS*, vol. 41, no. 12, pp. 1771-1779, Dec. 2016.

이 경 룰 (Kyungroul Lee)

2008년 8월 : 순천향대학교 정보보호학과(공학사)
2010년 8월 : 순천향대학교 정보보호학과(공학석사)
2015년 2월 : 순천향대학교 정보보호학과(공학박사)
2011년 5월~2011년 12월 : (미) 퍼듀대학교 방문연구원
2015년 6월~2016년 2월 : 순천향대학교 박사후연구원
2016년 3월~현재 : 순천향대학교 연구조교수
<관심분야> 취약점 분석, 시스템 보안, 하드웨어 보안, 인터넷 뱅킹, 사용자 인증, 디바이스 인증

오 인 수 (Insu Oh)

2012년 3월~현재 : 순천향대학교 정보보호학과 학사과정
<관심분야> 취약점 분석, 디바이스 분석, 사물인터넷 보안, 모바일 보안

이 선 영 (Sun-Young Lee)

1993년 2월 : 부경대학교 전자계산학과(이학사)
1995년 2월 : 부경대학교 전자계산학과(이학석사)
2001년 3월 : 일본동경대학 전자정보공학(공학박사)
2004년 3월~현재 : 순천향대학교 정보보호학과 교수
<관심분야> 콘텐츠 보안, 암호이론, 정보이론, 정보보안

임 강 빈 (Kangbin Yim)

1992년 2월 : 아주대학교 전자공학과(공학사)
1994년 2월 : 아주대학교 전자공학과(공학석사)
2001년 2월 : 아주대학교 전자공학과(공학박사)
1999년 3월~2000년 2월 : (미) 아리조나주립대학교 연구원
2003년 3월~현재 : 순천향대학교 정보보호학과 교수
2005년 3월~현재 : 한국정보보호학회 이사
2009년 3월~현재 : 한국인터넷정보학회 이사
2010년 12월~2012년 2월 : (미)퍼듀대학교 객원교수
<관심분야> 시스템보안, 접근제어, 보안구조설계, 취약점분석, 개념증명도구개발