Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.26, NO.1, Feb. 2016 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2016.26.1.153

마이크로소프트의 차세대 암호 라이브러리 구조에 관한 연구 및 오류-검출 도구 구현*

이 경 률,[1†] 유 일 선,[2] 임 강 빈[1,2‡]
[1]순천향대학교 보안안전융합기술사업화센터, [2]순천향대학교 정보보호학과

An Analysis of a Structure and Implementation of Error-Detection Tool of Cryptography API-Next Generation(CNG) in Microsoft*

Kyungroul Lee,[1†] Ilsun You,[2] Kangbin Yim[1,2‡]
[1]R&BD Center for Security and Safety Industries, [2]Soonchunhyang University

요 약

본 논문은 마이크로소프트사의 CAPI를 대체하기 위하여 제안된 CNG의 구조와 특징, 프로그래밍 기법을 분석하였다. CNG는 플러그인 구조 기반의 독립된 모듈들로 구성되어 있기 때문에 구현해야 할 함수 및 기능의 범위를 최소화할 수 있어 개발비용과 확장 용이성 부분에서 CNG의 우수성을 잘 설명하고 있다. 또한, 확장성과 함께 최신의 암호화 알고리즘 및 감사 기능, 커널 모드 지원이 기업 및 공공기관 등의 환경에서 핵심 암호화 서비스로의 역할을 할 수 있게 한다. 따라서 이러한 기능들을 기반으로 기업과 공공기관이 조직 고유의 보안 요구사항에 맞게 CNG를 확장할 수 있도록 CNG 암호 라이브러리 구조에 대하여 분석하였다. 또한, 분석 결과를 기반으로 CNG 라이브러리를 활용하는 프로그램의 오류를 검출하기 위한 도구를 구현하였다.

ABSTRACT

This paper introduces a structure, features and programming techniques for the CNG(Cryptography API: Next Generation), which is the substitution of the CAPI(Cryptography API) from Microsoft. The CNG allows to optimize a scope of functions and features because it is comprised of independent modules based on plug-in structure. Therefore, the CNG is competitive on development costs and agility to extend. In addition, the CNG supports various functions for the newest cryptographic algorithm, audit, kernel-mode programming with agility and possible to contribute for core cryptography services in a new environment. Therefore, based on these advantageous functions, we analyze the structure of CNG to extend it for the enterprise and the public office. In addition, we implement an error-detection tool for program which utilizes CNG library.

Keywords: CNG, Cryptography API (CAPI), Cryptography library, Error-detection tool

I. 서 론

전자상거래, 개인정보보호 등의 목적을 위하여 암/복호를 이용한 데이터의 보호가 필수적인 요소가 되었고, 시스템에서 이를 활용하기 위하여 라이브러리 형태로 배포되었으며, 이는 흔히 암호 라이브러리 또는 암호 API라 불린다. 국외 암호 라이브러리를 살펴보면 IETF의 GSS-API, The Open Group의 GCS-API, 마이크로소프트의 Cryptography API(CAPI), RSA의 PKCS#11, Intel의 CDSA

Received(12. 11. 2015), Modified(02. 05. 2016), Accepted(02. 07. 2016)
* 본 연구는 2015년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No. NRF-2015R1A6A3A01019717, No. NRF-2015R1D1A1A01057300)
† 주저자, carpedm@sch.ac.kr
‡ 교신저자, yim@sch.ac.kr(Corresponding author)

CSSM API 등이 있으며[1], 국내에는 퓨처시스템의 Cryp Tool, 이니텍의 INI Crypto, STI의 J/LOCK, 장미디어의 CEAL98, 트러스컴의 trusCrypt 등이 있다. 하지만 처음 개발되고 배포된 라이브러리들은 현재 급변하는 하드웨어, 플랫폼, 운영체제 등의 속도에 따라가지 못하고 있는 실정이며, 이는 설계상의 문제이다. 특히 마이크로소프트의 CAPI는 윈도우즈 운영체제가 XP에서 비스타, 윈도우즈 7로 업그레이드되면서 기존의 암호 라이브러리 CAPI를 대체하기 위한 수단이 필요하게 되었으며, 이에 마이크로소프트는 Cryptography API: Next Generation(CNG)를 제안하였다[2, 3, 4, 5]. 제안된 CNG는 최신 암호 알고리즘과 커널 모드, 감사 기능 등을 지원하며, 이후 새로운 요구사항이 도출되거나 환경이 변화하더라도 이를 유연하게 대응할 수 있도록 민첩성(agility)을 제공하고 있어 차세대 암호 라이브러리로서의 역할을 하고 있다.

과거 CAPI는 새로운 요구사항이나 알고리즘, 환경이 변화하게 되면 이를 대응하기 위해 각 요소별 대응방안을 별도로 관리해야 했으며, 알고리즘 역시 공통적으로 사용되는 부분들도 매번 새로이 작성해야 하기 때문에 급속히 변화하는 환경에 민첩하게 대응하지 못하였다. 따라서 이를 보완하고자 등장한 CNG는 CAPI의 민첩성을 가장 중요한 요소로 다루었으며[9], 이를 지원하기 위해 개발자가 쉽게 접근할 수 있도록 플러그인 모델 기반의 공급자(CNG provider) 개념을 도입하였다. 플러그인 모델 기반의 공급자는 새로운 암호 알고리즘이나 키 관리 방안 등 새로운 요구가 있을 경우, 개발자가 이를 구현한 후 기존에 존재하고 있는 라이브러리에 추가하는 방식으로 매번 새로이 개발할 필요가 없기 때문에 보다 효율적이다. CAPI에서는 Cryptographic Service Provider(CSP)를 새로이 개발하고, 이를 검증하기 위하여 마이크로소프트에 요청하여 서명을 받아야 했기 때문에 이와 같은 구조로 인하여 환경 변화에 민첩하게 대응하지 못하며, CNG를 활용할 경우에는 보완이 가능하다. 따라서 본 논문에서는 마이크로소프트에서 제공하는 CNG의 구조와 특성을 분석하고, 프로그래밍 기법에 대해 각 요소별로 분석하고자 한다. 또한, 분석 결과를 기반으로 CNG 라이브러리를 활용하는 프로그램에서 발생하는 오류를 검출하기 위한 도구를 구현하고자 한다.

본 논문의 구성은 다음과 같다. 제2장에서 CNG의 전체 구조 및 그 특징을 살펴보고, 제3장에서 프로그래밍 기법들에 대해 분석한 결과를 서술한 후, 제4장에 구현한 오류-검출 도구에 대해 서술하며, 제5장에 결론을 도출한다.

II. CNG의 전체구조 및 특징분석

2.1 CNG 개요 및 구조

1장에서 서술한 문제를 해결하기 위하여 제안된 CNG는 기존에 제시되었던 모든 암호 프리미티브를 대체하고 있으며, CAPI가 제공하던 모든 암호 알고리즘을 지원할 뿐만 아니라 추가적으로 더 많은 새로운 알고리즘을 제공한다. 따라서 개발자가 이들을 복합적으로 운용하여 다양한 기능을 만들어 내는 방법과 수행하는 방법 등에 대해 보다 적극적으로 개입할 수 있으므로 보다 유연한 설계 구조를 지원한다. CNG의 전체적인 구조를 Fig. 1에 나타내었다.

CNG는 난수생성기, 해쉬 함수, 서명 및 암호키 등과 관련된 암호 프리미티브를 제공하고 그 구조의 핵심은 BCrypt이며, NCrypt는 비대칭키 및 스마트카드 등의 하드웨어를 지원하기 위한 키 저장 기능을 제공한다. BCrypt의 B는 Base, NCrypt의 N은 New를 의미하며, CNG가 제공하는 개발환경에서의 헤더나 DLL 파일 이름에서 유래한다. CNG는 이전까지의 암호 라이브러리와 다르게 유저 모드, 커널 모드를 동시에 지원하기 위한 프레임워크를 제공하고 있으며(BCrypt), 키 보관 기능(NCrypt)은 유저 모드에서만 지원이 가능하다.

CNG가 제공하는 암호 프리미티브에 대한 관점은 두 가지가 있다. 첫 번째 관점은 CNG를 로컬 오브젝트의 집합으로 간주하는 것으로 제공되는 오브젝트는 메소드와 더불어 질의 및 변경 가능한 속성을 제공한다. 이들 오브젝트는 알고리즘 공급자(algorithm provider), 해쉬 함수, 키 및 비밀 합의(secret agreement) 등을 포함하며, 난수생성은

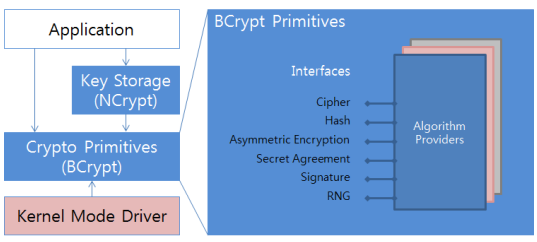

Fig. 1. The whole structure of CNG

알고리즘 공급자에 의하여 내부에서 직접 처리되고 기타의 것들은 모두 키를 통하여 제공된다. 여기에서 키는 대칭 또는 비대칭 키 형태로 표현될 수 있으며, 해쉬 서명을 수행하거나 검증하기 위해 사용된다. 해쉬 및 키 오브젝트는 알고리즘 공급자로부터 얻어지며, 비밀 합의는 키 오브젝트 쌍으로부터 얻어진다. 두 번째 관점은 CNG를 암호화 연산을 위한 소프트웨어 라우터 또는 중계자로 간주하는 것이다. CNG의 API는 논리 암호화 인터페이스 집합 위에 구축되어 있기 때문에 가령 특정의 해쉬 알고리즘을 구현하는 경우 하드-코딩하는 것을 배제하고 해쉬 인터페이스를 활용하여 구현이 가능하다. 대개의 암호화 API들은 알고리즘에 중심을 둔 접근방식을 사용하지만, CNG의 경우는 인터페이스에 중심을 둔 접근방식을 사용하기 때문에 개발자뿐만 아니라 경우에 따라 이미 공급된 응용 프로그램에서 결함이 있는 것으로 판명된 알고리즘을 교체해야 하는 응용 프로그램 관리자에게도 더 나은 민첩성을 제공하여 빠른 대처가 가능하다.

2.2 CNG의 주요 특성

CNG는 암호 민첩성, 연방표준 준수, Suite B 지원, 커널 모드 지원, 감사 기능 등의 주요 특성을 제공하며, 그 특성을 상세히 살펴보면 다음과 같다.

2.2.1 암호민첩성

CNG가 제안하고 있는 주요 가치 중의 하나는 암호 불가지론으로 일컫는 암호 민첩성이다. 암호 민첩성은 SSL/TLS, CMS, IPsec, Kerberos 등의 프로토콜 구현물을 CNG로 변환하는 작업에서 그 가치를 발휘하였는데 실제로 CNG의 입장에서는 대칭, 비대칭, 해쉬 함수 등의 모든 알고리즘 타입, 난수 발생기 및 기타 유용한 함수들을 대체해야만 했다. 프로토콜 수준에서의 변화는 많은 경우 프로토콜 API들이 기존에 없던 알고리즘 선택 기능이나 기타의 유연성을 위한 옵션들을 제공할 필요가 있었기 때문에 매우 중요한 것이라 판단된다.

2.2.2 연방표준 적합성 및 호환성 준수

CNG는 특정의 플랫폼에 대하여 공통평가기준과 함께 연방정보처리표준(FIPS) 140-2의 레벨 2인증을 목표로 하고 있으며, 기타의 플랫폼에서는 레벨 1 인증을 목표로 하고 있다. 이러한 플랫폼들은 동일한 CNG 구현물을 수행하지만, 다른 인증 수준을 제공한다.

2.2.3 Suite B 지원

CNG의 또 다른 중요한 특징 중의 하나는 Suite B 알고리즘을 지원하는 것이다. 2005년 2월, 미연방 국가보안국은 대칭키 암호화, 키 교환으로 알려진 비대칭 비밀 동의, 디지털 서명 및 해쉬 함수 등 미국 정부가 향후 사용할 Suite B라 불리는 잘 정돈된 암호 세트를 공개했다[6]. 미 정부는 인증된 Suite B 구현물이 과거 SBU(Sensitive But Unclassified)로 서술되던 최상위 기밀정보 및 개인정보로 분류되는 정보를 보호하기 위해 사용될 것이라 발표했다. 따라서 Suite B를 지원하는 것은 마이크로소프트뿐만 아니라 응용 프로그램 공급자 및 SI 업체에게도 매우 중요한 의미를 가진다.

2.2.4 커널 모드 지원

CNG에서는 커널 모드와 유저 모드 모두에서 동일한 API를 사용함으로써 보다 유연하게 암호화 기능을 지원한다. CNG를 사용하는 SSL/TLS 및 IPSec은 부트 프로세스에 추가되어 커널 모드에서 수행되지만 모든 CNG 함수가 커널 모드에서 수행되는 것은 아니다. 커널 모드에서 호출할 수 없는 것으로 명시된 함수들을 제외하고는 호출하는 프로세스의 IRQL이 PASSIVE_LEVEL로 되어 있는 경우 모든 CNG 함수를 호출할 수 있다. 마이크로소프트 커널 보안 지원 공급자 인터페이스인 Ksecdd.sys는 윈도우즈의 커널 모드 레벨에서 상주하는 범용의 소프트웨어 기반 암호 모듈이다. 이 드라이버는 커널 모드의 익스포트 드라이버로 실행되며, 커널 컴포넌트를 위하여 공개된 인터페이스를 이용하여 암호화 서비스를 제공하지만, DSA 알고리즘의 경우 예외적으로 이를 지원하지 않는다. CNG는 커널 모드에서 알고리즘이나 공급자를 상시 추가하는 기능을 제공하지 않으며, 커널 모드에서 제공하는 알고리즘으로는 마이크로소프트가 커널 모드 CNG API를 통하여 제공하는 구현물뿐이다.

2.2.5 감사 기능

상기의 보안 기능 제공과 더불어 공통평가기준의 몇몇 요구사항을 만족하기 위하여 CNG 계층에서 일어나는 많은 동작들은 마이크로소프트의 소프트웨어 키 저장 공급자, 즉, KSP(Key Storage Provider)에 의하여 감사된다. KSP는 하나의 가이드라인으로서 다음과 같은 경우 반드시 감사하여 보안 로그 안에 감사 레코드를 생성해야 한다.

- 자가 진단 실패를 포함한 키 및 키 쌍 생성 실패
- 키 반입 및 반출
- 키 폐기 실패
- 지속적으로 사용되는 키가 파일에서 읽히고 쓰이는 경우
- 키 쌍의 일관성 검사 실패
- 3-DES 키에 대한 패리티 검사 등 비밀키 유효성 검사 실패
- 암호화, 복호화, 해쉬, 서명, 검증, 키 교환, 난수 생성 등의 실패
- 암호화 자가진단

III. CNG의 프로그래밍 기법 분석

CNG를 사용하여 프로그래밍하기 위해서는 다음과 같은 전형적인 단계가 필요하다[10].

- 알고리즘 공급자 열기(Open) - BCryptOpenAlgorithmProvider 함수 사용
- 알고리즘 속성 조회 및 설정 - BCryptGetProperty와 BCryptSetProperty 함수 사용
- 키 생성 및 반입 - 대칭키 생성을 위해 BCryptGenerateSymmetricKey 함수와 비대칭 키 생성을 위해 BCryptGenerateKeyPair 함수 사용, 키 반입을 위해 BCryptImportKey와 BCryptImportKeyPair 함수 사용
- 암/복호화 연산 수행 - 암호화를 위해 BCryptEncrypt 함수와 복호화를 위해 BCryptDecrypt 함수 사용
- 알고리즘 공급자 닫기(Close) - BCryptCloseAlgorithmProvider 함수 사용

본 장에서는 위의 단계를 기본으로 CNG가 제공하는 알고리즘 공급자, 난수생성, 해쉬 함수, 대칭 및 비대칭 암호화, 서명 및 검증 기능 등의 프로그래밍 기법을 살펴본다.

3.1 알고리즘 공급자

알고리즘 공급자는 CNG의 기본일 뿐만 아니라 가장 중요한 요소이며, BCrypt 내에 구성되어 있다. BCrypt에 의하여 정의되는 모든 CNG 오브젝트는 BCRYPT_HANDLE에 의해 식별되며, 알고리즘 공급자도 마찬가지이다.

BCryptOpenAlgorithmProvider 함수는 알고리즘 및 추가적 구현에 대한 사용자의 선택을 바탕으로 알고리즘 공급자를 로드하며, 이후 CNG 함수 호출에 사용하기 위한 핸들을 반환한다. 또한, BCrypt는 유저 모드와 커널 모드에 상관없이 오류정보를 나타내기 위하여 Windows Driver Kit(WDK)의 NTSTATUS 타입을 사용하고 있다. 알고리즘 공급자의 로드, 언로드 과정은 Fig. 2와 같다.

BCryptOpenAlgorithmProvider 함수를 사용

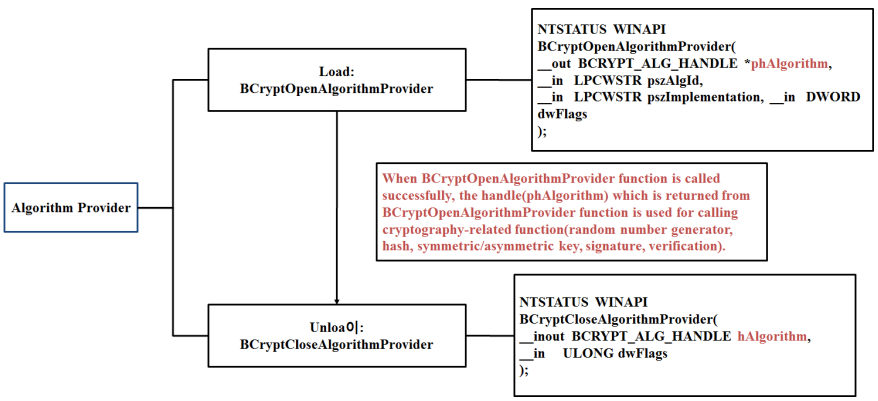

Fig. 2. Process calling the algorithm provider function

함에 있어 대개의 경우, pszImplementation 및 dwFlags 인자에 0을 전달한다. pszImplementation에 0을 전달하는 것은 pszAlgId 인자에 의하여 선택되는 특정 알고리즘을 위한 디폴트 알고리즘 공급자를 로드하도록 지시한다. 알고리즘 공급자를 다 사용한 후에는 다음과 같이 BCryptOpenAlgorithmProvider에 의하여 반환되었던 핸들을 BCryptCloseAlgorithmProvider 함수에 전달함으로써 알고리즘 공급자를 언로드해야 한다.

3.2 난수 생성

난수의 생성은 BCryptGenRandom 함수에 의해 이루어진다. 난수 생성 과정은 Fig. 3과 같다. 난수의 생성은 BCryptOpenAlgorithmProvider 함수 호출 시 알고리즘 식별자를 지정하고, BCryptGenRandom 함수를 호출하면 버퍼에 생성된 난수가 저장된다. 알고리즘 식별자는 선택적으로 사용 가능하며, 이를 Table 1에 나타내었다.

3.3 해쉬 함수

난수의 생성과 마찬가지로 해쉬 함수는 현대 컴퓨팅 환경의 다양한 측면에서 보안의 성향과 특징을 결정짓는 절대적인 역할을 한다. CNG에서는 해쉬 함수도 오브젝트로 표현되는데 API가 커널 모드의 코드에 의해서도 접근이 가능해야 하기 때문에 이러한 오브젝트를 만들어 내기 위하여 약간의 준비 작업이 요구된다. 물론 이들은 CNG 내의 오브젝트로서 BCrypt의 일부이므로 특정 오브젝트를 위해 명명된 프로퍼티를 읽거나 쓰기 위해 BCryptGetProperty 및 BCryptSetProperty 함수를 이용하며, NCrypt와 관련된 오브젝트 프로퍼티를 처리하기 위한 동일 기능의 함수들도 제공된다.

BCryptCreateHash 함수는 새로운 해쉬 오브젝트를 생성하는데 사용되며, 함수를 호출하기 전에 함수가 해당 작업을 처리하기 위하여 사용할 버퍼를 미리 할당하여야 한다. 이러한 상황에서 동일한 API를 가지고 유저 모드와 커널 모드를 동시에 지원하는 방식의 부작용이 나타나는데, 커널 모드는 메모리가 어디에 할당되는지, 특히, 할당 영역이 페이지 풀인지 비 페이지 풀인지에 대하여 매우 민감한 특징을 가지고 있다. 버퍼 이외에도 사용자는 해쉬 오브젝트를 위한 핸들이나 해쉬 테이블 관련 자원들을 관리하여야 한다. 해쉬 생성 및 그와 관련된 연산 과정을 Fig. 4에 나타내었다.

해쉬 오브젝트를 생성하기 위하여 요구되는 버퍼의 크기는 알고리즘 공급자의 프로퍼티 중 하나인 BCRYPT_OBJECT_LENGTH 식별자를 통하여 참조 가능하다. 속성을 설정할 때 사용하는 식별자의 종류는 Table 2와 같다.

BCryptGetProperty의 첫 인자는 프로퍼티를 찾기 위한 오브젝트를 가리키며 두 번째 인자는 해당 프로퍼티의 이름을 전달한다. 세 번째 및 네 번째 인자는 프로퍼티 값과 버퍼 크기가 저장될 공간을 나타낸다. pcbResult 인자는 예상되는 버퍼의 크기를

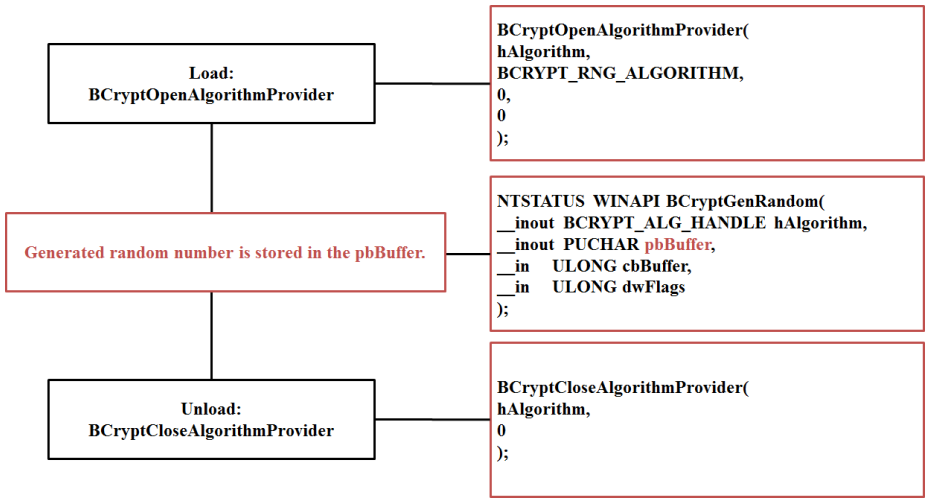

Fig. 3. Process calling the random number generation function

Table 1. Algorithm identifiers[7]

Algorithm Identifier	Description
BCRYPT_3DES_ALGORITHM ("3DES")	3-DES cipher algorithm standard: SP800-67, SP800-38A
BCRYPT_3DES_112_ALGORITHM ("3DES_112")	112bits 3-DES cipher algorithm standard: SP800-67, SP800-38A
BCRYPT_AES_ALGORITHM ("AES")	AES cipher algorithm standard: FIPS 197
BCRYPT_AES_GMAC_ALGORITHM ("AES-GMAC")	AES GMAC cipher algorithm standard: SP800-38D *Only supports Vista SP1 and Server 2008
BCRYPT_DES_ALGORITHM ("DES")	DES cipher algorithm standard: FIPS 46-3, FIPS 81
BCRYPT_DH_ALGORITHM ("DH")	Diffie-Hellman key exchange algorithm standard: PKCS #3
BCRYPT_DSA_ALGORITHM ("DSA")	Digital signature algorithm standard: FIPS 186-2
BCRYPT_ECDH_P256_ALGORITHM ("ECDH_P256")	256bits elliptic curve Diffie-Hellman key exchange algorithm standard: SP800-56A
BCRYPT_MD2_ALGORITHM ("MD2")	MD2 hash algorithm standard: RFC 1319
BCRYPT_RC2_ALGORITHM ("RC2")	RC2 block cipher algorithm standard: RFC 2268
BCRYPT_RNG_ALGORITHM ("RNG")	Random number generation algorithm standard: FIPS 186-2, FIPS 140-2, NIST SP 800-90
BCRYPT_RSA_ALGORITHM ("RSA")	RSA public key algorithm standard: PKCS #1 v1.5, v2.0
BCRYPT_SHA1_ALGORITHM ("SHA1")	SHA1 hash algorithm standard: FIPS 180-2, FIPS 198

사전에 알 수 없을 때 유용하다. 현재 플래그 값은 정의되어 있지 않으므로 0을 전달한다.

일단 해쉬 버퍼의 크기가 얻어지면 해당 해쉬 오브젝트를 위한 메모리를 할당하여야 하며, 커널 모드가 아닌 안전한 메모리일 경우, 메모리를 할당하는 위치나 크기는 별 문제가 되지 않는다. 예를 들어 버퍼 할당을 위하여 기본적인 Buffer 클래스를 활용할 수 있으며, 버퍼 할당을 마치면 BCryptCreateHash 함수를 이용하여 해쉬 오브젝트를 생성한다. BCryptCreateHash의 첫 번째 인자는 해쉬 인터페이스를 구현하고 있는 알고리즘 공급자를 의미하며, 두 번째 인자는 생성되는 해쉬 오브젝트의 핸들

Table 2. Property Identifiers[8]

Property Identifier	Description
BCRYPT_ALGORITHM_NAME	Algorithm name
BCRYPT_AUTH_TAG_LENGTH	Length of Authentication
BCRYPT_BLOCK_LENGTH	Block size
BCRYPT_BLOCK_SIZE_LIST	List of block size
BCRYPT_CHAINING_MODE	Setting mode (BCRYPT_CHAIN_MODE_CBC, BCRYPT_CHAIN_MODE_CCM, BCRYPT_CHAIN_MODE_CFB, BCRYPT_CHAIN_MODE_ECB, BCRYPT_CHAIN_MODE_GCM, BCRYPT_CHAIN_MODE_NA)
BCRYPT_DH_PARAMETERS	Parameter of Diffie-Hellman key
BCRYPT_DSA_PARAMETERS	DSA parameter
BCRYPT_EFFECTIVE_KEY_LENGTH	Effective key length of RC2 key
BCRYPT_HASH_BLOCK_LENGTH	Hash block size
BCRYPT_HASH_LENGTH	Hash length
BCRYPT_KEY_LENGTH	Symmetric key length
BCRYPT_KEY_STRENGTH	Number of bits in the key
BCRYPT_OBJECT_LENGTH	Object length
BCRYPT_SIGNATURE_LENGTH	Key length for signature

을 의미한다. 그 다음 두 인자는 버퍼와 그 크기를 지시하고, pbSecret와 cbSecret은 키 기반 해쉬 알고리즘의 경우 비밀키를 제공하기 위해 사용된다. 상기와 마찬가지로 dwFlags는 현재 플래그가 정의되어 있지 않으므로 0을 전달한다. 해쉬 오브젝트를 다 사용한 후에는 BCryptDestroyHash 함수를 사용하여 오브젝트를 폐기하고 할당한 버퍼를 해제해야 한다.

위에서 살펴 본 해쉬 오브젝트를 생성하고 폐기하는 방법을 실제로 활용하기 위해서는 해쉬 오브젝트를 생성한 후, BCryptHashData 함수를 이용하여 주어진 버퍼를 상대로 단방향 해쉬를 수행하며, 해당 함수를 반복적으로 수행함으로써 버퍼에 지속적으로 공급되는 새로운 값에 대하여 해쉬에 병합해야 한다. 이 때, 해쉬 값은 고정된 크기로서 알고리즘 공급자에 의하여 결정되므로 BCryptHashData 함수를 몇 번을 호출하던지 간에 그 결과 값은 항상 같은 크기임을 명심해야 한다. BCryptHashData의 첫 번째

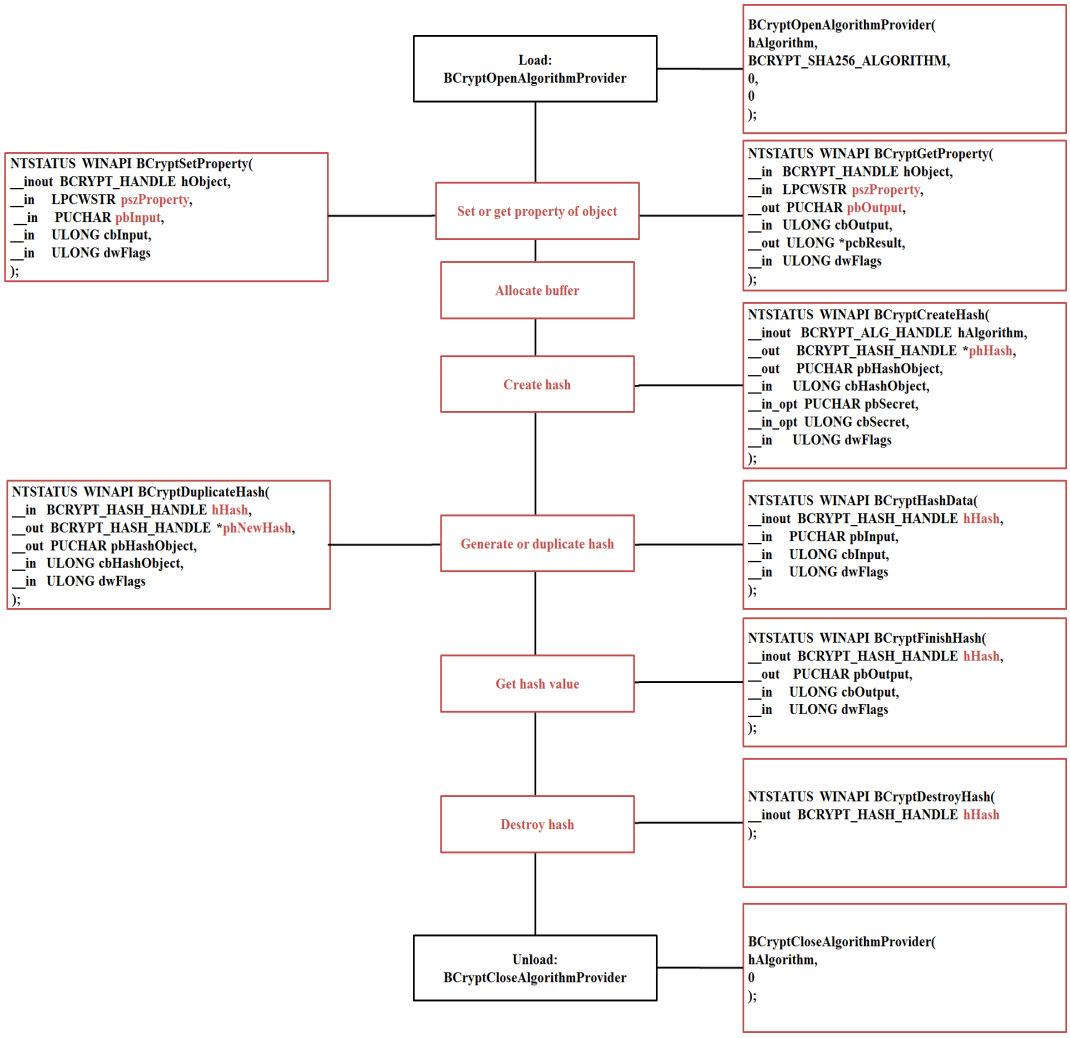

Fig. 4. Process calling the hash function

인자는 BCryptCreateHash 함수로부터 얻어진 핸들이며 그 다음 두 인자는 해쉬를 수행하기 위한 데이터를 담은 버퍼와 그 크기를 가리킨다. 정확한 버퍼의 크기를 전달하기 위해 안전하게 cbInput의 확인절차를 거칠 수 있으며, 마지막 인자 dwFlags는 현재 정의되어 있지 않으므로 반드시 0을 전달한다.

일단, 모든 데이터를 해쉬 함수에 전달한 후에는 BCryptFinishHash 함수의 호출을 통하여 해쉬 결과 값을 얻는다. 결과 해쉬 값의 크기는 사용하는 해쉬 알고리즘에 따라 달라지는데 만약 그 크기를 모를 경우, 이를 BCRYPT_HASH_LENGTH 프로퍼티를 이용하여 해쉬 오브젝트로부터 읽어 낸다. 현재까지의 함수들과 유사한 형태로서 BCryptFinishHash 함수의 첫 번째 인자는 해쉬 오브젝트의 핸들을 나타내고, 그 다음 두 인자는 해쉬 결과 값을 받을 버퍼 및 그 크기를 나타낸다. 마지막 인자인 dwFlags는 현재 정의되어 있지 않으므로 0을 전달한다.

해쉬 오브젝트는 복제될 수 있어서 공통적인 데이터를 바탕으로 두 개 이상의 해쉬 값을 생성하고자 할 때 유용하다. 이는 공통의 데이터를 해쉬하기 위한 하나의 해쉬 오브젝트를 생성한 후, 이를 하나 또는 그 이상으로 복제하고 그 복사본에 델타를 추가하는 것으로 가능하다. 일단 하나의 해쉬 오브젝트가 복제되면 두 해쉬 오브젝트는 동일한 상태를 가지게 되나, 서로 별도의 것으로서 각각에 고유한 데이터를

추가하여 독립적인 해쉬 값을 생성할 수 있으며, 서로에게 어떤 방법으로든 전혀 영향을 주지 않으면서 하나의 해쉬 오브젝트를 폐기할 수도 있다. 이는 BCryptDuplicateHash 함수에서 제공하고 있으며, 복제할 해쉬 오브젝트의 핸들과 복제되는 오브젝트가 사용할 새로운 버퍼만을 요구한다.

3.4 대칭키 암호화

대칭키 암호화는 대칭키와 이를 위한 오브젝트 버퍼를 준비하여야 한다는 점에서 해쉬 함수와 매우 유사한 특성을 가진다. 즉, 앞에서 설명한 절차와 같이 알고리즘 공급자를 생성하고 그에 속한 BCRYPT_OBJECT_LENGTH 프로퍼티를 찾아 키 오브젝트 버퍼의 사이즈를 결정하며 그 크기만큼의 버퍼를 할당한다. 키 값을 인자로 하여 BCryptGenerateSymmetricKey 함수를 호출함으로써 비밀키 오브젝트를 생성한다. 그 절차를 Fig. 5에 보인다.

BCryptGenerateSymmetricKey 함수의 첫 번째 인자는 대칭키 암호화 알고리즘을 구현하는 알고리즘 공급자를 가리키며, 두 번째 인자는 키 오브젝트 핸들의 수신을 위한 것이다. 다음 두 인자는 키 오브젝트 버퍼 및 그 크기를 나타내고, 그 다음 두 인자는 송신자와 수신자에 의하여 공유되는 대칭키를 전달할 버퍼 및 크기를 가리킨다. 대칭키는 임의의 바이트 어레이이나 패스워드 등이 사용된다. 키 오브젝트를 사용하고 난 후에는 BCryptDestroyKey 함수를 이용하여 폐기한 후 오브젝트 버퍼를 해제하여야 한다.

3.5 비대칭키 암호화

비대칭키 암호화는 양자 간 개인키와 공개키를 유지하는 공개키 방식을 취함으로써 초기화 벡터 및 비밀정보의 공유 문제를 해결하는 것으로, 공개키는 누구에게나 공개된다. 키 쌍은 해당 공개키로 암호화된 암호문을 해당 개인키만으로 복호화할 수 있도록 짝을 이룬다. 이러한 기능을 실현하려면 당연히 대가가 따르는데 비대칭키 암호화는 그 계산량을 비교할 때 대칭키 암호화와는 상당한 차이가 있는 것으로 알려져 있지만, 비밀키 정보를 상호 비밀리에 공유하기 위한 구조를 포함하는 통신에서는 매우 유용하다. 비대칭키 암호화의 절차는 Fig. 6과 같다.

공개키 및 개인키 쌍을 생성하기 위해서는 BCryptGenerateKeyPair 함수를 호출한다. 첫 번째 인자는 비대칭 암호화 알고리즘을 구현하고 있는 알고리즘 공급자를 가리키며, 두 번째 인자는 키 오브젝트의 핸들을 받기 위한 것이다. 세 번째 인자는 알고리즘이 사용할 키의 크기를 가리키는데, 단위는 바이트가 아닌 비트로 표현되고 알고리즘마다 서로 다른 값을 가진다. 예를 들어, RSA 알고리즘의 경우는 512비트부터 16384비트 사이의 64의 배수 크기의 키를 지원한다.

일반적으로 키의 크기가 커지면 알고리즘 성능이 저하되지만, 무차별 공격을 통한 무력화 공격에 더욱 강인해진다. 이러한 우려성의 이유가 아니더라도 키의 크기는 알고리즘이 사용하는 블록의 크기를 의미한다는 점에서도 매우 중요한 의미가 있다. 블록의 크기는 키 크기를 8로 나눔으로써 얻는다. 우선 BCryptGenerateKeyPair 함수를 이용하여 키 쌍을 생성한 후, BCryptSetProperty 함수를 이용하여 알고리즘에 종속적인 다양한 프로퍼티들을 설정하며, 키 생성을 마무리하기 위해 BCryptFinalizeKeyPair 함수를 호출한다. 상기와 같은 준비 절차가 이루어진 후에는 기 서술한 바와 같이 BCryptEncrypt 및 BCryptDecrypt 함수를 이용하여 데이터를 자유롭게 암호화 및 복호화할 수 있다. 다만, 대칭키에서와 약간 다른 점은 비대칭키의 경우는 초기화 벡터를 필요로 하지 않으며, 사용자가 선택한 플래그에 따라 좀 더 복잡한 패딩방식이 수반된다는 것이다. 암호화할 메시지가 블록 크기의 정수배 길이인 버퍼를 통하여 제공한다는 가정을 하면 패딩이 필요 없으므로 플래그로 BCRYPT_PAD_NONE을 지정한다. 반면에 메시지의 암호화 이전에 패딩을 해야 하는 경우에는 몇 가지 가능한 옵션이 있다. 가장 단순한 옵션은 BCRYPT_PAD_PKCS1 플래그를 지정함으로써, 알고리즘 공급자로 하여금 PKCS-1 표준에 근거하여 난수를 이용하여 버퍼를 패딩하도록 하는 것이다. 물론 키 쌍의 공개 부분을 공유하지 못할 경우 비대칭 암호화의 유용성이 크게 손상되겠지만, 키를 반입하거나 반출하는데 BCryptExportKey 함수 및 BCryptImportKeyPair 함수를 사용하여 이를 처리한다. 다만, BCryptExportKey 함수는 대칭키를 반출하는데 사용하지만, 반입의 경우에는 버퍼를 키 오브젝트와 연결시켜야 하므로 BCryptImportKey 함수를 필요로 한다. BCryptExportKey 및 BCryptImportKeyPair 함수는 완전한 키 쌍을

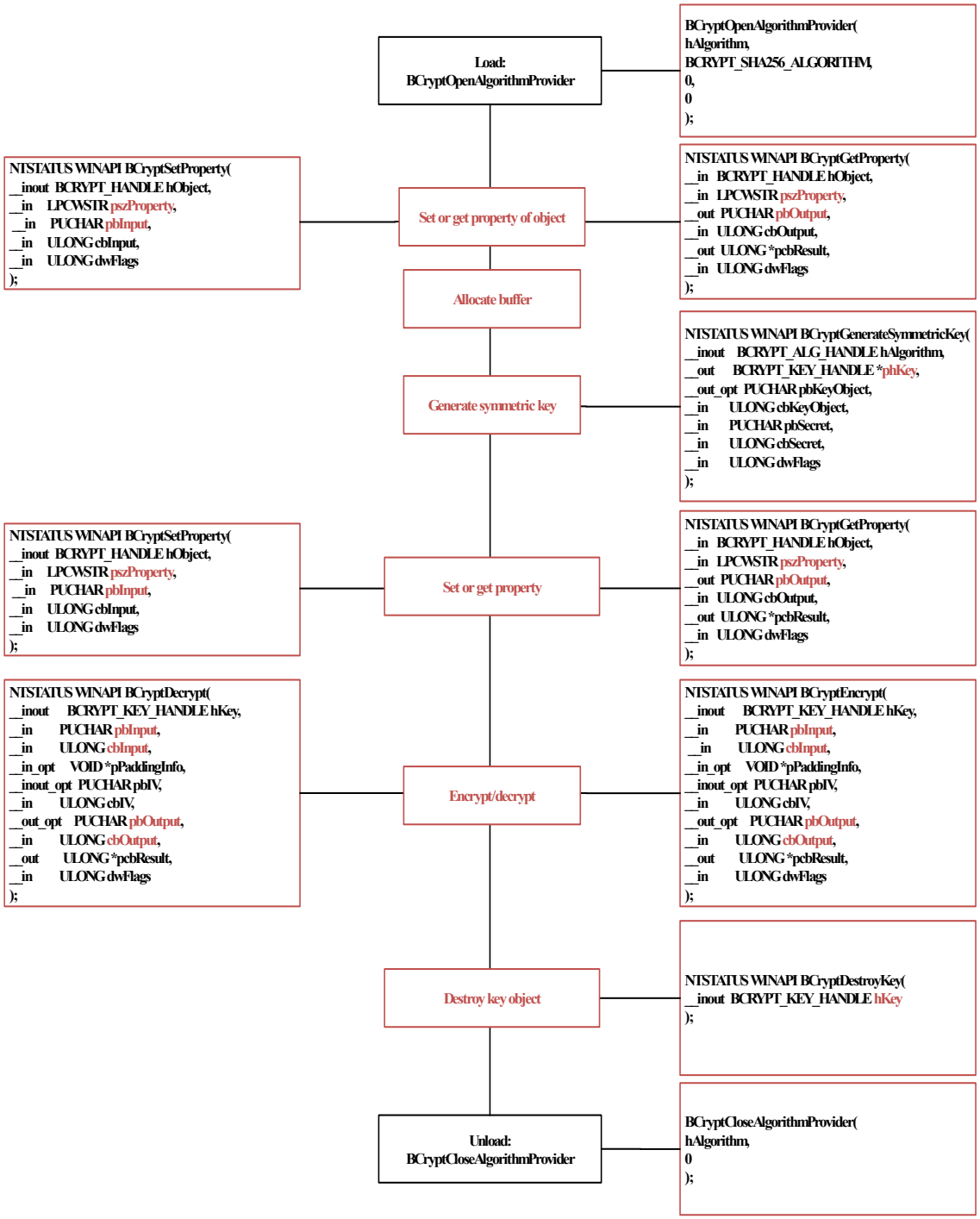

Fig. 5. Process calling the symmetric key encryption/decryption

반출할 수도 있고, 공개키만을 반출할 수도 있다. 예를 들어, 키 쌍을 반출한 후, 차후의 활용을 위하여 보관할 수도 있고, 통신을 요하는 상대에게 공개키를 제공할 수 있다.

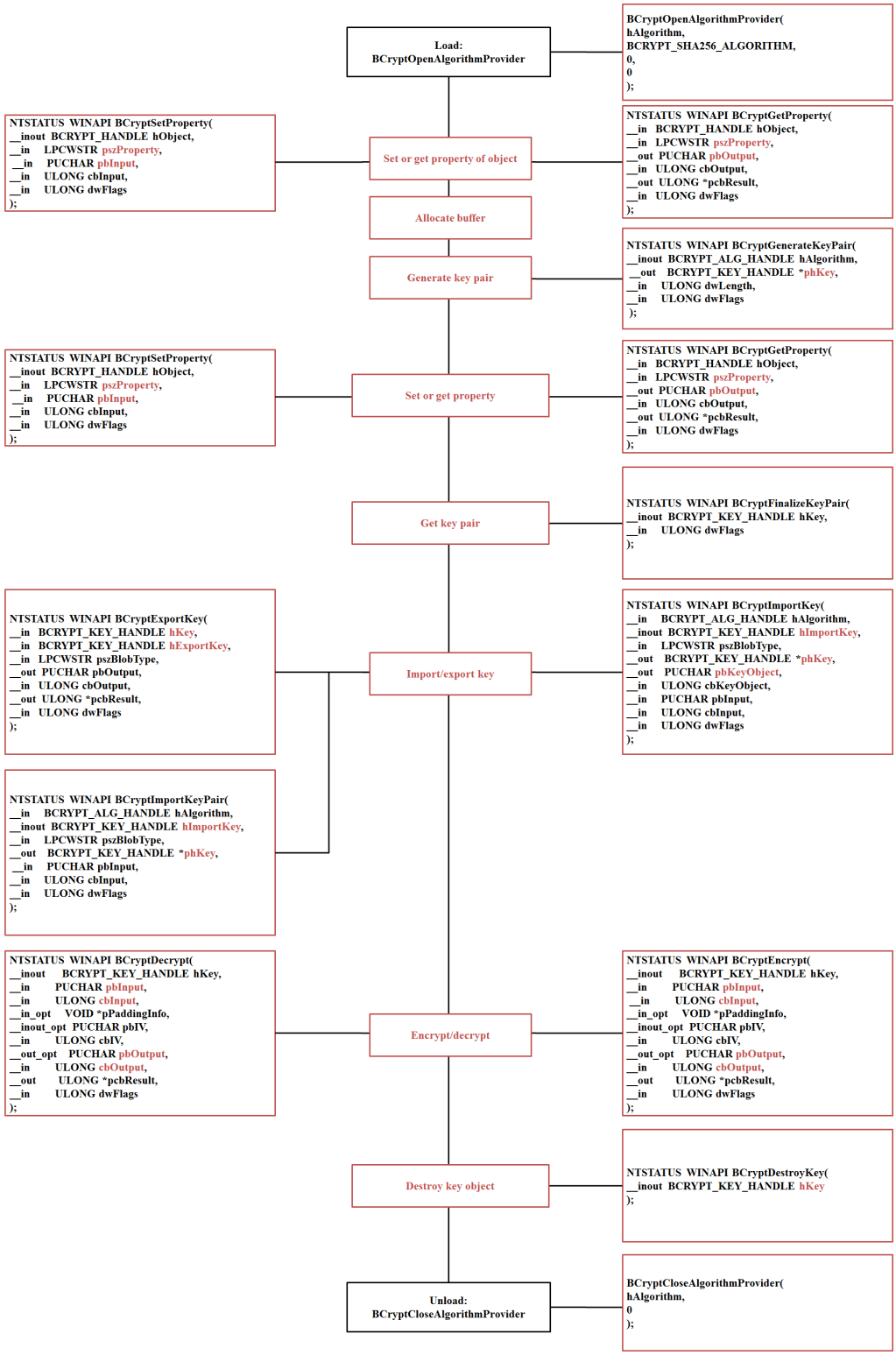

Fig. 6. Process calling the Asymmetric key encryption/decryption function

3.6 서명 및 검증

비대칭 암호화는 대개 디지털 서명을 생성하기 위하여 사용한다. 이는 메시지나 실행파일 등의 무결성을 입증하기 위하여 윈도우즈 및 마이크로소프트 닷넷 프레임워크에서 주요하게 사용된다. 서명은 이미 서술한 바와 같이 비대칭 암호화 및 복호화 절차와는 다르게 비밀키에 의하여 생성되고 공개키에 의하여 검증된다. 그렇게 함으로써 공개키를 소유한 자로 하여금 비밀키 없이 주어진 서명이 해당 비밀키를 소유한 자에 의해 이루어진 것임을 검증 가능하도록 해준다. 이미 살펴 본 내용을 기반으로 하면 서명을 생성하고 검증하는 절차는 매우 간단하다. 서명하고자 하는 데이터를 선택하고, 그 데이터에 해쉬 연산을 수행한다. 그 후, 해쉬 결과 값과 공개키를 이용하여 BCryptSignHash 함수를 호출하면 서명이 계산된다. Fig. 7은 이를 위한 예를 보여주고 있다. 마찬가지로, 처음 호출은 결과 서명의 크기를 계산하고 두 번째는 실제 서명을 수행하며, 해쉬 값의 크기 및 사용 알고리즘에 따라 추가적인 패딩정보가 필요해질 수 있다. 서명의 검증은 더욱 간단하다. 해쉬를 독립적으로 수행하고 그 해쉬 값과 서명자의 공개키 및 수신한 서명 값을 BCryptVerifySignature 함수에 전달하는 것이다. BCryptVerifySignature 함수는 서명 값이 해쉬 값과 일치하면 STATUS_SUCCESS를, 그렇지 않은 경우는 STATUS_INVALID_SIGNATURE 를 반환한다.

IV. 오류-검출 도구 구현

상기 CNG의 구조에 대한 분석 결과를 토대로 CNG를 활용하는 프로그램에서 발생 가능한 오류를 검출하는 도구를 구현하였다. 프로그래밍에서 발생하는 오류의 주된 원인은 개발자의 실수에 의하여 할당한 메모리를 해제하지 않음으로써 발생하는 누수이다. 적은 수의 메모리를 할당하는 단순한 프로그램의 경우에는 큰 문제가 발생하지 않지만, 서버나 지속적으로 실행되어야 하는 프로그램의 경우에는 많은 수의 메모리를 할당하기 때문에 올바르게 해제하지 않는다면 메모리 누수로 인하여 시스템이 다운된다. 따라서 본 논문에서는 메모리 누수로 인하여 발생하는 문제점을 해결하기 위한 오류-검출 도구를 구현하였다.

평가 대상이 되는 오류의 종류는 크게 네 가지로 메모리와 알고리즘 공급자, 해쉬 핸들, 키 핸들로 분류하였다. 메모리의 경우에는 CNG가 아닌 일반적인 프로그램에서도 발생하는 문제점이지만, 프로그래밍에서 발생하는 문제의 주된 원인이기 때문에 평가 대상으로 선정하였다. 알고리즘 공급자와 해쉬 핸들, 키 핸들의 경우에는 CNG를 활용하는 프로그래밍에서 발생하는 문제이므로 평가 대상으로 선정하였다. 이러한 평가 대상에 해당하는 함수를 Table 3에 나타내었다.

오류를 검출하기 위하여 평가 대상의 프로그램에 Table 3에 해당하는 함수를 후킹한 후, 생성된 메모리에 대한 주소를 기반으로 링크드 리스트를 구성하여 프로그램이 종료되는 시점에 해제되지 않고 남겨진 메모리를 검사함으로써 오류를 검출하였다. 예를 들어, malloc 함수로 할당된 메모리의 주소가 0x100일 경우 이를 링크드 리스트에 추가하며, free 함수가 호출될 때의 메모리 주소가 0x100일 경우 이를 링크드 리스트에서 삭제하는 방식이다. 따라서 만약 메모리를 할당한 후, 해제하지 않는다면 링크드 리스트에 할당된 주소가 삭제되지 않고 남겨지게 되므로 누수된 메모리에 대한 검출이 가능하다. 알고리즘 공급자의 경우에는 로드하는 함수인 BCryptOpenAlgorithmProvider 함수를 호출한 결과로 반환되는 주소를 링크드 리스트에 추가하며, 언로드하는 함수인 BCryptCloseAlgorithmProvider 함수를 호출할 때의 주소를 링크드 리스트에서 검색하여 삭제한다. 해쉬 핸들의 경우에는 해쉬를 생성하는 함수인 BCryptCreateHash 함수와 해쉬를 복제하는 함수인 BCryptDuplicateHash 함수를 호출한 결과로

Table 3. Function list for the evaluation

Evaluation item		Function name
Memory	Allocation	• malloc • calloc
	Deallocation	• free
Algorithm provider	Loading	• BCryptOpenAlgorithmProvider
	Unloading	• BCryptCloseAlgorithmProvider
Hash handle	Generation	• BCryptCreateHash • BCryptDuplicateHash
	Destruction	• BCryptDestroyHash
Key handle	Generation	• BCryptGenerateSymmetricKey • BCryptGenerateKeyPair • BCryptImportKey • BCryptImportKeyPair
	Destruction	• BCryptDestroyKey

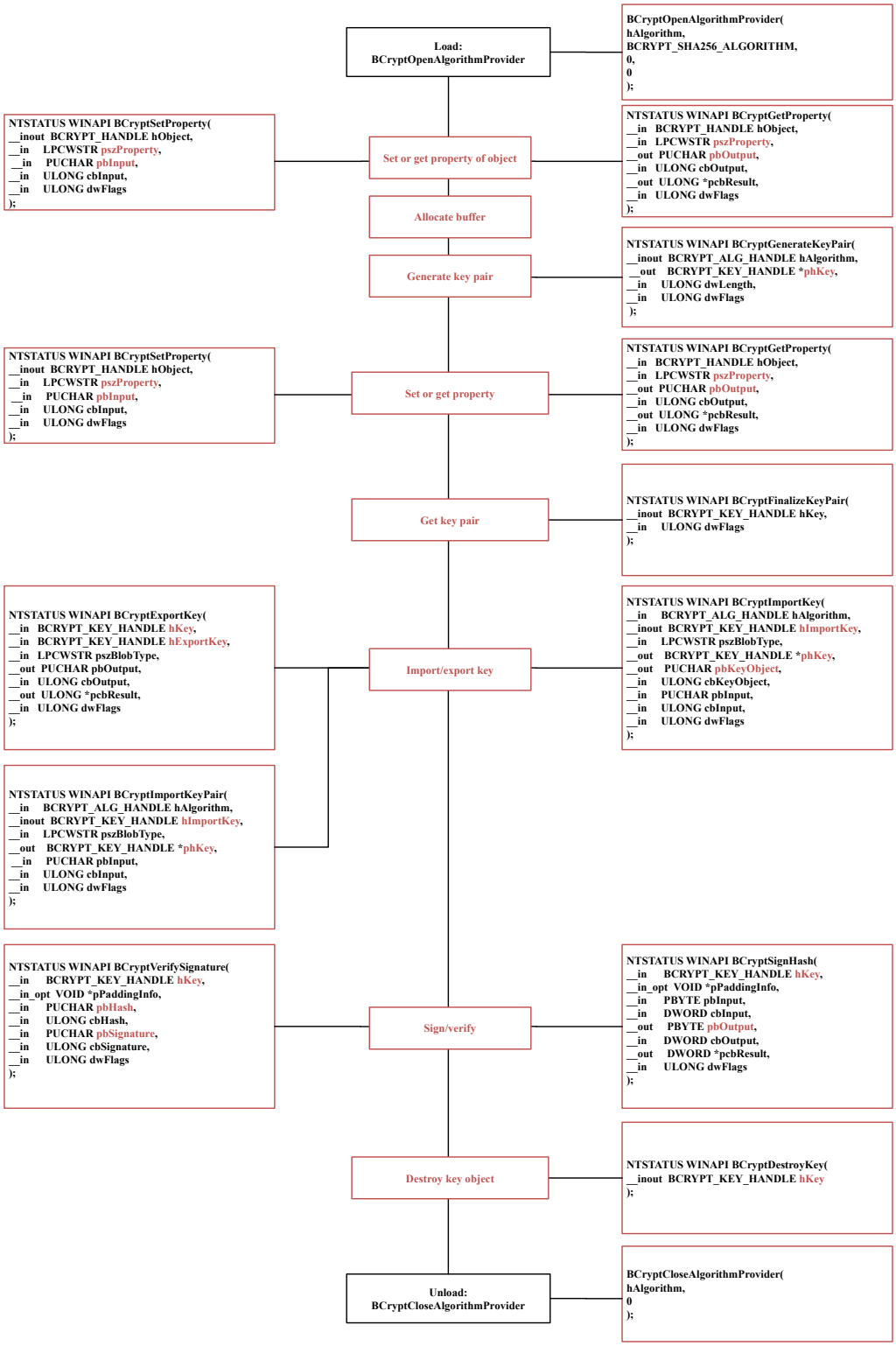

Fig. 7. Process calling the signature and verification function

반환되는 핸들에 대한 주소를 링크드 리스트에 추가하며, 파괴하는 함수인 BCryptDestroyHash 함수를 호출할 때의 주소를 링크드 리스트에서 검색하여 삭제한다. 키 핸들의 경우에는 대칭키를 생성하는 함수인 BCryptGenerateSymmetricKey 함수와 비대칭키를 생성하는 함수인 BCryptGenerateKeyPair 함수, 키를 반입하는 함수인 BCryptImportKey 함수, 키 쌍을 반입하는 함수인 BCryptImportKeyPair 함수가 키 핸들을 생성하기 때문에 호출한 결과로 반환되는 주소를 링크드 리스트에 추가하며, 키를 파괴하는 함수인 BCryptDestroyKey 함수를 호출할 때의 주소를 링크드 리스트에서 검색하여 삭제한다.

오류-검출에 대한 실험을 위하여 해쉬 연산을 수행하는 샘플 프로그램을 작성하였다. 샘플 프로그램은 Fig. 4와 같은 과정을 통하여 해쉬 결과를 생성하며, 오류-검출 과정을 Fig. 8에 나타내었다.

결과를 살펴보면 해쉬 연산을 위한 공급자를 로드한 후, 해쉬 핸들을 생성하여 연산을 수행한다. 연산이 완료되면 로드한 공급자를 언로드하며, 해쉬 핸들을 파괴함으로써 프로그램을 종료한다. 상기 그림은 오류-검출 과정을 개발자가 확인하기 위한 로그에 해당하며, 실제 누수에 대한 평가는 "결과 확인" 버튼을 클릭함으로써 확인이 가능하다. 동일한 샘플 프로그램에 대한 평가 결과를 Fig. 9에 나타내었다.

Fig. 9의 결과를 살펴보면, 해쉬 연산을 위한 공급자를 로드한 후, 정상적으로 언로드하였으므로 누수가 없으며, 연산을 수행하기 위하여 해쉬 핸들을 생성한 후, 정상적으로 파괴하였으므로 누수가 없다. 샘플에서의 해쉬 연산은 키를 필요로 하지 않기 때문에

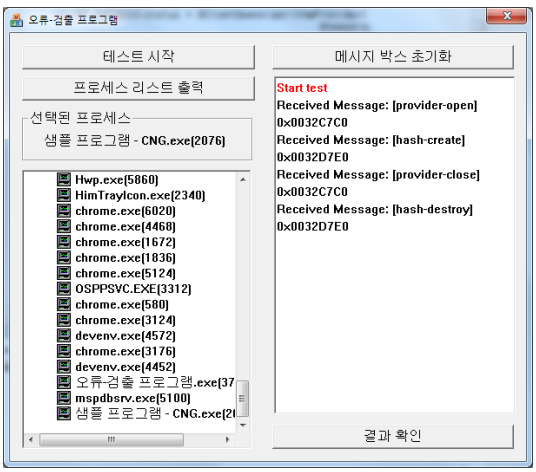

Fig. 8. Process running the error-detection

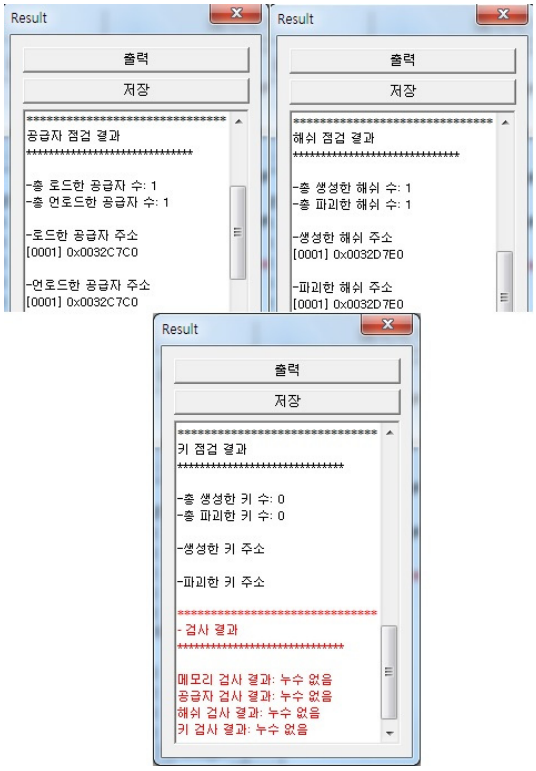

Fig. 9. The first result of error-detection

키 핸들을 생성하고 파괴하지 않아 누수가 없다. 따라서 에러-검출에 대한 종합적인 결과로 샘플 프로그램이 누수가 없다고 도출하였다.

하지만, 만약 키를 생성한 뒤 파괴하지 않았을 경우에는 키 핸들이 누수된 것을 검출하여야 한다. 실험을 위하여 비밀키 쌍을 생성하여 암호화를 수행하는 샘플 프로그램을 작성하였으며, 오류를 발생시키기 위하여 생성한 키 핸들을 해제하지 않도록 구현하였다. 본 논문에서 구현한 도구를 통하여 오류가 존재하는 샘플 프로그램에 대한 실험결과를 Fig. 10에 나타내었다.

결과를 살펴보면, 알고리즘 공급자는 정상적으로 언로드하였지만, 키 핸들을 해제한 로그가 존재하지 않는다. 실제 검사 결과에서도 동일하게 생성한 하나의 키 핸들은 존재하지만, 파괴한 키 핸들이 없는 것으로 나타나며, 키 검사 결과로 총 1개의 핸들이 파괴되지 않은 것으로 도출하였다. 따라서 본 논문에서 구현한 에러-검출 도구가 CNG를 활용하는 프로그램에서 발생하는 오류를 효과적으로 검출하는 것을 확인하였다. 추가적으로 CNG를 활용하는 상용 프로그램인

outlook 프로그램의 에러-검출 결과를 도출하였으며, 이를 Fig. 11에 나타내었다.

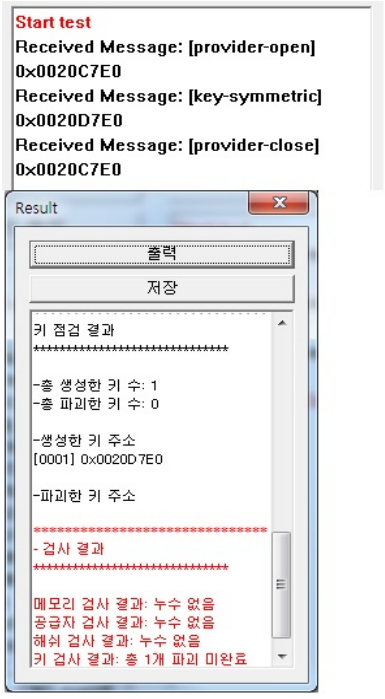

Fig. 10. The second result of error-detection

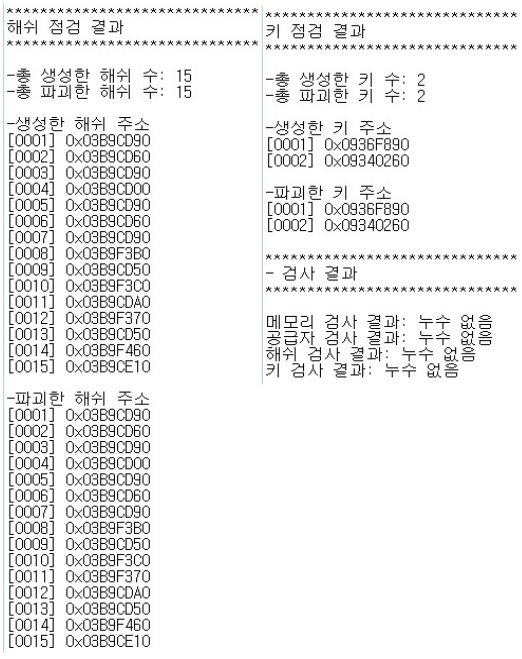

Fig. 11. The third result of error-detection

V. 결 론

본 논문은 암호 라이브러리가 급변하는 환경 변화, 즉, 플랫폼, 운영체제 등의 속도에 따라가지 못하여 발생하는 문제점을 지적하고, 이를 해결할 수 있는 마이크로소프트사의 CNG에 대해 분석하였다. CNG가 등장하게 된 배경, 구조, 그리고 주요 특성들에 대해 서술하였고, 개발자가 직접 코딩하는데 참조할 수 있도록 프로그래밍 기법을 분석하였다. 프로그래밍 기법은 알고리즘 공급자, 난수 생성, 해쉬 함수, 대칭키 암호화, 비대칭키 암호화, 서명 및 검증으로 분류되며, 함수의 호출과정과 흐름도에 대해 기술하였다. 또한, 분석 결과를 기반으로 CNG를 활용하는 프로그램에서 발생 가능한 에러를 검출하기 위한 도구를 구현하였다. 실험 결과, 할당받은 메모리를 정상적으로 해제하지 않은 경우, 에러를 검출함으로써 누수에 의하여 시스템이 다운되는 것을 방지하기 위한 도구로 활용될 것으로 판단된다.

References

[1] Kyungroul Lee, Youngjun Lee, Junyoung Park, Kangbin Yim, and Ilsun You, "Security Issues on the CNG Cryptography Library(Cryptography API: Next Generation)," Proceedings of the Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing(IMIS), pp.709-713, Jul. 2013.

[2] Microsoft, "Using Cryptography," Microsoft Developer Network, http://msdn.microsoft.com/en-us/libraryaa388162(VS.85).aspx

[3] Microsoft, "Cryptography API: Next Generation," Microsoft Developer Network, http://msdn.microsoft.com/en-us/library/aa376210(VS.85).aspx

[4] Nick Wienholt, "Windows Cryptography API: Next Generation(CNG)," Codeguru, 2007, http://www.codeguru.com/columns/kate/article.php/c13813

[5] Quartz, "Next Generation of Cryptography for Microsoft Windows Vista," Codeproj

ect, 2007, http://www.codeproject.com/KB/vista-security/CryptographyNextGenDemo.aspx
〔6〕 NSA, http://www.nsa.gov/ia/programs
〔7〕 /suiteb_cryptography/index.shtml
〔8〕 Microsoft, http://msdn.microsoft.com/
〔9〕 en-us/library/aa375534(v=VS.85).aspx
〔10〕 Microsoft, http://msdn.microsoft.com/
〔11〕 en-us/library/aa376211(v=VS.85).aspx
〔12〕 Jaecheon Byun, Kyungroul Lee, Wansoo Kim, Ilsun You, Kangbin Yim, "Analysis of agility on a new crypto framework CNG," Proceedings of the Fall Conference on the Korean Society for Internet Information, pp.77-78, Oct. 2010.
〔13〕 Wan Soo Kim, Youngtae Choi, Hyeong Jun Yuk, Sitha Pho, Kang Bin Yim, "A Study on the Data Backup Protection through Kernel Data Encryption based on CNG," Proceedings of the Winter Conference on the Korean Institute of Communications and Information Sciences, pp.188-189, Feb. 2012.

〈저자소개〉

이 경 률 (Kyungroul Lee) 정회원
2008년 8월: 순천향대학교 정보보호학과(공학사)
2010년 8월: 순천향대학교 정보보호학과(공학석사)
2015년 2월: 순천향대학교 정보보호학과(공학박사)
2011년 5월~2011년 12월: (미)퍼듀대학교 방문연구원
2015년 6월~현재: 순천향대학교 박사후연구원
〈관심분야〉 취약점 분석, 시스템 보안, 하드웨어 보안, 인터넷 뱅킹, 사용자 인증, 디바이스 인증, 악성코드 분석

유 일 선 (Ilsun You) 종신회원
1995년 2월: 단국대학교 컴퓨터공학과(공학사)
1997년 2월: 단국대학교 컴퓨터공학과(공학석사)
2002년 2월: 단국대학교 컴퓨터공학과(공학박사)
2012년 4월: 일본 큐슈대학교(공학박사)
2005년 3월~2015년 8월: 한국성서대학교 컴퓨터소프트웨어학과 교수
2010년 6월~현재: JoWUA 저널 편집장(Scopus)
2014년 3월~현재: 영국 공학기술학회 석학회원(IET Fellow)
2014년 3월~현재: 국제전기전자학회 준석학회원(IEEE Senior Member)
2015년 1월~현재: 한국정보보호학회 협동이사
2015년 9월~현재: 순천향대학교 정보보호학과 교수
〈관심분야〉 모바일 인터넷 보안, 인증 및 접근통제, 정형화된 보안검증

임 강 빈 (Kangbin Yim) 종신회원
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
2010년 12월~2012년 2월: (미)퍼듀대학교 객원교수
〈관심분야〉 취약점분석, 악성코드분석, 보안관제시스템, 제어시스템보안, 보안하드웨어구조, 인증프로토콜