Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.25, NO.6, Dec. 2015 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2015.25.6.1327

마이크로소프트 차세대 암호 라이브러리의 확장성 분석: 국산 암호화 알고리즘 HAS-160 연동 구현사례를 중심으로*

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

An Analysis of Agility of the Cryptography API Next Generation in Microsoft: Based on Implementation Example of Applying Cryptography Algorithm HAS-160 in South Korea*

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 대체하는 CNG의 구조와 특징, 프로그래밍 기법에 대해 알아보고, 국산 암호 알고리즘 중 HAS-160을 지원하기 위한 해쉬 공급자를 구현하였으며, CNG의 확장성에 대하여 구현결과를 기반으로 다양한 각도에서 분석하고, 이에 대한 커스터마이징 전략에 대해 제안하였다. 구현된 HAS-160 해쉬 공급자를 비롯해 CNG에 대한 기본적인 분석결과는 국산 암호 알고리즘의 비스타 환경에 적용하기 위한 방안으로 활용될 것으로 예상되며, CNG가 아직 갖추지 못한 안전한 배포 방안에 대해 향후 연구할 예정이다.

ABSTRACT

This paper surveys structures, features and programming techniques of CNG that is substitution of CAPI in Microsoft, and implements hash provider for support HAS-160 that is one of the Korean hash algorithm. After that, we analysis agility from different perspective based on implemented results, and propose customizing stratagem. Analyzed results of basic concepts and implemented HAS-160 hash provider are expected applying measure for Korean cryptography algorithm in Vista environment. Consequently, we will research secure distribution way due to it is not apply on CNG.

Keywords: CAPI, CNG, CSP, HAS-160

I. 서 론

마이크로소프트는 윈도우즈 비스타 출시와 함께 기존의 암호 라이브러리인 Cryptography API (CAPI)를 대체하는 Cryptography: Next Generation(CNG)을 제안하였다[1][2][3][4]. CNG는 최신 암호 알고리즘 및 커널모드, 감사기능 등을 지원하고 새로운 요구와 환경변화에 적절히 대응할 수 있는 민첩성(Agility)을 제공함으로써 차세대 암호 라이브러리로 자리매김하였다[5]. 우리나라는 국

Received(09. 03. 2015), Modified(10. 27. 2015), Accepted(11. 29. 2015)
* 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)

산 암호 알고리즘의 사용이 필수이기 때문에 CAPI 기반 암호 서비스를 제공하는 마이크로소프트 윈도우즈와 관련된 제품들이 국내 보안 환경(특히, 공개키 기반 환경)과 연동되지 않는다는 문제점이 있어[7], CNG의 등장은 호환성 측면에 있어 크게 주목할 만한 점이다. 예를 들어, 국외에서는 Secure Socket Layer(SSL)를 통하여 인터넷뱅킹 서비스를 제공하지만[8], 우리나라에서는 국내 암호 알고리즘 SEED의 사용이 필수적이기 때문에 SSL을 사용할 수 없으므로 액티브 X 기반의 플러그인이 필요하다. 윈도우즈 비스타 이후로는 CNG 제공자 구조를 기반으로 앞서 언급된 문제들이 해결될 것으로 기대되며, 이를 위한 첫 번째 시도로서 (구)한국정보보호진흥원이 128비트 국산 암호 알고리즘 SEED의 CNG 제공자를 개발하였다[5]. 그러나 비스타 환경과 폭넓은 국산 암호 알고리즘 연동을 위해서는 SEED와 더불어 HAS-160, ARIA, KCDSA 등과 같은 알고리즘을 지원하는 CNG 공급자의 구현이 필요하며, 이를 위한 체계적인 지침과 활용방안의 제시가 요구된다. 따라서 본 논문에서는 플러그인 모델 기반의 CNG 공급자 구조와 CNG의 확장성에 대하여 분석하여 그 특징을 도출하고, 국산 암호 알고리즘 중 HAS-160의 암호 알고리즘 공급자 구현을 통하여 CNG의 확장과정을 단계별로 분석하며, 그 결과를 바탕으로 CNG 제공자 구현을 위한 체계적인 지침과 활용방안을 제시한다. 마지막으로 분석결과를 토대로 CNG 라이브러리의 커스터마이징 전략을 제안하고자 한다.

본 논문의 구성은 다음과 같다. 제2장은 기존에 연구된 CNG에 대해 서술하고, 제3장은 CNG를 위한 국산 암호 알고리즘 공급자 구현, 제4장은 CNG의 확장성 분석과 커스터마이징 전략, 제5장에서 결론을 도출한다.

II. 관련 연구

2.1 CNG의 구조 및 특징

기존에 존재하던 암호 알고리즘의 안전성 문제와 더불어 새로이 연구되고 개발되는 암호 알고리즘으로 인하여 확장성에서 상대적으로 단점을 지닌 마이크로소프트 CAPI의 문제점이 드러나면서 CAPI를 대체할 수 있는 새로운 구조의 암호 라이브러리가 요구되었다. 암호 라이브러리는 로컬에서 실행되는 단일 모듈이나 서버-클라이언트 기반의 구조 등에서 사용되고 있으며, 최근 여러 형태의 소규모 장비, 예를 들면, 하드웨어 보안 토큰(HSM)이나 스마트 폰 등의 장치에서 사용되는 것으로 볼 때, 다양한 알고리즘과 응용에서의 요구를 빠르게 대처할 수 있는 능력을 뜻하는 확장성과 민첩성이 보다 강조되고 있는 추세이다. 따라서 마이크로소프트는 기존 암호 알고리즘인 CAPI의 상기 문제점을 보완하고자 확장성과 민첩성에 용이한 구조를 가진 CNG를 발표하였으며, CNG는 CAPI가 제공하는 모든 암호 알고리즘을 제공하는 동시에 새로이 연구되고 개발된 알고리즘을 추가로 제공하기 때문에 보다 통합되고 복합적으로 구성될 수 있으므로 개발자가 이를 이용하여 다양한 동작을 만들거나 수행방법에 대해 스스로 관여할 수 있도록 보다 유연하게 설계되었다. Fig. 1.에 CNG의 전체적인 구조를 나타내었다[6].

CNG는 크게 BCrypt와 NCrypt로 나누어지며, BCrypt가 핵심적인 역할을 한다. BCrypt는 난수 생성, 해쉬, 서명, 키 등과 관련된 암호 프리미티브를 제공하고, NCrypt는 비대칭키 및 스마트카드 등과 같은 하드웨어를 지원하기 위한 키 저장 기능 등을 제공한다. CNG의 큰 장점 중에 하나는 사용자 모드와 커널 모드를 동시에 지원한다는 점이며, BCrypt는 두 모드 모두 지원하고 있지만, NCrypt의 키 보관 기능은 사용자 모드에서만 가능하다. CNG는 암호 민첩성, 연방표준 준수, Suite B 지원, 커널모드 지원, 감사 기능의 6가지 특성을 가진다.

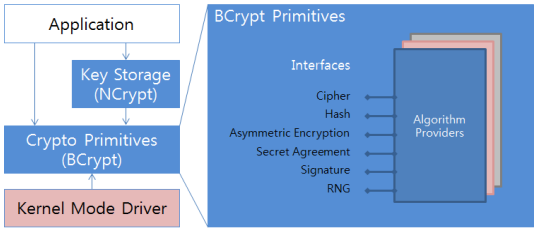

Fig. 1, The whole structure of CNG

2.2 CNG의 프로그래밍 기법

CNG API를 사용하여 프로그래밍하기 위해서는 전형적인 단계가 필요하며, 이를 Table 1.에 나타내었다.

CNG 프로그래밍 기법으로는 알고리즘 공급자, 난수 생성, 해쉬 함수, 대칭 및 비대칭키 암호, 서명

Table 1. Typical steps of the CNG programming

Step	Calling function
Open algorithm provider	BCyptOpenAlgorithmProvider
Get or set algorithm property	· Set: BCryptSetProperty · Get: BCryptGetProperty
Generate, import/export, and destroy key	· Generate: BCryptGenerateSymmetricKey, BCryptGenerateKeyPair · Import/export: BCryptImportKey, BCryptImportKeyPair, BCryptExportKey · Destroy : BCryptDestroyKey, BCryptFinalizeKeyPair
Cryptographic operations	· Generate random number: BCryptGenRandom · Hash operation: BCryptCreateHash, BCryptHashData, BCryptDuplicateHash, BCryptDestroyHash · Symmetric key operation: BCryptEncrypt, BCryptDecrypt · Asymmetric key operation: BCryptFinalizeKeyPair, BCryptEncrypt, BCryptDecrypt · Signature and verification: BCryptSignHash, BCryptVerifySignature
Close algorithm provider	BCryptCloseAlgorithmProvider

및 검증 등이 있으며[6], 본 논문은 HAS-160에 대한 구현이 핵심이므로 구현 시 반드시 필요한 알고리즘 공급자와 해쉬 함수의 프로그래밍 기법에 대하여 상세히 다루었다.

· 알고리즘 공급자

알고리즘 공급자는 CNG의 가장 기본적이면서도 중요한 요소이며, BCrypt 내에 구성되어 있다. 알고리즘 공급자의 함수 호출과정은 Fig. 2.와 같다.

Step 1. 알고리즘 공급자 로드 : BCryptOpenAlgorithmProvider 함수로 수행하며, 호출 결과 phAlgorithm 포인터에 핸들을 반환한다.

Step 2. 알고리즘 공급자 언로드 : BCryptCloseAlgorithmProvider 함수로 수행되며, 입력받은 phAlgorithm 핸들을 해제한다.

· 해쉬 함수

해쉬 함수는 보안적인 성향을 가지는 역할을 하는 기능 중 하나이며, 다양한 범위에서 활용되고 있다. 해쉬 함수 호출과정은 Fig. 3.과 같다.

Step 1. 알고리즘 공급자 로드 : BCryptOpenAlgorithmProvider 함수로 수행하며, 호출 결과 phAlgorithm 포인터에 핸들을 반환한다.

Step 2. 오브젝트 속성 설정 및 추출 : BCryptGetProperty, BCryptSetProperty 함수로 수행하며, 해쉬를 위한 버퍼 크기를 추출하여 메모리를 할당한다.

Step 3. 해쉬 생성 : BCryptCreateHash 함수로 수행하며, 해쉬연산을 수행하기 위한 핸들을 반환한다.

Step 4. 해쉬 연산 및 복제 : BCryptHashData 함수로 수행하며, 할당한 버퍼에 단방향 해쉬 연산 결과가 저장된다. BCryptHashData 함수를 여러 번 반복하여 호출할 경우 단일 체인 형태로 해쉬 값

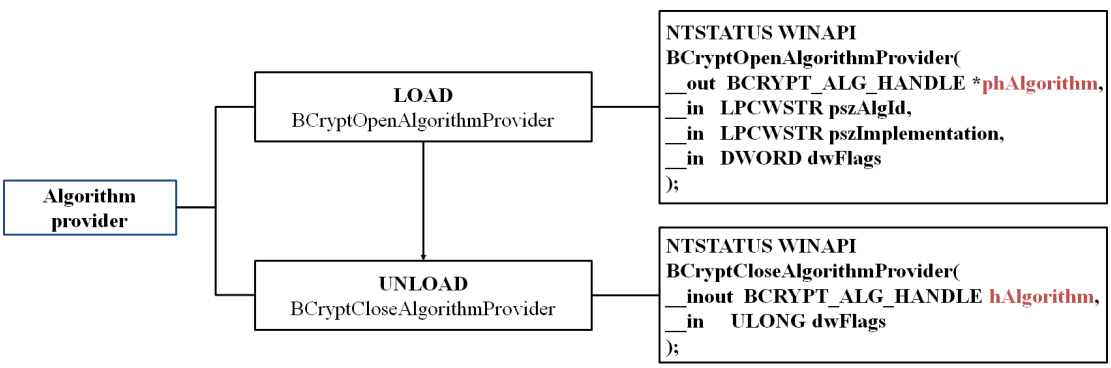

Fig. 2. Process calling the algorithm provider function

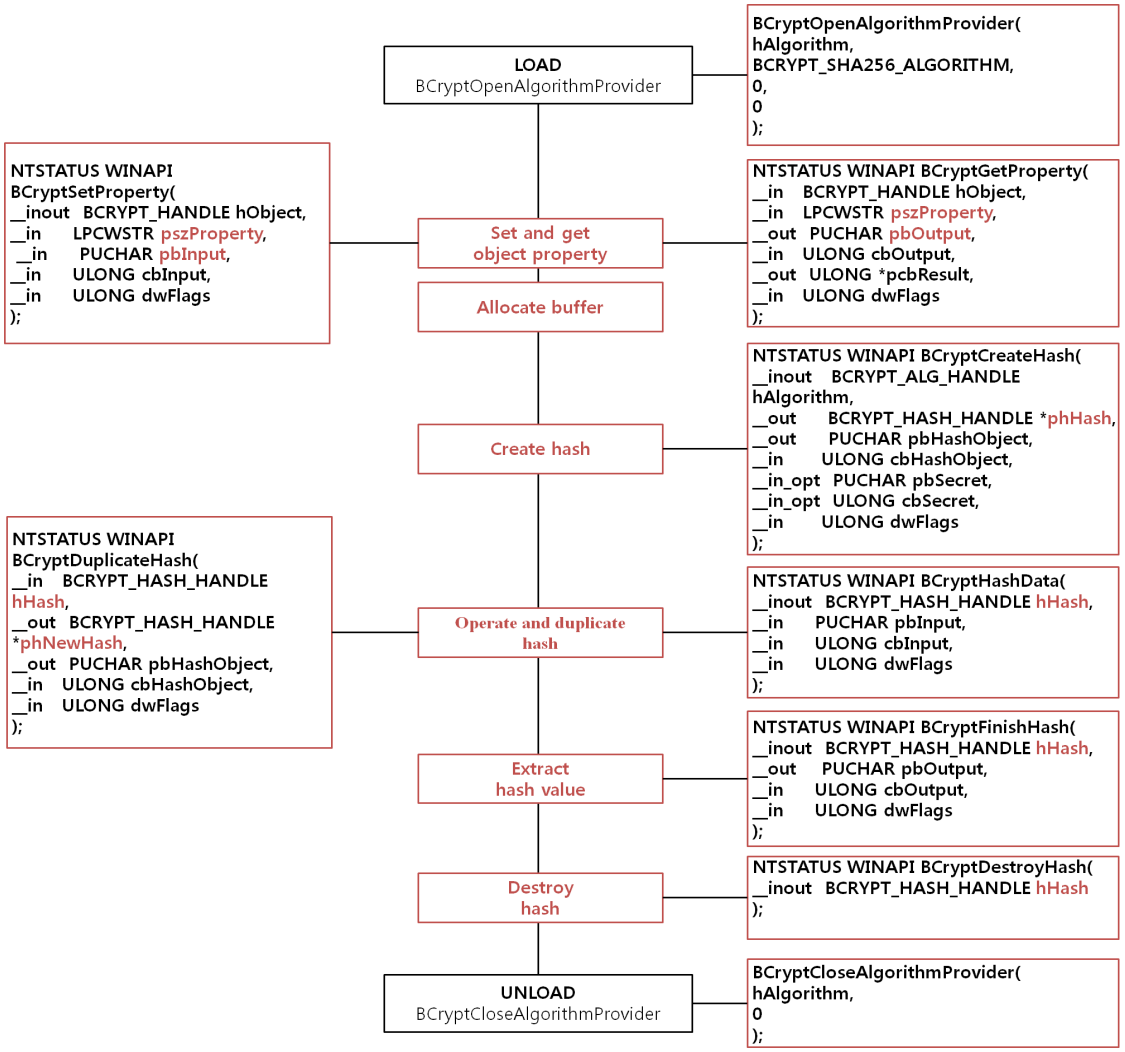

Fig. 3. Process calling the hash function

이 생성되며, BCryptDuplicateHash 함수를 활용하면 공통적인 해쉬 연산 결과를 기반으로 서로 다른 여러 개의 결과를 생성할 수 있다.

Step 5. 해쉬 값 추출 : BCryptFinishHash 함수로 수행하며, 해쉬 연산 결과를 반환한다. 해쉬 연산 결과의 크기는 사용된 해쉬 알고리즘에 종속적이므로 BCryptGetProperty 함수를 호출하여 그 크기를 얻은 후 해쉬 결과를 추출한다.

Step 6. 해쉬 종료 : BCryptDestroyHash 함수로 수행하며, 할당한 오브젝트와 버퍼 등을 해제한다.

Step 7. 알고리즘 공급자 언로드 : BCryptCloseAlgorithmProvider 함수로 수행한다.

III. CNG를 위한 국산 암호 알고리즘 제공자 구현

CNG는 새로운 암호 알고리즘을 추가하거나 기존의 표준 암호 알고리즘의 성능을 향상시킬 필요성이 있을 경우, 이와 더불어 보안 관점에서 향상된 새로운 버전으로 갱신할 수 있도록 개발자 친화적인 플러그 인 모델 기반의 CNG 공급자를 지원한다. 즉, 사용자는 주어진 요구사항에 따라 새로운 CNG 공급자를 개발하고, 현재의 CNG에 장착함으로써 CNG의 성능과 보안성을 주어진 환경에 맞게 강화할 수 있다. CNG 공급자는 암호 알고리즘 공급자와 키 관리 공급자로 분류할 수 있다. 암호 알고리즘 공급자는 하

나 이상의 암호 알고리즘 또는 난수생성 기능을 지원할 수 있고, 키 저장 공급자는 암호화에 사용되는 키를 저장하기 위한 기능을 제공한다. 새롭게 구현된 사용자 알고리즘은 공급자 등록 함수를 사용하여 CNG 라우터에 추가한 후 사용 가능하며, 필요 없을 경우 등록 해제 함수를 사용하여 제거한다.

3.1 암호 알고리즘 공급자 구현

암호 알고리즘 공급자는 알고리즘 종류에 따라 암호 공급자(Cipher Provider), 해쉬 공급자(Hash Provider), 공개키 암호 공급자(Asymmetric Encryption Provider), 키 교환 공급자(Secret Agreement Provider), 서명 공급자(Signature Provider), 난수 생성기 공급자(Random Number Generator Provider)로 분류된다. 각 공급자를 위해서 고유의 동작을 수행하는 콜백함수들을 구현하여야 하고, 아울러 공급자명과 알고리즘 아이디, 그리고 콜백함수들의 함수포인터 테이블을 제공하는 인터페이스함수를 제공하여야 한다. 각 알고리즘별로 구현하여야 하는 인터페이스함수와 콜백함수들은 Table 2. - 7.과 같다.

Table 2. List of callback and interface functions for cipher provider

Interface function	GetCipherInterface
Callback function	· BCryptOpenAlgorithmProviderFn · BCryptGetPropertyFn · BCryptSetPropertyFn · BCryptCloseAlgorithmProviderFn · BCryptGenerateSymmetricKeyFn · BCryptEncryptFn · BCryptDecryptFn · BCryptImportKeyFn · BCryptExportKeyFn · BCryptDuplicateKeyFn · BCryptDestroyKeyFn

Table 3. List of callback and interface functions for hash provider

Interface function	GetHashInterface
Callback function	· BCryptOpenAlgorithmProviderFn · BCryptGetPropertyFn · BCryptSetPropertyFn · BCryptCloseAlgorithmProviderFn · BCryptCreateHashFn · BCryptHashDataFn · BCryptFinishHashFn · BCryptDuplicateHashFn · BCryptDestroyHashFn

Table 4. List of callback and interface functions for public key provider

Interface function	GetAsymmetricEncryptionInterface
Callback function	· BCryptOpenAlgorithmProviderFn · BCryptGetPropertyFn · BCryptSetPropertyFn · BCryptCloseAlgorithmProviderFn · BCryptGenerateKeyPairFn · BCryptFinalizeKeyPairFn · BCryptEncryptFn · BCryptDecryptFn · BCryptImportKeyPairFn · BCryptExportKeyFn · BCryptDestroyKeyFn · BCryptSignHashFn · BCryptVerifySignatureFn

Table 5. List of callback and interface functions for secret agreement provider

Interface function	GetSecretAgreementInterface
Callback function	· BCryptOpenAlgorithmProviderFn · BCryptGetPropertyFn · BCryptSetPropertyFn · BCryptCloseAlgorithmProviderFn · BCryptSecretAgreementFn · BCryptDeriveKeyFn · BCryptDestroySecretFn · BCryptGenerateKeyPairFn · BCryptFinalizeKeyPairFn · BCryptImportKeyPairFn · BCryptExportKeyFn · BCryptDestroyKeyFn

Table 6. List of callback and interface functions for signature provider

Interface function	GetSignatureInterface
Callback function	· BCryptOpenAlgorithmProviderFn · BCryptGetPropertyFn · BCryptSetPropertyFn · BCryptCloseAlgorithmProviderFn · BCryptGenerateKeyPairFn · BCryptFinalizeKeyPairFn · BCryptSignHashFn · BCryptVerifySignatureFn · BCryptImportKeyPairFn · BCryptExportKeyFn · BCryptDestroyKeyFn

Table 7. List of callback and interface functions for random number generator provider

Interface function	GetRngInterface
Callback function	· BCryptOpenAlgorithmProviderFn · BCryptGetPropertyFn · BCryptSetPropertyFn · BCryptCloseAlgorithmProviderFn · BCryptGenRandomFn

3.2 암호 알고리즘 공급자 등록

사용자가 구현한 CNG 공급자는 다음 단계를 거쳐 CNG 라우터에 추가된다.

· 공급자 모듈 등록

Step 1. BCryptRegisterProvider 함수를 호출하여 사용자 공급자를 CNG 라우터에 추가한다.

Step 2. BCryptAddContextFunctionProvider 함수를 호출하여 공급자가 제공하는 암호화 함수에 대한 리스트를 등록한다.

· 공급자 모듈 등록 해제

CNG 라우터에 추가된 사용자가 구현한 공급자는 다음의 단계를 수행하여 등록을 해제한다. 이때 등록된 공급자 전체가 해제되며 공급자의 일부분만을 변경할 수는 없다. 그러므로 등록된 공급자의 일부분을 수정할 경우에도 전체 공급자를 등록 해제한 후 일부분이 수정된 공급자를 다시 등록하여야 한다.

Step 1. BCryptRemoveContextFunctionProvider 함수를 호출하여 함수 구성 등록과 관련된 정보를 제거한다.

Step 2. BCryptUnregisterProvider 함수를 호출하여 CNG 라우터에서 공급자 등록을 해제한다.

```
CRYPT_IMAGE_REG UserModeProviderImage = {
                L"symmprov.dll",
                1,              // Number of algorithm classes the binary supports
                AllUserModeAlgorithmClasses };
CRYPT_PROVIDER_REG SymmetricCipherProvider = {
                0,
                NULL,
                &UserModeProviderImage,         // image that provide user-mode
                NULL };                         // image that provide kernel-mode
ntStatus = BCryptRegisterProvider(
                SYMM_CIPHER_PROVIDER_NAME,
                0,                      // Flags: fail if provider is already registered
                &SymmetricCipherProvider
        );
```

Fig. 4. Registration example of user provider

```
ntStatus = BCryptAddContextFunctionProvider(
                CRYPT_LOCAL,                    // Scope: local machine only
                NULL,                           // Application context: default
                BCRYPT_CIPHER_INTERFACE,        // Algorithm class
                SYMM_CIPHER_ALGID,              // Algorithm name
                SYMM_CIPHER_PROVIDER_NAME,      // Provider name
                CRYPT_PRIORITY_BOTTOM           // Lowest priority
        );
```

Fig. 5. Registration example of function information of user provider

```
ntStatus = BCryptRemoveContextFunctionProvider(
                CRYPT_LOCAL,                    // Scope: local machine only
                NULL,                           // Application context: default
                BCRYPT_CIPHER_INTERFACE,        // Algorithm class
                SYMM_CIPHER_ALGID,              // Algorithm name
                SYMM_CIPHER_PROVIDER_NAME       // Provider
        );
```

Fig. 6. Removal example of function information of user provider

```
ntStatus = BCryptRemoveContextFunction(
                CRYPT_LOCAL,                    // Scope: local machine only
                NULL,                           // Application context: default
                BCRYPT_CIPHER_INTERFACE,        // Algorithm class
                SYMM_CIPHER_ALGID               // Algorithm name
        );
```

Fig. 7. Removal example of user provider

3.3 HAS-160을 지원하는 사용자 해쉬 공급자 구현

본 논문은 국산 알고리즘 HAS-160을 지원하는 해쉬 공급자를 구현하기 위하여 GetHashInterface 함수와 BCRYPT_HASH_FUNCTION_TABLE 구조체, 그리고 구조체 내 포함되는 콜백함수들을 모두 구현하였다. CNG 클라이언트가 암호 공급자를 요청한 후, BCRYPT_HASH_FUNCTION_TABLE 구조체를 수신하는 과정은 Fig. 8.과 같다.

Step 1. CNG Client는 CNG Provider에게 암호 공급자를 요청한다.

Step 2. CNG Provider는 CNG 라우터/로드를 통하여 LoadLibrary 함수를 호출하여 DLL을 로드한다.

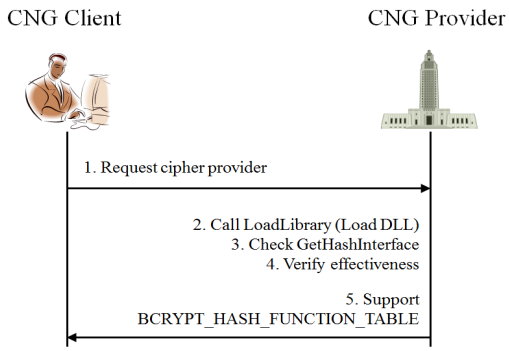

Fig. 8. Supply process of cipher provider

```
NTSTATUS WINAPI GetHashInterface(
                    IN LPWSTR pszProviderName,
                    IN LPWSTR pszAlgId,
                    OUT BCRYPT_HASH_FUNCTION_TABLE **ppFunctionTable,
                    IN DWORD dwFlags)
{
    NTSTATUS Status = STATUS_NOT_SUPPORTED;

    UNREFERENCED_PARAMETER(pszProviderName);
    UNREFERENCED_PARAMETER(dwFlags);

    if(wcscmp(L"HAS160", pszAlgId)) {
        Status = STATUS_NOT_SUPPORTED;
        goto Cleanup;
    }

    *ppFunctionTable = &Has160HashFunctionTable;
    Status = STATUS_SUCCESS;
Cleanup:
    return Status;
}
```

Fig. 9. Implementation of get hash interface

Step 3. DLL 로드가 성공적이면 GetHashInterface가 존재하는지 확인한다. 이는 미리 이름이 정해져 있어야만 확인이 가능하다.

Step 4. 검색된 알고리즘이 유효한 알고리즘 아이디를 가지는지 검사한다(유효성 검사).

Step 5. 검사 결과가 성공적이라면 CNG Provider는 BCRYPT_HASH_FUNCTION_TABLE 구조체를 CNG Client에게 제공한다.

구현한 GetHashInterface와 BCRYPT_HASH_FUNCTION_TABLE 구조체를 Fig. 9., 10.에 나타내었으며, 기타 콜백함수를 Table 8.에 나타내었다.

```
BCRYPT_HASH_FUNCTION_TABLE Has160HashFunctionTable =
{
    BCRYPT_HASH_INTERFACE_VERSION_1,
    Has160OpenHashProvider,
    Has160GetHashProperty,
    Has160SetHashProperty,
    Has160CloseHashProvider,
    Has160CreateHash,
    Has160HashData,
    Has160FinishHash,
    Has160DuplicateHash,
    Has160DestroyHash,
};
```

Fig. 10. Define BCRYPT_HASH_FUNCTION_TABLE

Table 8. List of implemented HAS-160 callback functions

Name of callback function	Name of implemented function
BCryptOpenAlgorithm ProviderFn	Has160OpenHashProvider
BCryptGetPropertyFn	Has160GetHashProperty
BCryptSetPropertyFn	Has160SetHashProperty
BCryptCloseAlgorithm ProviderFn	Has160CloseHashProvider
BCryptCreateHashFn	Has160CreateHash
BCryptHashDataFn	Has160HashData
BCryptFinishHashFn	Has160FinishHash
BCryptDuplicateHashFn	Has160DuplicateHash
BCryptDestroyHashFn	Has160DestroyHash

3.4 HAS-160 해쉬 공급자 등록 및 사용

구현된 HAS-160 해쉬 공급자는 공급자 이름을 "HAS160 Hash Provider", 알고리즘 아이디를 "HAS160"로 설정하여 시스템에 등록하였으며, 등록 결과를 Fig. 11., 12.에 나타내었다.

등록된 HAS-160 해쉬 공급자를 테스트하기 위한 클라이언트 프로그램을 구현하였으며, 테스트를 위한 CNG 클라이언트 프로그램의 함수 호출과정은 Fig. 13.과 같다.

Fig. 14.는 입력 스트림 값을 "123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890"로 설정하여 프로그램을 수행한 결과이다.

```
FunctionsArray[0] = L"HAS160"
InterfaceReg.dwInterface = BCRYPT_HASH_INTERFACE;
InterfaceReg.dwFlags = 0;
InterfaceReg.cFunctions = 1;
InterfaceReg.rgpszFunctions = FunctionsArray;

ImageReg.pszImage = L"has160pv.dll";        //provider dll
ImageReg.cInterfaces = 1  ;                 //only one interface - hash
ImageReg.rgpInterfaces = &pInterfaceReg;

ProvReg.pUM = &ImageReg;
 If ( !NT_SUCCESS(status = BCryptRegisterProvider(
                                        L"HAS160 Hash Provider",
                                        0,
                                        &ProvReg)) ) {
        wprintf(L"Error 0x%x returned by BCryptRegisterProvider\n", status);
        goto Cleanup;
 }
if( !NT_SUCCESS(status = BCryptAddContextFunctionProvider(
                                        CRYPT_LOCAL,
                                        NULL,
                                        BCRYPT_HASH_INTERFACE,
                                        L"HAS160",
                                        L"HAS160 Hash Provider",
                                        CRYPT_PRIORITY_TOP)) ) {
        wprintf(L"Error 0x%x returned by BCryptAddContextFunctionProvider\n", status);
        goto Cleanup;
 }
```

Fig. 11. Registration of user hash provider

```
C:₩Has160Provider>has160_config -enum
Microsoft Primitive Provider
Microsoft Smart Card Key Storage Provider
Microsoft Software Key Storage Provider
Microsoft SSL Protocol Provider

C:₩Has160Provider>has160_config -register

C:₩Has160Provider>has160_config -enum
Microsoft Primitive Provider
Microsoft Smart Card Key Storage Provider
Microsoft Software Key Storage Provider
Microsoft SSL Protocol Provider
HAS160 Hash Provider

C:₩Has160Provider>_
```

Fig. 12. Registered result of HAS-160 hash provider

```
void __cdecl wmain(int argc, __in_ecount(argc) LPWSTR *wargv)
{
      STEP 1: OpenAlgorithmProvider()
      STEP 2: GetAlgorithmProperties()
      STEP 3: CreateHash()
      STEP 4: HashData()
      STEP 5: FinishHash()
      STEP 6: CloseAlgorithmProvider()
      STEP 7: DestroyHash()
}
```

Fig. 13. Example of use for CNG provider

```
C:₩Has160Provider>has160_client
   DV = 07 f0 5c 8c 07 73 c5 5c a3 a5
        a6 95 ce 6a ca 4c 43 89 11 b5
C:₩Has160Provider>
```

Fig. 14. Execution result of client program for HAS-160 hash provider

IV. CNG의 확장성 분석 및 커스터마이징 전략 연구

본 장에서는 CNG의 확장성을 개발 비용과 확장 용이성, 보안성 측면에서 분석한다. 이를 바탕으로 기업과 공공기관 등에서 주어진 환경과 보안 요구사항에 적합하도록 CNG 커스터마이징 전략을 제안한다.

4.1 CNG 확장성 분석

4.1.1 개발 비용

CNG의 확장에 대한 개발비용은 암호 알고리즘 공급자와 키 관리 공급자를 구현하는 비용으로 측정하였으며 이를 Table 9.에 나타내었다. 여기서 개발 비용은 구현에 필요한 함수의 수와 알고리즘 고유의 기능으로 측정하였다. CNG는 독립적인 모듈단위로 잘 설계되어 있기 때문에 각 공급자별로 요구되는 고유의 기능만을 구현하면 다른 알고리즘과 유기적인 연동이 가능하다. 예를 들어 HAS-160을 위한 해쉬 공급자를 추가할 경우, CNG에서 HAS-160 알고리

Table 9. Development costs for CNG extension

Algorithm provider		Number of implemented functions	Implemented unique features
Cipher algorithm provider	Cipher provider	12	- Generate symmetric key - Encrypt/decrypt based on symmetric key - Import/export symmetric key
	Hash provider	10	- Generate hash
	Asymmetric key cipher provider	14	- Generate asymmetric key pair - Encrypt/decrypt based on public key - Signature/verification based on public key - Import/export public key
	Key Exchange Provider	13	- Generate public key - Exchange key - Import/export public key
	Signature provider	12	- Generate public key - Signature/verification based on public key - Import/export public key
	Random number generator provider	6	- Generate random number
Key management provider		27	- Generate key - Manage key - Encrypt/decrypt - Signature/verification - key exchange

증은 다른 공개키 암호 알고리즘과 함께 전자서명을 위해 사용될 수 있다.

CAPI의 경우, 새로운 암호 알고리즘 혹은 키 저장 구조를 추가하거나 기존의 암호 알고리즘을 개선하고자 할 때 하나의 CSP를 개발하여야 한다. 즉, CAPI는 CNG와 달리 기능단위의 독립적인 모듈 구조를 지원하지 않기 때문에 CSP 개발 시 다른 알고리즘의 연관관계를 고려하여야 하는 문제점을 갖는다. 예를 들어 CPDeriveKey, CPExportKey, CPImportKey, CPSignHash 그리고 CPVerifySignature와 같은 함수들의 경우, 여러 알고리즘들이 복합적으로 사용되어야 하기 때문에 하나의 암호 알고리즘을 추가하기 위하여 관련 알고리즘들을 모두 구현하여야 하는 경우가 발생한다. 이로 인하여 CSP는 자체적으로 중요한 암호 알고리즘들을 모두 지원하도록 개발되는 것이 일반적이었다.

한편, CNG와 같이 CSP에서도 필요한 콜백함수를 개발하고 나머지는 지원하지 않을 수 있다. 그러나 이러한 경우에도 응용 개발자 입장에서는 여러 개의 CSP를 동시에 오픈하여 프로그램 하여야 하기 때문에 프로그램의 일관성이 떨어지고 복잡도가 높아지는 문제점이 있다. Table 10.은 CSP를 개발하기 위해 구현하여야 하는 콜백함수들과 개발비용을 보여주며, Table 11.은 HAS-160 구현을 위한 CNG와 CSP의 개발비용을 비교하였다. 여기서 CSP의 경우, 필요한 부분만 구현하는 방법을 채택한다고 가

Table 10. Development cost of CSP

Functions	Cost
CPAcquireContext CPReleaseContext CPGenKey CPDeriveKey* CPDestroyKey CPSetKeyParam CPGetKeyParam CPGetProvParam CPSetProvParam CPSetHashParam CPGetHashParam CPExportKey* CPImportKey* CPEncrypt CPDecrypt CPCreateHash CPHashData CPHashSessionKey CPSignHash* CPDestroyHash CPVerifySignature* CPGenRandom CPGetUserKey CPDuplicateHash CPDuplicateKey	1) Number of implemented function: 25 2) Implemented unique features: - Various asymmetric key cipher algorithms (Key generation, encryption/decryption, signature) - Various symmetric key cipher algorithms (Key generation, encryption/decryption) - Various hash algorithms (Hash operation) - Key management (Key import/export, key generation/destroy/query)

Table 11. Comparison of the development cost of CSP and CNG: Through the HAS-160 implementation

비교항목	CNG	CSP
Implement callback functions	GetHashInterface BCryptOpenAlgorithmProviderFn BCryptGetPropertyFn BCryptSetPropertyFn BCryptCloseAlgorithmProviderFn BCryptCreateHashFn BCryptHashDataFn BCryptFinishHashFn BCryptDuplicateHashFn BCryptDestroyHashFn	CPAcquireContext CPReleaseContext CPGetProvParam CPSetProvParam CPSetHashParam CPGetHashParam CPCreateHash CPHashData CPHashSessionKey CPDuplicateHash
Linkage problem with public key cryptography	HAS-160 is able to apply without difficulty for the signature and the verification in order to flexible linkage with the existing public key cryptography.	The asymmetric cipher algorithm has to implement separately for CPSignHash and CPVerifySignature because it is impossible to flexible linkage with the existing CSP. Alternatively, the asymmetric cipher operation of the existing CSP is able to utilize. However, in this case, multiple CSPs is used at the same time in the application program. Furthermore, there is a problem about consistency and complexity of program. Because CPSignHash and CPVerifySignature cannot use directly in the application program so hash or encryption or decryption operation has to programming separately.

정하였다. CNG와 CSP는 단순히 함수기능의 구현만을 고려하였을 때 거의 개발비용이 동일함을 확인할 수 있으나, CSP 개발 시 다른 알고리즘의 연관관계를 고려하여야 하는 문제점으로 인하여 기존의 공개키 암호 알고리즘과 연동을 고려한다면 개발비용이 증가한다는 결과를 볼 수 있다.

4.1.2 확장 용이성

암호 알고리즘 혹은 키 저장 모듈의 개발측면에서 CNG 확장은 독립적인 모듈화와 플러그인 기반 구조로 인하여 전체 구조에 대한 최소한의 이해로 최대한 개발할 수 있다는 장점이 있다. 암호 라이브러리를 적용하는 프로그램의 측면에서 볼 때, CryptAcquireContext에서 CSP 공급자의 이름을 명시하여야 하는 CAPI와 달리 CNG는 BCryptOpenAlgorithmProvider에서 암호 알고리즘만을 명시하기 때문에 암호 알고리즘 공급자를 교체하여야 할 경우에도 응용 프로그램의 소스코드를 변경할 필요가 없어 개발자 입장해서 유지보수에 대한 부담을 최소화하여 확장을 용이하게 한다. 암호 라이브러리 공급자를 변경할 경우 응용 프로그램에서의 변화를 Fig. 15.에 나타내었다.

CAPI의 확장을 위하여 새로운 CSP를 개발할 경우, 보안상의 이유로 반드시 마이크로소프트사로부터 서명을 받아야 배포되고 실행된다. 이러한 구조는 보안성을 강화시킬 수 있으나 라이브러리의 신속한 배포를 저해하는 걸림돌이 되지만, CNG의 경우, 마이크로소프트사의 서명 없이 자유롭게 배포하고 실행 가능하다.

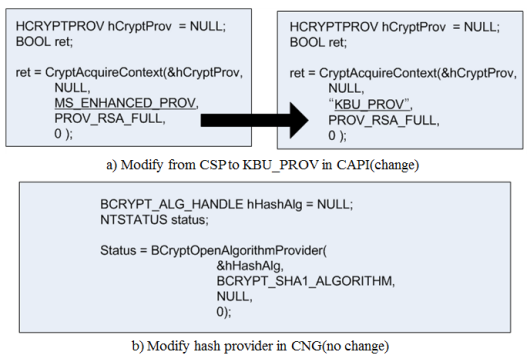

Fig. 15. Changing code when modification of cryptography library provider

4.1.3 보안성

마이크로소프트사의 서명을 반드시 받아야 새로운 CSP를 적용할 수 있는 CAPI와 달리, 아무런 제약이 없는 CNG의 경우 악의적인 공급자들이 설치되

고 실행되는 것을 방지하기에는 다소 무리가 있다. 따라서 signature를 이용하여 등록된, 혹은 등록하는 공급자의 식별이 가능하도록 할 수 있으며, 식별자를 토대로 공급자의 위험성을 검증하기 위한 별도의 보안 솔루션이 필요할 것으로 판단된다. 이러한 솔루션은 등록 전, 정적으로 패턴을 검사하거나, 실시간으로 동적 검사를 수행하여 보안성을 평가하기 위한 방안이 요구될 것으로 사료된다.

4.1.4 국산 암호 알고리즘과 CNG

앞서 언급된 것처럼 CNG는 기존의 암호 라이브러리와 비교할 때, 뛰어난 확장성을 제공한다. 이러한 특징은 국산 암호 알고리즘 사용이 필수인 국내 환경에 매우 적합하며, 암호 알고리즘 공급자와 키 관리 공급자이외에도 SSL 공급자를 지원함으로써 기존의 SSL 구조에 대해 손쉬운 확장이 가능하도록 하였다. 국산 암호 알고리즘의 장착을 위한 SEED, ARIA, HAS-160의 경우, 암호 알고리즘 공급자를 구현함으로써 CNG의 확장이 가능하고, KCDSA의 경우는 암호 알고리즘 공급자를 구현하거나 키 관리 공급자를 구현함으로써 확장이 가능하다. 아울러, SSL 확장을 위해서 국산 알고리즘 공급자 혹은 키 관리 공급자를 활용한 SSL 공급자를 개발할 수 있다.

4.2 CNG 커스터마이징 전략 연구

일반적으로 기업이나 공공기관의 환경에서는 조직의 업무를 효과적으로 지원할 수 있는 다양한 응용 프로그램들이 구축되어 활용된다. 응용 프로그램들의 보안성을 일관되고 중앙 집중적인 방식으로 신뢰할 수 있도록 관리하기 위해서는 보안 인프라의 핵심 요소인 암호 라이브러리에 대한 개발지침과 관련 정책을 마련하는 것이 매우 중요하다. 이를 위해 먼저 적합한 암호 라이브러리를 선정하여 조직의 요구사항에 맞게 커스터마이징하여야 하며, 개발자들에게 개발지침에 따라 커스터마이징된 라이브러리를 활용할 것을 강력히 권고하여야 한다. 아울러 각 응용 프로그램들이 실제로 개발지침에 맞게 암호 라이브러리를 사용했는지 평가하여야 하며, 실행 중에 암호 라이브러리가 어떻게 활용되는지 추적하여야 한다. 상기에서 분석된 것처럼 CNG는 잘 정의된 플러그인 기반의 모듈구조로 구성되어 있고 확장에 용이한 기능을 제공함으로써 최소의 개발비용으로 신속한 확장을 가능하게 한다. 이러한 장점은 기업이나 공공기관에서 조직 고유의 보안 요구사항에 따라 빠르고 쉽게 암호 라이브러리를 커스터마이징하는데 큰 이점을 가진다. 또한, CNG는 이미 Suite B 알고리즘을 비롯하여 최신 암호 알고리즘들을 지원하고 커널모드에서 활용가능하며, 라이브러리에 대한 감사기능을 제공하기 때문에 기업이나 공공기관을 위한 암호 라이브러리로서 가장 적합하다고 볼 수 있다.

한 조직에서 CNG의 도입을 결정하였다면, 먼저 암호 라이브러리 활용에 대한 조직 고유의 보안요구사항을 분석한다. 분석과정에서 CNG의 확장이 필요하다면, 이와 관련된 암호 알고리즘 혹은 키 관리 공급자를 구현한다. 그 다음 CNG 활용지침을 작성하여 새롭게 구현된 공급자와 함께 응용 프로그램 개발자에게 전달하면, 응용 프로그램 개발자는 주어진 활용지침과 확장된 CNG 라이브러리를 사용하여 프로그램을 개발하거나 갱신한다. 만일, 개발된 응용 프로그램이 CNG 활용지침에 맞게 개발되었다면, 새롭게 구현된 CNG 공급자와 함께 배포하여 사용한다. CNG를 조직의 암호 라이브러리로 도입을 하기 위해 특별히 고려하여야 할 사항은 다음과 같다.

・안전한 CNG 공급자 배포

새롭게 개발된 CNG 공급자를 쉽고 안전하게 사용자 컴퓨터에 배포하는 것이 매우 중요하다. CAPI의 경우, 마이크로소프트의 서명을 바탕으로 신규 CSP를 검증할 수 있으나, CNG에는 이러한 메커니즘이 정의되어 있지 않다. 따라서 CNG 도입을 결정한 조직에서는 잘 정의된 스마트 업데이트 메커니즘을 구축하는 것이 중요하다.

・CNG 라이브러리 모니터링

응용 프로그램이 CNG 라이브러리를 어떻게 활용하는지 중앙 집중적인 방식으로 모니터링을 하는 것이 요구된다. 이를 위하여, 각 사용자 컴퓨터에서 생성된 CNG 감사기록을 모니터링 서버에 보내는 방식으로, CNG 라이브러리 활용 현황을 모니터링 서버 측에서 실시간 그리고 통합적으로 분석할 수 있게 해야 하는 것이 바람직하다.

・응용 프로그램 평가 방안

실제 응용 프로그램들이 확장된 CNG 라이브러리를 주어진 활용지침에 따라 적합하게 적용하였는지를 평가하는 평가 도구가 필요하다.

V. 결 론

본 연구에서는 마이크로소프트의 차세대 암호 라이브러리 CNG의 구조 및 특징과 전형적인 프로그램 기법을 연구하였고, 국내 해쉬 알고리즘인 HAS-160을 CNG에 장착하기 위한 해쉬 공급자를 구현하였다. 아울러, CNG의 확장성 분석과 함께 CNG의 커스터마이징 전략을 제안하였다.

본 논문에 의하면 CNG는 잘 정의된 독립적인 모듈과 플러그인 구조를 바탕으로 최소의 개발비용과 함께 신속한 라이브러리 확장을 지원한다. 또한, 최신 암호 알고리즘 제공과 감사기능, 커널지원 등과 같이 차세대 암호 라이브러리에 요구되는 의미있는 기능을 가진다. 이러한 장점은 CNG가 기존 암호 라이브러리의 한계를 뛰어 넘어 '암호 서비스'로서의 새로운 패러다임 시대를 열게 한다. 이에 다양하고 많은 응용 프로그램을 활용하는 기업 혹은 공공기관에서 각 프로그램의 암호화 동작을 일관된 방식으로 통합 관리하고 보안수준을 유지할 수 있는 보안 인프라로서의 역할을 제시하였다.

본 연구과정에서 구현된 국산 암호 알고리즘 HAS-160을 위한 CNG 공급자는 윈도우즈 비스타로 대표되는 차세대 64비트 윈도우즈 컴퓨팅 환경에서 국산 암호 알고리즘의 본격적인 연동을 위한 중요한 사례로 활용될 것이다. 향후 연구로서 새롭게 구현된 CNG 공급자를 안전하게 배포하기 위한 방안과 CNG 라이브러리 연산에 대한 통합적인 모니터링 기법에 대한 연구가 요구되며, HSA-160 이외에 ARIA와 SEED, KCDSA를 위한 암호 알고리즘 공급자의 구현과 함께 이를 활용한 SSL 공급자의 개발을 통해 윈도우즈 비스타 환경에서 국산 알고리즘의 유기적인 연동을 도모하는 연구를 할 예정이다.

References

[1] Microsoft, "Using Cryptography," Microsoft Developer Network, http://msdn.microsoft.com/en-us/library/aa388162(VS.85).aspx

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

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

[4] Quartz, "Next Generation of Cryptograpgy for Microsoft Windows Vista," Codeproject, http://www.codeproject.com/KB/vista-security/CryptographyNextGenDemo.aspx, 2007

[5] Jaecheon Byun, Kyungroul Lee, Wansoo Kim, Ilsun You, and Kangbin Yim, "Analysis of agility on a new crypto framework CNG," KSII-A'10, pp. 77-78, Oct. 2010

[6] K. Lee, Y. Lee, J. Park, I. You, and K. Yim, "Security Issues on the CNG Cryptography Library (Cryptography API: Next Generation)," In Proceedings of the International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), pp. 709-713, Jul. 2013

[7] Kyungroul Lee, Ilsun You, and Kangbin Yim, "Vulnerability Analysis on the CNG Crypto Library," In Proceedings of the International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing(IMIS), Jul. 2015

[8] Hyoungshick Kim, Junho Huh, and Ross Anderson, "On the Security of Internet Banking in South Korea," CS-RR-10-01, Computer Science Group, Oxford University Computing Laboratory, 2011

〈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

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