전력 계통 보안을 위한 자바용 KCMVP 암호모듈 구현 및 검증에 관한 연구

김수현, 김태훈, 김민용, 이선우, 이준영*

한전KDN(주)

suhyeon2_12@kdn.com, thkim_5117@kdn.com, kmyong_0902@kdn.com, hagler_sunwoo1@kdn.com, *ljy.953386@kdn.com

A Study on Implementation and Verification of KCMVP Cryptographic Module for Java in Power System Security

Suhyeon Kim, Taehun Kim, Minyong Kim, Sunwoo Lee, Junyoung Lee*

KEPCO KDN

요 약

전력 분야에서는 암호모듈의 안전성과 구현 적합성 검증을 위해 KCMVP 인증 획득이 의무화되어 있다. 그러나 현재 Java 환경을 지원하는 소프트웨어 암호모듈은 제한적이다. 본 논문에서는 다양한 전력 계통 시스템에서의 안전한 암호모듈 적용을 위해 표준 라이브러리인 JCA/JCE를 활용하여 Java용 KCMVP 암호모듈을 구현한다. 이를 통해 개발자는 암호 표준 라이브러리를 수정하지 않고도 안전한 암호모듈 개발이 가능하다. 구현된 암호모듈은 KCMVP 검증 대상 암호 알고리즘(V3.0)을 포함하며, 엔트로피 테스트 및 CAVP를 통해 암호모듈 구현 적합성을 검증한다.

Ⅰ. 서 론

전력 분야에서는「사이버 안보 업무 규정」제9조,「전자정부 시행령」제 69조 등 관련 법령에 의거하여 암호모듈의 안전성과 구현 적합성을 검증하는 KCMVP 인증을 획득한 암호모듈 사용을 의무화하고 있다.

객체지향 언어의 편리성, 전자정부 표준 프레임워크 도입 등으로 최근 공공 분야에서 Java가 지속적으로 활용되고 있으며 전력 분야의 AMI, DAS 등 상위 시스템에도 Java가 활용되고 있다. 이에 따라 상위 시스템 및 연계 설비 간 통신 시 보안성 강화를 위해 다양한 암호 알고리즘 및 기능을 지원할 수 있는 Java 기반의 암호모듈 개발 요구가 지속 요청되고 있지만 현재 Java 환경을 지원하는 소프트웨어 암호모듈은 전체 알고리즘 목록 중 약 10% 수준에 불과하여 암호 기술 선택의 자율성과 다양한 활용에 어려움을 겪고 있는 실정이다.

본 논문에서는 다양한 전력 계통 시스템에서 안전한 적용을 위해 암호 표준 라이브러리인 JCA/JCE를 활용하여 Java 기반 암호모듈을 구현하는 방안을 제안하고, 구현 및 검증을 통해 전력 계통 기기의 암호화 통신 효율성을 높인다.

Ⅱ. 본론

2.1 암호모듈 검증 기준

국내 암호모듈 검증 제도(KCMVP)에서는 안전성, 신뢰성, 상호운용성 등을 고려하여 표준 암호 알고리즘을 선정하고 있다. 암호모듈에 탑재된 암호 알고리즘은 '암호 알고리즘 검증 기준 V3.0'에 따라 정상 구현 여부를 시험받는다.

[표 1]과 같이 KCMVP 검증 대상 암호 알고리즘은 크게 블록암호, 공개키 암호, 해시함수, 메시지 인증, 난수 발생기, 키 설정, 키 유도, 전자서명 등으로 구분된다. 각 알고리즘 별로 다양한 모드, 키 길이, 기능 등에 대한 요구사항이 명시되어 있다. 이러한 요구사항을 모두 충족하는 암호모듈에 한하여 KCMVP 인증이 부여된다.

구 분	알고리즘 목록
블록암호	ARIA, SEED, LEA, HIGHT
공개키 암호	RSAES
해시함수	SHA2, SHA3, LSH
메시지 인증	CMAC, GMAC, HMAC
난수 발생기	CTR-DRBG, HASH-DRBG, HMAC-DRBG
키 설정	DH, ECDH
키 유도	KBKDF, PBKDF
전자서명	RSA-PSS, KCDSA, EC-KCDSA, ECDSA

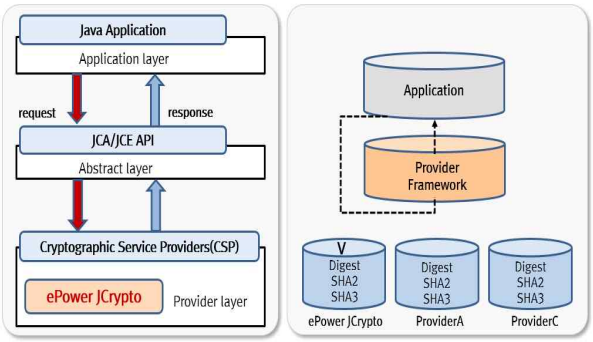
[표 1] KCMVP 검증대상 암호 알고리즘(V3.0)

2.2 Java 암호모듈 구현

JCA(Java Cryptography Architecture)와 JCE(Java Cryptography Extension)는 Java 플랫폼에서 암호화, 키 관리, 전자서명, 해시 등의 보안기능을 제공하는 프레임워크로, Provider Framework와 Provider를 통해 암호 서비스를 정의 및 지원한다. JCA/JCE를 이용한 방식은 표준 라이브러리의 활용으로 인해 안전성이 보장된다. 또한, 응용 프로그램에서 인스턴스 인자만 수정함으로써 Java 환경에서 암호모듈을 활용할 수 있어 사용자의 편의성이 향상된다.

본 논문에서는 Java 암호 아키텍처인 JCA와 Java 암호화 확장인 JCE를 활용하여 Java 기반 KCMVP 인증 암호모듈을 구현하였다. 구현된 Java 암호모듈은 [그림 1]과 같이 Provider와 암호모듈로 구성되어 있다. Java Application에서 JCA/JCE 인터페이스를 통한 암호 기능 호출 시, Provider Layer에서는 지정된 Provider가 있는지 탐색한다. 사용자 관점에서 보았을 때, JCA/JCE 인터페이스 getInstance를 통해 특정 Provider와 특정 알고리즘을 지정하여 사용

할 수 있다.



〔그림 1〕 Java 암호모듈 구성도

Provider는 JCA/JCE 엔진 클래스를 통해 구현되었다. [그림 2]와 같이 엔진 클래스에는 각 암호 알고리즘 별 구현체가 포함되어 있으며, 새로운 알고리즘 추가를 위해 엔진 클래스를 상속하여 구현하였다. 이렇게 구현된 Provider를 암호모듈에 탑재하여 Java 애플리케이션에서 사용할 수 있도록 하였다.

블록암호 알고리즘으로는 ARIA, LEA를 지원하며, 해시함수는 SHA2, SHA3, 메시지 인증은 HMAC을 지원한다. 난수 발생기는 CTR-DRBG, , 키 설정은 ECDH, 키 유도 기능에서는 KBKDF, PBKDF, 전자서명은 ECDSA와 EC-KCDSA의 알고리즘을 구현하였다. 이와 같이 구현한 Java 암호모듈은 KCMVP 검증 대상 암호 알고리즘을 포함하고 있다.

```
Cipher cipher = Cipher.getInstance("ARIA/GCM/NOPADDING",
                                   new EpowerJCryptoProvider());
SecretKeySpec keySpec = new SecretKeySpec(key, "ARIA");
GCMParameterSpec gcmParamSpec = new GCMParameterSpec(tagString.length()
                                        * 4, iv, 0, iv.length);
cipher.init(Cipher.ENCRYPT_MODE, keySpec, gcmParamSpec);
cipher.updateAAD(aad);
byte[] enc = cipher.doFinal(pt);
TestUtils.printHex(prefix:"ENC", enc);

Assertions.assertArrayEquals(ct, enc);

cipher.init(Cipher.DECRYPT_MODE, keySpec, gcmParamSpec);
cipher.updateAAD(aad);
byte[] dec = cipher.doFinal(enc);
TestUtils.printHex(prefix:"DEC", dec);
```

〔그림 2〕 JCA/JCE엔진 클래스

2.3 Java 암호모듈 시험 및 결과

KCMVP 인증시험에서는 엔트로피 테스트와 CAVP(Cryptographic Algorithm Validation Program)로 구현 적합성을 확인한다.

엔트로피 수집은 난수발생기 사용을 위해 이루어진다. 엔트로피는 SecureRandom 클래스를 사용하여 수집하였다. 암호모듈 검증 관리 및 사전 검증 서비스를 통해 측정한 잡음원의 엔트로피는 7.227618로, 인증 기준을 충족한다.

〔그림 3〕 엔트로피 테스트 결과

CAVP는 KAT(Known-Answer Test)와 MCT(Monte-Carlo Test)를 이용하여 알고리즘 구현 적합성을 검증한다. 각 검증 대상 암호 알고리즘에 대한 시험 시뮬레이터를 제작한 후, 테스트는 KISA 암호 알고리즘 검증 기준 V3.0의 테스트 벡터를 사용하였다. 배포되는 txt 파일을 가공해 req 파일 및 rsp를 생성한 후, txt 파일과 비교하여 답안 일치 여부를 판별한다, 전체 알고리즘 정상 동작을 확인하였고, 모든 테스트 벡터 통과를 완료하였다.

```
INFO| -------------- TEST [ ECDSA_(P-256)(SHA-256) Sign ] START --------------
     EXP R| F3AC8061B514795B8843E3D6629527ED2AFD6B1F6A555A7ACABB5E6F79C8C2AC
     EXP S| 8BF77819CA05A6B2786C76262BF7371CEF97B218E96F175A3CCDDA2ACC058903

INFO| Constructor [EpowerJCryptoProvider]
INFO| Check if module state is [KDN_MODULE_APPROVAL]
INFO| Module state is [KDN_MODULE_APPROVAL]
INFO| EpowerJCryptoProvider setup the algorithms
        R| F3AC8061B514795B8843E3D6629527ED2AFD6B1F6A555A7ACABB5E6F79C8C2AC
        S| 8BF77819CA05A6B2786C76262BF7371CEF97B218E96F175A3CCDDA2ACC058903
INFO| ------------------------------ TEST END ------------------------------


INFO| -------------- TEST [ ECDSA_(P-256)(SHA-256)_KPG ] START ---------------

    EXP Yx| 1CCBE91C075FC7F4F033BFA248DB8FCCD3565DE94BBFB12F3C59FF46C271BF83
    EXP Yy| CE4014C68811F9A21A1FDB2C0E6113E06DB7CA93B7404E78DC7CCD5CA89A4CA9

INFO| Constructor [EpowerJCryptoProvider]
INFO| Check if module state is [KDN_MODULE_APPROVAL]
INFO| Module state is [KDN_MODULE_APPROVAL]
INFO| EpowerJCryptoProvider setup the algorithms
       Yx| 1CCBE91C075FC7F4F033BFA248DB8FCCD3565DE94BBFB12F3C59FF46C271BF83
       Yy| CE4014C68811F9A21A1FDB2C0E6113E06DB7CA93B7404E78DC7CCD5CA89A4CA9
INFO| ------------------------------ TEST END ------------------------------


INFO| -------------- TEST [ ECDSA_(P-256)(SHA-256)_SGT ] START ---------------

INFO| Constructor [EpowerJCryptoProvider]
INFO| Check if module state is [KDN_MODULE_APPROVAL]
INFO| Module state is [KDN_MODULE_APPROVAL]
INFO| EpowerJCryptoProvider setup the algorithms
    EXPECT| TRUE

   message| 036C411A7C3CCAC2519C2D99260470CF2347AEDC16DB632A941ACCB1390C9F51
          C9C02C2D3BBC28CE69872C5C8DE136AE2BA121838ED1819C9203B40FCB72B196
          C0E95D1B640BE74E68F2029797B76E18E6E7B5165C74858D75198E3433D44DBF
          8060AAA0024FD6131AC048C11E354CB892A704066E3D44C42AD137CBB1B52AEB
       Yx| DE4B0ADFC08C51A8D74F5A459676FEA8B18E78C2882533373BB3310F7C3D40BE
       Yy| 0F7A5DA1C5AF8016A0D99ECF364032C6F38ACDC980BC13FEC41FC392ACBFA0DC
        R| 65D63F035966FC50881AB58B95A57B17279B28D7CE366D5DC9E0BB083C2E4387
        S| AED594BFCB528B334739BAD1BA4440F4452ED802B6760C9D7EA16011F335CC41

    RESULT| TRUE
INFO| ------------------------------ TEST END ------------------------------
```

〔그림 4〕 ECDSA CAVP 테스트 결과

Ⅲ. 결론

본 논문에서는 전력 계통 기기의 암호화 통신 효율성을 높이기 위해 JCA/JCE 프레임워크를 통한 보안기술을 제안하고 구현 및 검증하였다. 이를 통해 전력 계통 암호모듈이 전력 계통 상위 시스템에 적용되어 ICT 전 분야로 확대할 수 있다. 향후 KCMVP 인증을 획득 후, Java 서버 환경에서 JNI를 활용하는 C언어 암호모듈과 JCA/JCE를 활용하는 Java 암호모듈을 비교 및 고찰할 예정이다.

참 고 문 헌

[1] 박재필. “KCMVP를 준수하는 TLS Library의 구현”, 한국컴퓨터정보학회 학술발표논문집, pp 331-334, 2019.7.

[2] 이희웅, 홍도원, 김현일, 서창호, 박기식. “SHA-3 해시 함수 검정 프로그램과 16bit-UICC용 SHA-3 구현”, Journal of KIISE, pp 885-891, 2014.11

[3] 손미경, 강남희. “안드로이드 플랫폼을 위한 자바 보안 프로바이더 설계 및 구현”, 한국통신학회논문지, 37(9), pp 851-858, 2012.09

[4] 김영범, 서석충. “Metamorphic Testing 기반 암호알고리즘 구현 정확성 검증 기술 동향 조사”, 한국정보과학회 학술발표논문집. 2021.06

[5] 강희운, 이기영. “QRNG 암호모듈 내장형 마이크로 PMU를 활용한 전력계통 보안강화”, 대한전기학회 학술대회 논문집. 2023.05