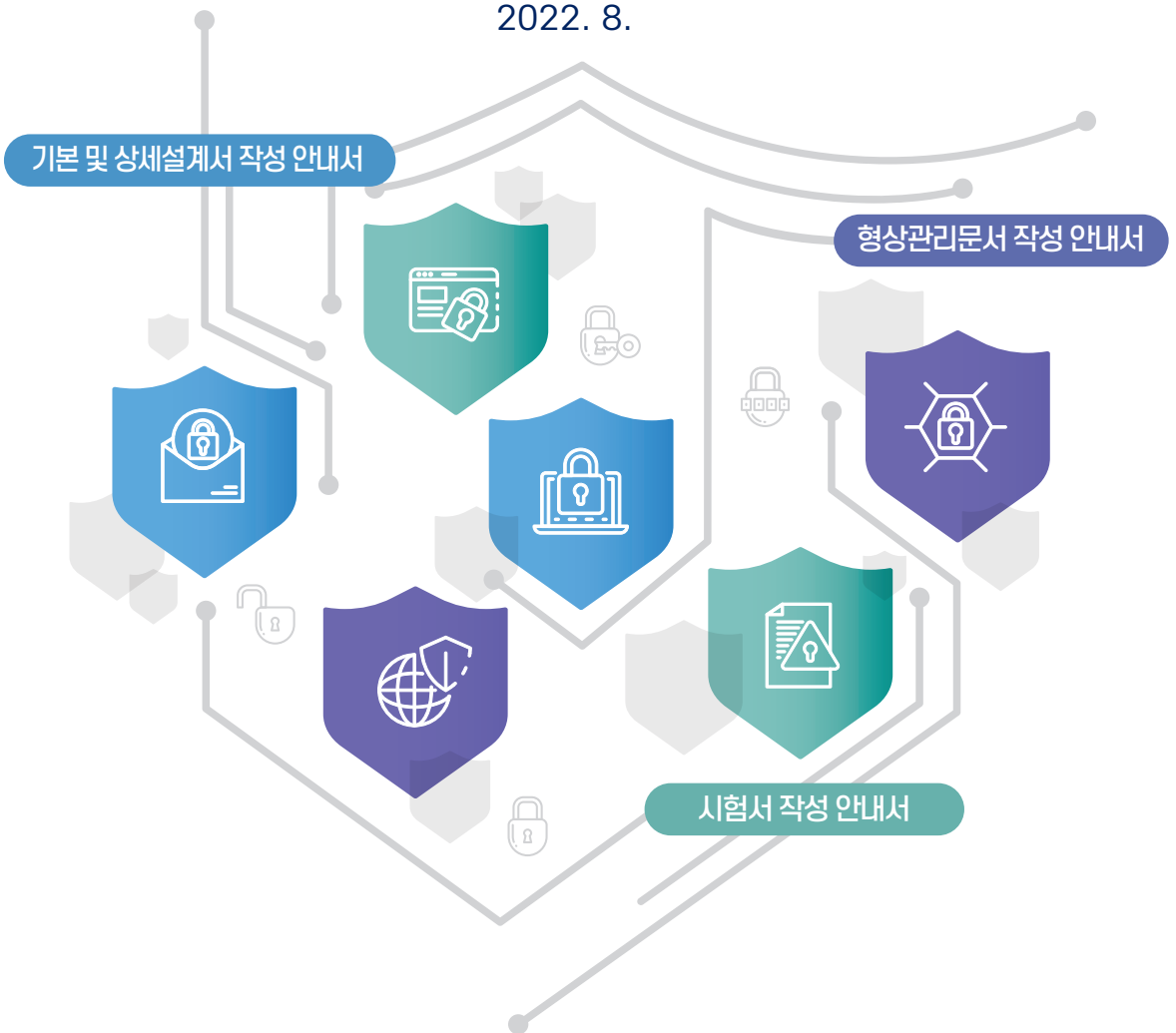
암호모듈 제출물 작성 안내서

2022. 8.



과학기술정보통신부 Ministry of Science and ICT
국가정보원 NATIONAL INTELLIGENCE SERVICE
KISA 한국인터넷진흥원 KOREA INTERNET & SECURITY AGENCY
NSR 국가보안기술연구소

CONTENTS

I

기본 및 상세설계서 작성 안내서

● **제·개정 이력**

순번	제·개정일	제·개정 내역
1	2022. 8.	“암호모듈 제출물 작성 안내서“ 발간

암호모듈 제출물
작성 안내서

II
형상관리문서 작성 안내서

III
시험서 작성 안내서

암호모듈 제출물
작성 안내서

I

기본 및 상세설계서 작성 안내서

Ⅰ 기본 및 상세설계서 작성 안내서

1 개요

이 가이드는 KS X ISO/IEC 19790:2015, KS X ISO/IEC 24759:2015 표준문서에 근거하여 국내 암호모듈검증제도에서 요구되는 기본 및 상세설계서를 작성하는 데 참고할 수 있는 내용을 담고 있다. 해당 표준에서 정의한 보안수준 1에 해당하는 소프트웨어 암호모듈로서 갖추어야 할 보안요구사항에 대해 기본 및 상세설계서를 작성하는 방법을 안내한다.

2 문서의 구성 및 용어

가. 구성

이 가이드는 KS X ISO/IEC 19790:2015와 KS X ISO/IEC 24759:2015에서 제시하고 있는 총 11가지 중 9가지 보안요구사항 영역에 대해서 순차적으로 기본 및 상세설계서의 내용을 작성하는 방법에 대해 다룬다. 물리적 보안 및 비침투 보안은 소프트웨어 암호모듈에서 다루지 않는다.

나. 시험항목과 보안요구사항

KS X ISO/IEC 19790:2015는 암호모듈에 대한 보안요구사항을 영역별로 제시하고 있다. 시험항목은 다음과 같이 요구사항 영역별 시험항목 번호(AS: Assertion)로 나타낸다.

AS〈요구사항 번호〉.〈시험항목 번호〉

여기에서 '요구사항 번호'는 해당 보안요구사항 영역 번호이며, '시험항목 번호'는 보안요구사항 영역 내에서 시험항목의 식별을 위한 번호이다. 각 시험항목 다음에는 벤더가 고려해야 하는 요구사항(VE: Vendor Evidence)이 있다. 즉 암호모듈 개발업체가 시험항목 번호에서 제시하는 내용을 충족하기 위해 어떻게 암호모듈을 설계하고 구현했는지에 대한 증빙을 제시하는 부분이다. 벤더는 개발한 암호모듈의 유형, 보안수준 등에 연관된 시험항목에서 요구하는 모든 벤더요구사항에 대해 성실하게 설계서를 작성하고 암호모듈을 구현해야 한다. 벤더가 작성을 해야 하는 벤더요구사항은 다음과 같은 형태로 표기된다.

VE〈요구사항 번호〉.〈시험항목 번호〉.〈순서 번호〉

3 암호모듈 명세

가. 목적

암호모듈은 검증대상 암호알고리즘이나 프로세스를 사용하여 최소한 하나 이상의 암호 서비스를 구현하고 암호경계 내에 포함된 하드웨어나 소프트웨어, 펌웨어 또는 이들의 결합 형태이어야 한다. 암호모듈 명세는 암호모듈의 전체적인 구조와 경계, 보안수준, 탑재하고 있는 보안 서비스에 대한 전반적인 보안요구사항을 제시하며, 세부적인 요구사항은 다음과 같다.

- 암호모듈 유형 및 판단근거
- 암호모듈 내 모든 검증대상 암호알고리즘 및 동작모드
- 암호모듈의 (통합)경계
- 암호모듈과 연결된 모든 구성요소
- 노출/변경 시 암호모듈의 안전을 손상시킬 수 있는 모든 보안 관련 정보 명세

이 장에서는 암호모듈 명세의 주요 보안요구사항을 만족시키기 위해 개발업체가 작성해야 하는 AS 항목별 VE에 대한 작성 가이드를 제시한다.

나. 암호모듈 유형

암호모듈은 하드웨어, 소프트웨어, 펌웨어, 하이브리드(소프트웨어, 펌웨어)의 유형 중 한 가지로 정의되어야 하며, 암호모듈 유형을 기술한 개발 문서를 제출해야 한다. 개발 문서에는 암호모듈의 전반적으로 소스코드 설명, 동작 과정이 필요한 경우 포함해야 한다.

항목	AS02.03
보안요구사항 개요	암호모듈 유형의 선택 및 근거 제시
VE02.03.01	모듈 유형과 유형 선택의 근거를 제시한 개발 문서 제출
VE02.03.02	암호모듈의 모든 구성요소를 식별하는 개발 문서 제출

작성 예시

KISACrypto(암호모듈명)는 검증대상 암호알고리즘과 키 생성을 포함하는 알고리즘 함수와 프로세스로 구현한 소프트웨어 조합의 집합으로 '.dll' 및 '.so' 형태(또는 .dylib, .ko 등으로도 구현된다.)로 구성된 소프트웨어 모듈이다. 이 암호모듈은 KS X ISO/IEC 19790:2015 암호모듈 보안요구사항의 보안수준 1을 준수하여 개발되었으며, 비검증대상 암호알고리즘을 포함하지 않는다.

암호모듈명	KISACrypto
버전	V1.0
제조사	KISA(한국인터넷진흥원)
보안수준	보안수준1
암호모듈 유형	소프트웨어 모듈(동적 라이브러리)
구성요소	단일 파일로 구성 - KISACrypto32.dll KISACrypto64.dll(윈도우 환경) - libKISACrypto32.so, libKISACrypto64.so(리눅스 환경) 등
암호 서비스	블록암호, 해시함수, 메시지인증코드, 난수발생기, 공개키 암호, 전자서명, 키 설정 등
운영환경	윈도우 환경 - Win10 32/64비트 리눅스 환경 - Ubuntu 20.04 32/64비트

해설

암호모듈 유형의 선택과 근거를 제시한다. 위의 예시와 유사한 형태로 증빙 가능하다. 구성요소는 위의 예시와 같이 한 개의 단일 파일 형태, 암호모듈 파일과 무결성 검증 파일이 분리된 형태, .dll(또는 .so 등)이 여러개 의 파일로 구성된 형태 등 다양한 형태로 존재할 수 있다.

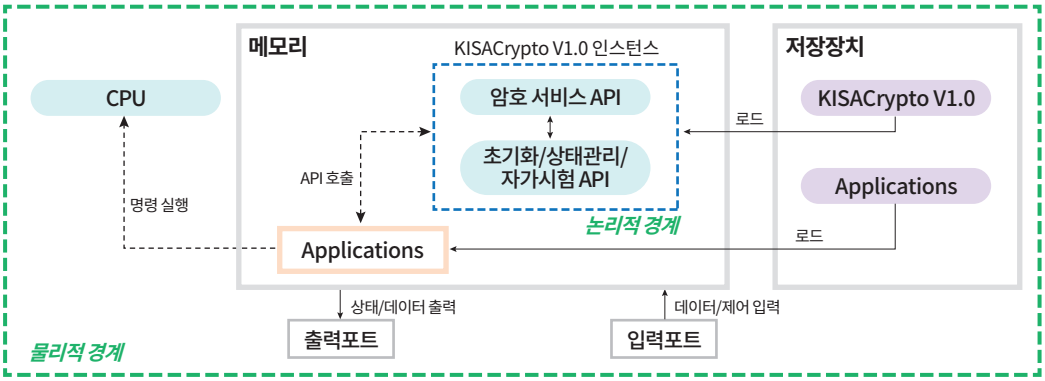
다. 암호모듈 경계

항목	AS02.07
보안요구사항 개요	암호경계 및 경계 내의 구성요소 명세 필요
VE02.07.01	경계 안의 모든 구성요소를 명세한 개발 문서 제출

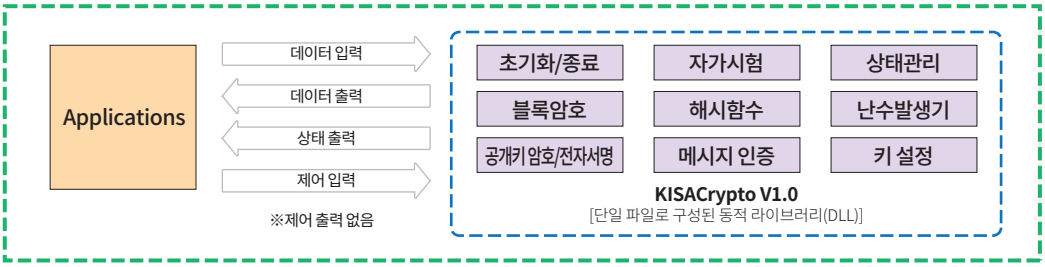
작성 예시

구분	구성요소	구성 형태	비고
SW	암호모듈	KISACrypto32.dll	윈도우용 라이브러리 32비트
		KISACrypto64.dll	윈도우용 라이브러리 64비트
		libKISACrypto32.so	리눅스용 라이브러리 32비트
		libKISACrypto64.so	리눅스용 라이브러리 64비트

[암호모듈의 구성요소]



[암호모듈 경계 예시(소프트웨어 암호모듈)]



[암호모듈의 기능적 구성요소]

해설

암호모듈의 파일별 구성요소, 암호모듈의 기능적 구성요소를 명세한다. 소프트웨어 암호모듈의 경계는 전형적인 형태이기 때문에 위의 예시를 참고하여 작성 가능하다.

항목	AS02.09
보안요구사항 개요	경계 내의 알고리즘, 프로세스 등 보안관련 요소 명세 필요
VE02.09.01	경계 안의 모든 구성요소를 명세한 개발 문서 제출

작성 예시

구분	핵심보안매개변수	검증대상 암호알고리즘
블록암호	비밀키, 라운드 키	SEED • 키 길이: 128비트 • 운용모드: ECB / CBC / CFB128 / OFB / CTR • 『KSX3254 n비트 블록 암호 운영모드-제1부:일반(2016), KSX3276 n비트 블록 암호 운영모드-제4부:SEED(2019)』에 기술된 SEED 대칭 블록 암호 알고리즘
해시함수	해당사항 없음	SHA-256 • 『KS X ISO/IEC 10188-3:2001-제3부 전용 해시함수(2018)』에 기술된 SHA-2 해시 알고리즘
메시지인증코드 (HMAC)	HMAC 키	HMAC(SHA-256) • 『KS X ISO/IEC 9797-2 메시지인증코드-제2부:전용 해시함수를 이용한 메커니즘(2018)』 에 기술된 HMAC 알고리즘
공개키 암호 (비대칭키)	개인키 파라미터 (d, dP, dQ, qInv), p, q, 시드	RSAES • 『KS X ISO/IEC 18033-2 암호알고리즘-제2부:비대칭형 암호(2017)』에 기술된 RSAES-OAEP 공개키 암호알고리즘 • 공개키 길이: 2048, Hash: SHA-256
난수발생기	엔트로피, 내부상태(Key, V), 시드, 난수	Hash_DRBG • 『KS X ISO/IEC 18031 난수발생기(2018)』에 기술된 난수발생기 • 예측내성 미지원, 추가 입력 미지원 • SHA256 사용
전자서명 (비대칭키)	개인키 파라미터 (d, dP, dQ, qInv), p, q, 솔트	RSA-PSS • 『KS X ISO/IEC 14888-2 부가형 디지털 서명-제2부:정수 인수분해 기반 메커니즘(2011)』에 기술된 RSASSA-PSS 전자서명 알고리즘 • 공개키 길이: 2048, Hash: SHA-256
키 설정	개인키, 공유키	DH 키 설정 • 『KS X ISO/IEC 11770-3 키 관리-제3부:비대칭 기법을 이용한 메커니즘(2019)』에 기술된 방식 • (공개키 길이, 개인키 길이) : (2048,256)

[암호알고리즘 목록 및 중요 보안매개변수 예시]

해설

암호모듈에 구현된 검증대상 암호알고리즘과 해당 알고리즘에서 사용되는 핵심 보안매개변수를 명세한다.
(관련 표준은 국가정보원 홈페이지 또는 한국인터넷진흥원 암호이용활성화 홈페이지 참고)

항목	AS02.10
보안요구사항 개요	비보안 요소 명세 및 비보안 요소가 검증대상 서비스에 영향을 미치지 않는 근거 명세
VE02.10.01	비보안 요소 명세
작성 예시	
KISACrypto V1.0은 보안수준 1의 범용 소프트웨어 라이브러리형 암호모듈로서 검증대상 암호알고리즘에 대한 암호 서비스만을 제공하고 비보안 요소를 포함하지 않는다.	
해설	
범용 소프트웨어 라이브러리형 암호모듈의 경우, 비보안 요소가 없을 경우 명확하게 없다는 것을 명세한다.	

항목	AS02.11
보안요구사항 개요	적절한 명칭의 사용 필요
VE02.11.01	적절한 명칭 사용의 근거 제시
작성 예시	
KISACrypto V1.0은 암호모듈검증제도 활성화를 위해 한국인터넷진흥원에서 개발한 검증대상 암호알고리즘만을 탑재한 범용 소프트웨어 라이브러리형 암호모듈이다. 이에, 개발기관과 암호 기능의 목적을 모두 포함한 KISACrypto V1.0으로 명명한다.	
해설	
개발업체, 용도 등을 확인할 수 있는 명칭을 부여한다.	

항목	AS02.12
보안요구사항 개요	구성요소에 대한 버전 정보 명시
VE02.12.01	구성요소에 대한 버전 정보
작성 예시	

구분	운영환경	구성 형태	비고
SW	Windows 10 32-bit	KISACrypto32.dll	KISACrypto V1.0 (무결성 검증 데이터는 암호모듈에 포함됨.)
	Windows 10 64-bit	KISACrypto64.dll	
	Ubuntu 20.04 32-bit	libKISACrypto32.so	
	Ubuntu 20.04 64-bit	libKISACrypto64.so	

[암호모듈의 구성요소 버전 정보]

해설
암호모듈의 구성요소에 대한 버전 정보를 명세한다.

항목	AS02.13
보안요구사항 개요	경계 외부의 요소가 검증대상 동작을 방해하거나 손상을 초래하지 않아야 함.
VE02.13.01	경계 외부의 요소에 영향받지 않는 근거 제시
작성 예시	
KISACrypto V1.0은 범용 소프트웨어 라이브러리형 암호모듈로서 해당 암호모듈을 링크하여 사용하는 응용 프로그램과 인터페이스를 통해 정보를 교환한다. 또한 암호모듈은 실행가능한 DLL(또는 so) 파일 형태이기 때문에 다른 외부 프로세서로부터의 간섭은 범용 운영체제를 통해 보호된다. 즉 암호모듈이 인스턴스 형태로 메모리에 로드된 이후에는 운영체제로부터 암호모듈 자체 및 암호모듈이 사용하는 메모리 영역은 외부 요소로부터 보호된다.	
해설	
운영환경과 관련하여 소프트웨어 암호모듈에서 사용되는 내부 데이터가 외부로부터 보호됨을 명세한다.	

항목	AS02.14
보안요구사항 개요	보안요구사항을 적용받지 않는 암호모듈의 구성요소 명세
VE02.14.01	보안요구사항을 적용받지 않는 모든 구성요소를 명세한 개발 문서 제출
VE02.14.02	보안요구사항을 적용받지 않는 모든 구성요소 각각에 대해 근거 제출
작성 예시	
KISACrypto V1.0은 보안요구사항을 적용받지 않는 구성요소가 존재한다(모든 구성요소 명세 필요). 각 구성요소는 오용 시 검증대상 동작모드에 손상을 초래할 수 있는 SSP, 평문 데이터 또는 기타 정보를 처리하지 않는다.	
해설	
보안요구사항을 적용받지 않는 모듈의 구성요소가 존재한다면, 각 구성요소가 오작동하거나 오용 시에도 암호모듈의 검증대상 동작모드를 손상시키지 않는 근거를 명세한다.	

항목	AS02.16
보안요구사항 개요	소프트웨어 암호모듈의 논리적 경계에 대한 범위 설정
VE02.16.01	구성요소를 명세한 개발 문서 제출 필요(구성요소, 소프트웨어 구조, 실행 환경 명세)

작성 예시

구분	구성요소		비고
하드웨어	저장장치	HDD	
		Main Memory	
		Flash Memory	
	연산장치	CPU	
	전원장치	Power	
	입출력장치	Mouse	
		Keyboard	
		Monitor	
소프트웨어	운영체제	Windows 10 (32-bit)	KISACrypto32.dll
		Windows 10 (64-bit)	KISACrypto64.dll
		Ubuntu Linux 20.04 (32-bit)	libKISACrypto32.so
		Ubuntu Linux 20.04 (64-bit)	libKISACrypto64.so

[암호모듈의 운영환경]

구분	소스코드	비고
암호모듈 관리	kisa_kcmvp_main.c kisa_kcmvp_main.h	- 모듈관리 - 모듈상태 출력 - 동작 전 자가시험 수행 - 모듈 버전 정보 출력 - 암호알고리즘에 대한 KCMVP 외부 인터페이스 정의
자가시험	kisa_kcmvp_selftest.c kisa_kcmvp_selftest.h	- 동작 전 자가시험 - 무결성 시험 - 조건부 자가시험
블록암호	kisa_seed.c kisa_seed.h	- SEED 블록암호
해시함수	kisa_sha.c kisa_sha.h	- SHA2 해시함수
메시지인증코드	kisa_hmac.c kisa_hmac.h	- HMAC_SHA2
난수발생기	kisa_hashdrbg.c kisa_hashdrbg.h	- Hash_DRBG 난수발생기(난수 비트열 생성)
공개키 암호	kisa_rsaes.c kisa_rsaes.h	- 공개키 암호(키쌍 생성, 암/복호화)
전자서명	kisa_rsapss.c kisa_rsapss.h	- 전자서명(키쌍 생성, 서명 생성/검증)
키 설정	kisa_dh.c kisa_dh.h	- 키 합의(키쌍 생성, 키 합의 수행)

해설

소프트웨어 암호모듈의 논리적 경계에 포함되는 파일에 대해 명세한다.

라. 동작모드

운영자는 검증대상 동작모드에서 암호모듈을 동작시킬 수 있어야 한다. 검증대상 동작모드는 검증대상 암호알고리즘 또는 보호함수를 이용하는 최소 하나 이상의 서비스 집합을 의미한다.

항목	AS02.19
보안요구사항 개요	검증대상 동작모드에 대한 필수 적용
VE02.19.01	암호모듈이 제공하는 검증대상 동작모드(구조, 절차) 설명
VE02.19.02	검증대상 동작모드에 대한 실행 방법 설명
작성 예시	
KISACrypto V1.0 암호모듈은 검증대상 동작모드로만 운영되며 비검증대상 동작모드는 지원하지 않는다. 윈도우 환경의 경우 암호모듈을 초기화하고 동작 전 자가시험을 수행하는 함수인 KISA_Crypto_initialize()가 DLL 파일이 메모리에 로드될 때 자동으로 호출되는 함수인 DllMain()에서 자동으로 호출되도록 구현하였다. 또한 리눅스 환경의 경우 KISA_Crypto_load() 함수를 constructor로 구현하여 내부에서 KISA_Crypto_initialize()가 호출되도록 하였다. 이와 같이 암호모듈이 메모리에 로드될 때 암호모듈이 자동으로 암호모듈의 내부 상태를 초기화하고 동작 전 자가시험을 수행하는 KISA_Crypto_initialize()를 호출하도록 하였으며, 동작 전 자가시험이 성공적으로 수행되면 검증대상 동작모드로 자동천이되어 동작하도록 하였다. 암호모듈을 검증대상 동작모드로 동작시키기 위해서는 암호모듈을 이용하는 응용 프로그램 빌드 시에 링크 과정을 통해 동적 라이브러리 형태의 암호모듈을 링크한 후에 응용 프로그램을 실행시키면 된다.	
해설	
암호모듈이 제공하는 검증대상 동작모드를 서술하고, 검증대상 동작모드를 실행시키기 위한 동작과정 등을 포함하여 명세해야 한다. 만약 위와 같이 암호모듈이 로드될 때 자동으로 초기화와 동작 전 자가시험을 수행하는 함수가 수행될 수 없는 환경이라면, 해당함수를 수동적으로 어떻게 호출하는지를 명세해야 한다.	

항목	AS02.20
보안요구사항 개요	탑재된 검증대상 암호알고리즘 명세
VE02.20.01	사전검토단계에서 구현적합성 검증을 수행할 검증대상 알고리즘 목록 제출

작성 예시

구분	알고리즘	중요 보안매개변수		검증대상 여부	구현적합성 검증
		핵심 보안매개변수	공개 보안매개변수		
블록암호	SEED-128	비밀키, 라운드 키	IV, 카운터	○	KAT, MMT, MCT
해시함수	SHA-256	-	-	○	짧은 메시지/긴 메시지/ 임의메시지 검증
메시지 인증	HMAC-SHA256	HMAC 키	-	○	임의메시지 검증
난수발생기	HMAC_DRBG (SHA-256)	엔트로피, 내부상태(Key, V), 시드, 난수	개별화 문자열, 추가입력	○	임의메시지 검증
공개키 암호	RSAES (2048, SHA256)	개인키 파라미터, 시드	공개키 파라미터	○	키쌍 생성/암호화/ 복호화 검사
전자서명	RSA-PSS (2048,SHA256)	개인키 파라미터, 솔트	공개키 파라미터	○	키쌍 생성/서명생성/ 서명검증 검사
키 설정	DH (2048, 256)	개인키, 공유키	공개키	○	키쌍 생성/ 공개키 유효성/키 설정기지 답안 검사

[암호모듈에 탑재된 검증대상 알고리즘 및 관련 중요 보안매개변수 명세]

해설

탑재한 검증대상 암호알고리즘에 따라 위의 표와 같이 명확한 알고리즘명과 관련 키 길이, 중요 보안매개변수, 구현적합성 검증 방법에 대해 명세가 필요하다.

항목	AS02.21
보안요구사항 개요	비검증대상 암호알고리즘 및 서비스에 대한 명세
VE02.21.01	검증대상 동작모드에서 사용되고 있는 비검증대상 암호알고리즘에 대한 명세 필요
VE02.21.02	비검증대상 암호알고리즘이 검증대상 동작모드의 동작과 관련이 없다는 근거 제시

작성 예시

KISACrypto V1.0 암호모듈은 내부에 비검증대상 암호알고리즘을 포함하지 않으며 검증대상 암호알고리즘만을 이용하여 보안서비스를 응용 프로그램에게 제공한다.

해설

비검증대상 알고리즘은 기본적으로 비검증대상 동작모드에서만 이용되어야 하나, 경우에 따라서는 검증대상 동작모드에서 비검증대상 암호알고리즘을 사용 가능함. 하지만, 해당 비검증대상 암호알고리즘이 검증대상 동작모드의 안전성에 영향을 미치지 않는 다는 명확한 근거 제시가 필요하다.

항목	AS02.22
보안요구사항 개요	동작모드 간, 서비스 간 CSP에 대한 독립적 분리 필요
VE02.22.01	동작모드/서비스별 사용되는 CSP 명세 필요
VE02.22.02	동작모드/서비스 간 CSP가 분리되는 방법 명세 필요

작성 예시

KISACrypto V1.0은 검증대상 동작모드만을 지원하며, 검증대상 동작모드에서 지원하는 검증대상 암호알고리즘 및 관련 CSP 목록은 VE02.09.01에서 제시하였다.

해설

하나의 암호모듈에서 검증대상 동작모드와 비검증대상 동작모드를 지원하는 경우 VE02.22.01에서 검증대상 동작모드와 비검증대상 동작모드에서 사용되는 암호알고리즘에 대한 CSP 목록을 제시해야 하며, 또한 검증대상 동작모드에서 비검증대상 동작모드로 천이할 경우, 사용되던 검증대상 동작모드의 CSP(핵심 보안매개변수)를 제로화하는 방법을 제시해야 한다. 암호모듈은 기본적으로 동작 전 자가시험을 거쳐 성공 시, 검증대상 동작모드로 자동천이되어야 하며, 관리자가 동작모드 변경 함수를 통해 비검증대상 동작모드로 천이될 수 있다. 모드 간 천이 시에는 기존 모드에서 사용하던 CSP에 대해서는 천이될 동작모드와 독립적으로 분리할 수 있는 방법을 제시해야 하며, 대표적인 방법으로는 제로화가 사용될 수 있다.

항목	AS02.24
보안요구사항 개요	검증대상 동작모드에서 서비스에 대한 표시기 제공 필요
VE02.24.01	개발문서에 서비스에 대한 표시기 명세 필요

작성 예시

KISACrypto V1.0 암호모듈은 제공하는 암호서비스별로 성공·실패 여부를 확인할 수 있는 표시기를 제공한다. 또한 암호모듈의 현재 동작모드와 상태를 확인할 수 있는 서비스도 제공한다. 암호모듈이 제공하는 각 암호서비스(함수 또는 API)에 대한 상태출력에 대한 정의는 기본 및 상세설계서의 '암호모듈 인터페이스'에서 확인 가능하다. 아래 표는 일부 암호 서비스에 대한 상태출력을 통한 표시기 명세이다.

함수	int KISA_Crypto_initialize(void)
데이터 입력	없음.
데이터 출력	없음.
제어 입력	API 호출
상태 출력 (표시기)	- KISA_SUCCESS (0x0001) : 초기화 성공 - KISA_CRYPTO_ERROR_SELFTEST_FAILED (0x0002) : 자가시험 실패 - KISA_CRYPTO_ERROR_STATE_IN_ERROR (0x0003) : 오류 상태
제어 출력	없음.
설명	암호모듈을 초기화하고 동작 전 자가시험을 수행한다. 동작 전 자가시험이 성공할 경우, 암호모듈을 검증대상 동작모드로 자동천이하고, 성공에 대한 코드를 반환한다. 실패할 경우, 오류코드를 반환한다.

함수	int KISA_Crypto_getState(void)
데이터 입력	없음.
데이터 출력	없음.
제어 입력	API 호출
상태 출력 (표시기)	– KISA_CM_LOAD (0x10001) – KISA_CM_PRE_SELFTEST (0x10002) – KISA_CM_COND_SELFTEST (0x10003) – KISA_CM_NORMAL_EXECUTION (0x10004) – KISA_CM_SIMPLE_ERROR (0x10005) – KISA_CM_HARD_ERROR (0x10006) – KISA_CM_EXIT (0x10007)
제어 출력	없음.
설명	암호모듈의 모드 상태와 동작모드에 대한 표시기를 반환한다. 암호모듈 운영자는 해당 함수에서 반환된 표시기를 통해 현재 암호모듈의 동작 상태를 확인할 수 있다.

해설

소프트웨어 암호모듈의 경우에는 암호모듈에 대한 서비스 수행은 함수 또는 API 호출을 통해 이루어지며 반환값(상태정보)이라는 표시기를 통해 성공 또는 실패 여부를 확인할 수 있다. 또한, 현재 동작모드에 대한 표시기를 출력하는 함수 또는 서비스를 포함하여 운영자에게 암호모듈의 동작모드 정보를 제공할 수 있다. 따라서, 암호모듈 내부에서는 각 암호 서비스에 대한 성공/실패 여부에 대한 오류값 또는 상태값을 정의해야한다. 또한, 암호모듈의 현재 상태를 확인할 수 있도록 암호모듈이 가질 수 있는 상태 정보를 정의해야 하며, 동작모드 확인 함수가 호출되었을 때 암호모듈의 현재 동작모드 또는 상태정보를 반환해야 한다.

4 암호모듈 인터페이스

가. 목적

암호모듈은 모든 논리적 정보의 흐름을 암호모듈 경계의 입출구로 식별되는 물리적 접근 지점과 논리적 인터페이스로 제한해야 한다. 또한 논리적 인터페이스가 하나의 물리적 포트를 공유하거나 하나 이상의 물리적 포트로 분산된 경우에도 각 인터페이스는 구분될 수 있어야 한다. 논리적 인터페이스의 종류는 총 5가지로, 데이터 입력, 데이터 출력, 제어 입력, 제어 출력, 상태 출력 인터페이스이다. 데이터는 평문, 암호문, 암호키와 같이 암호 서비스를 수행하는 데 필수적으로 요구되는 정보이며, 소프트웨어 암호모듈의 경우 관련 서비스 함수나 API 호출 시 매개변수를 통해 전달되거나(데이터 입력) 또는 서비스의 결과로 반환될 수 있다(데이터 출력). 제어 입력의 경우에는 서비스 함수나 API 호출 자체를 의미하며, 제어 출력은 하나의 서비스에서 추가로 다른 서비스를 수행하는 경우에 발생할 수 있다. 상태 출력은 표시기에 해당하며, 서비스의 수행 결과에 대한 정보를 반환한다.

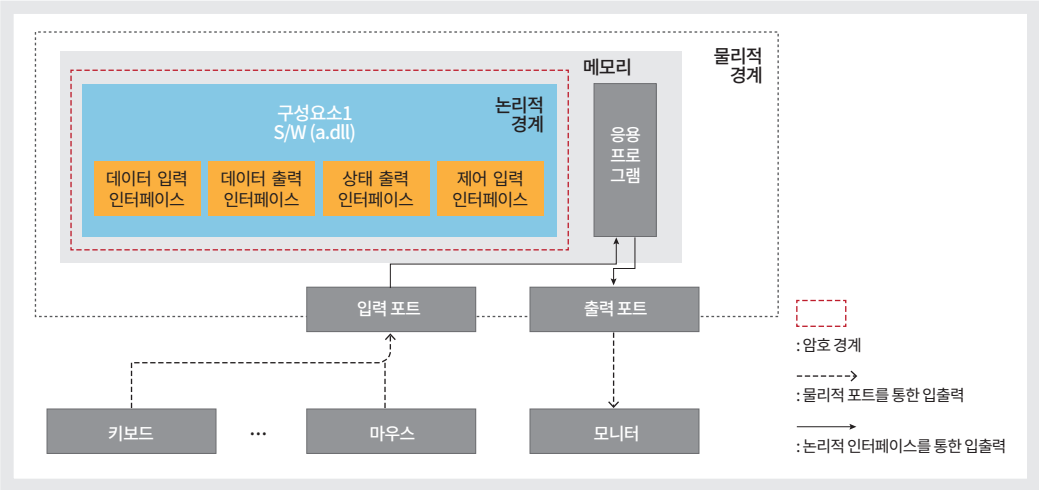
- 모든 논리적 정보 흐름이 암호경계의 입출구로 식별되는 물리적 접근 지점과 논리적 인터페이스로 제한되어야 함.
- 암호모듈의 모든 물리적 포트와 논리적 인터페이스 명세(함수 또는 API)
- 암호모듈의 모든 정보의 흐름 및 물리적 접근 지점 명세
- 동작 전 자가시험, 제로화, 오류상태, 수동 SSP 주입, 소프트웨어/펌웨어 로드 시 데이터 출력 금지
- 오류상태 시 제어 출력 인터페이스를 통한 모든 제어 출력 금지
- 정보 간 물리적·논리적 분리 방법 제시
- 설계서에 명세된 정보의 흐름과 실제 구현물과의 일치

이 장에서는 암호모듈 인터페이스의 주요 보안요구사항을 만족시키기 위해 개발업체가 작성해야 하는 AS 항목별 VE에 대한 작성 가이드를 제시한다.

나. 기본사항

항목	AS03.01
보안요구사항 개요	암호경계의 물리적 접근 지점과 인터페이스를 통한 정보의 흐름 명세
VE03.01.01	물리적 포트와 논리적 인터페이스 명세
VE03.01.02	암호모듈 정보의 흐름과 물리적 접근 지점 명세
VE03.01.03	논리적 인터페이스가 어떤 물리적 입력 또는 출력 포트에 속하는지 명세
작성 예시	

KISACrypto V1.0의 논리적 인터페이스는 제어 입력, 데이터 입력, 데이터 출력, 상태 출력 인터페이스로 구성된다. 응용 프로그램은 논리적 인터페이스를 통하여 암호모듈의 서비스를 이용할 수 있다. 예를 들어 암호모듈이 제공하는 서비스에 대해서 제어 입력 인터페이스를 통해 관련 API를 호출할 수 있으며, 데이터 입력 인터페이스를 통해 서비스와 연관된 데이터를 전달하여 처리할 수 있다. 예를 들어 암호화와 관련된 API를 호출할 때, API 호출 자체는 제어 입력 인터페이스이고, 전달되는 평문과 암호키는 데이터 입력 인터페이스에 해당한다. API 호출의 결과로 반환되는 암호문은 데이터 출력 인터페이스에 해당하고, 성공·실패 여부에 대한 상태 코드는 상태 출력 인터페이스에 해당한다.



[논리적 인터페이스에 대한 물리적 포트 접점]

위의 그림은 이 암호모듈이 사용하는 논리적 인터페이스에 대한 물리적 접근 포트를 명시한 것이다. KISACrypto V1.0 소프트웨어 암호모듈이기 때문에 논리적 인터페이스가 물리적 포트와 직접 만나는 지점이 없으며, 암호모듈을 이용하는 응용 프로그램을 통해 물리적 포트와 만난다.

아래 표는 이 암호모듈에서 사용하는 논리적 인터페이스와 물리적 포트에 대한 상세한 설명이다. 또한 동일한 물리적 포트를 사용하는 경우 논리적 인터페이스가 분리되는 것의 근거를 제시한다.

인터페이스	논리적 인터페이스	물리적 포트
데이터 입력	* 라이브러리의 API 호출 시 모듈로 전달되는 데이터 * 암호 관리자나 사용자에 의해 암호모듈에 입력되어 처리되는 모든 함수의 입력 파라미터 * 제어 입력 인터페이스와 동일한 물리적 포트를 사용하나, 제어 입력 인터페이스를 통하여 입력되는 제어 데이터는 API 함수 호출의 형태이므로 논리적으로 명백히 분리됨.	• 네트워크 포트 • 관리자 PC의 키보드 • 시리얼 • USB
데이터 출력	* 라이브러리의 API 호출 시 모듈로 출력되는 값 * 인자나 반환값으로, 모든 함수의 출력 파라미터 * 상태 출력 인터페이스를 통해 출력되는 상태 데이터는 API에 정의된 성공·실패 형태로만 존재하므로 데이터 출력 인터페이스가 동일한 물리적 포트를 사용한다고 하더라도 논리적으로 명백히 분리됨.	• 관리자 PC의 모니터
제어 입력	* 라이브러리의 API 호출 시 암호모듈 시작/종료, 암호 운영을 위하여 전달되는 값 * 암호모듈의 동작을 제어하기 위해 암호 관리자나 사용자에 의해 모듈로 입력되는 모든 API 함수 호출 입력	• 네트워크 포트 • 관리자 PC의 키보드 • 시리얼 • USB
제어 출력	해당사항 없음.	해당사항 없음.
상태 출력	* 라이브러리의 API 호출 시 반환되는 값 * 암호 관리자나 사용자에게 반환되는 상태 정보로, API에 정의된 성공·실패 형태로만 존재	• 관리자 PC의 모니터

[암호모듈의 논리적 인터페이스 및 물리적 포트 접점 상세 설명]

해설

위와 같이 암호모듈에 입력되는 각 물리적·논리적 입력이나 암호모듈에서 출력되는 각 물리적·논리적 출력에 대해 물리적 입출력 포트 및 물리적 입출력에 해당하는 논리적 인터페이스를 명세해야 한다.

다. 인터페이스 유형과 정의

항목	AS03.04, AS03.05, AS03.06, AS03.08
보안요구사항 개요	암호모듈의 논리적 인터페이스(데이터 입/출력, 제어 입력) 명세
VE03.04.01	암호모듈이 제공하는 서비스에 대한 논리적 인터페이스 명세
VE03.05.01	데이터 입력 인터페이스를 통해 입력되는 모든 데이터 명세
VE03.06.01	데이터 출력 인터페이스를 통해 출력되는 모든 데이터 명세
VE03.08.01	제어 입력 인터페이스를 통해 입력되는 제어 데이터 명세

작성 예시

서비스 분류	API 명칭	동작 내용	입력	출력
초기화	KISA_Crypto_initialize()	암호모듈 초기화	-	오류코드 (성공/실패)
상태확인	KISA_Crypto_getState()	암호모듈 현재 상태확인	-	암호모듈 상태값
버전확인	KISA_Crypto_getVersion()	암호모듈 이름 및 버전 확인	-	암호모듈 버전값
자가시험	KISA_Crypto_selftest()	(주기적) 자가시험 (암호 알고리즘 시험, 소프트웨어 무결성 시험)	-	오류코드 (성공/실패)
블록암호	KISA_Crypto_encryptInit(); KISA_Crypto_encryptUpdate(); KISA_Crypto_encryptFinal(); KISA_Crypto_encrypt();	블록암호 암호화	블록암호 설정정보, 평문, 키, IV	암호문, 오류코드
	KISA_Crypto_decryptInit(); KISA_Crypto_decryptUpdate(); KISA_Crypto_decryptFinal(); KISA_Crypto_decrypt();	블록암호 복호화	블록암호 설정정보, 암호문, 키, IV	평문, 오류코드
해시함수	KISA_Crypto_hashInit(); KISA_Crypto_hashUpdate(); KISA_Crypto_hashFinal(); KISA_Crypto_hash();	해시함수 수행	해시함수 설정정보, 메시지	해시값, 오류코드
메시지 인증코드	KISA_Crypto_hmacInit(); KISA_Crypto_hmacUpdate(); KISA_Crypto_hmacFinal(); KISA_Crypto_hmac();	HMAC 연산 수행	HMAC 설정정보, 메시지, MAC 키	HMAC값, 오류코드
난수 발생기	KISA_Crypto_drbgInit(); KISA_Crypto_drbgReseed(); KISA_Crypto_drbgGenerate(); KISA_Crypto_drbgClose(); KISA_Crypto_drbgRand();	난수 생성	난수발생기 설정정보, 엔트로피, 난수출력길이	난수비트열, 오류코드

서비스 분류	API 명칭	동작 내용	입력	출력
공개키 암호	KISA_Crypto_genKeyPair();	키쌍 생성	키 길이	공개키 및 개인키, 파라미터 오류코드
	KISA_Crypto_publicEncrypt();	공개키 암호 암호화	평문, 공개키 파라미터	암호문, 오류코드
	KISA_Crypto_privateDecrypt();	공개키 암호 복호화	암호문, 개인키 파라미터	평문, 오류코드
전자서명	KISA_Crypto_genKeyPair();	키쌍 생성	키 길이	공개키 및 개인키, 파라미터 오류코드
	KISA_Crypto_privateSign();	서명키로 서명 생성	메시지, 개인키 파라미터	서명, 오류코드
	KISA_Crypto_publicVerify();	검증키로 서명 검증	서명, 메시지, 공개키 파라미터	오류코드
키설정	KISA_Crypto_DH_ genParameter();	도메인 변수 생성	키길이	도메인 파라미터, 오류코드
	KISA_Crypto_DH_ genKeyPair();	키쌍 생성	도메인 파라미터	공개키 및 개인키, 파라미터 오류코드
	KISA_Crypto_DH_ computeSharedSecret();	키 설정	도메인, 개인키 및 공개키 파라미터	공유키, 오류코드

[암호모듈에서 제공하는 암호 서비스 목록]

구분	설명
함수	int KISA_Crypto_encryptUpdate (u8 *output, u32 outputLength, KISA_Block_CTX *ctx, const u8 *input, const u32 inputLength);
설명	– 블록암호모듈의 암호화를 수행하기 위한 중간 단계 수행 – 평문을 입력받아 운용모드에 따라 암호화를 수행하고, 암호문/평문 출력
데이터 입력	– KISA_Block_CTX *ctx: 블록암호에 대한 컨텍스트 구조체(암호화를 위한 정보 포함) – const u8 *input: 평문 – const u32 inputLength : 평문 길이
데이터 출력	– KISA_Block_CTX *ctx: 블록암호에 대한 컨텍스트 구조체(암호화를 위한 정보 포함) – u8 *output, : 암호문 – u32 outputLength, : 암호문 길이
제어 입력	API 호출
상태 출력	– 성공 · CIPHER_CipherProcess_SUCCESS – 실패 · CIPHER_ERROR_INVALID_INPUT (올바르지 않은 입력) · CIPHER_ERROR_UNKNOWN_ID (잘못된 암호알고리즘)
제어 출력	없음

[암호 서비스에 대한 상세 논리적 인터페이스 명세]

해설
AS03.04는 AS03.05(데이터 입력), AS03.06(데이터 출력), AS03.08(제어 입력), AS03.11(상태 출력)과 연관되어 있다. 따라서 AS03.04 항목별 VE 항목 시 AS03.05, AS03.06, AS03.08, AS03.11의 VE 항목의 내용까지 모두 포함하여 기본 및 상세설계서에 명세하는 것이 가능하다. 위의 작성 예시는 소프트웨어 암호모듈이 제공하는 암호 서비스에 대해 데이터 입력, 데이터 출력, 제어 입력, 상태 출력 관점에서 논리적 인터페이스를 정의한 것이다. 소프트웨어 암호모듈의 경우 위의 예시와 같이 암호모듈에서 제공하는 암호 서비스를 목록 형태로 간단히 제시한 후, 각 암호 서비스의 논리적 인터페이스에 대해 상세한 명세를 하는 것이 가능하다. 각 암호 서비스에 대해 5개의 논리적 인터페이스를 상세히 명세함으로써 AS03.04, AS03.05, AS03.06, AS03.08, AS03.11 보안요구사항을 만족시킬 수 있다. AS03.11에서 요구하는 상태 출력에 대해서는 암호 서비스별로 명세된 상태 출력을 종합하여 추가로 하나의 표 형태로 제시하는 것을 권고한다. 이 때 오류코드에 대해서는 단순한 오류와 심각한 오류로 나누는 것이 필요하다.

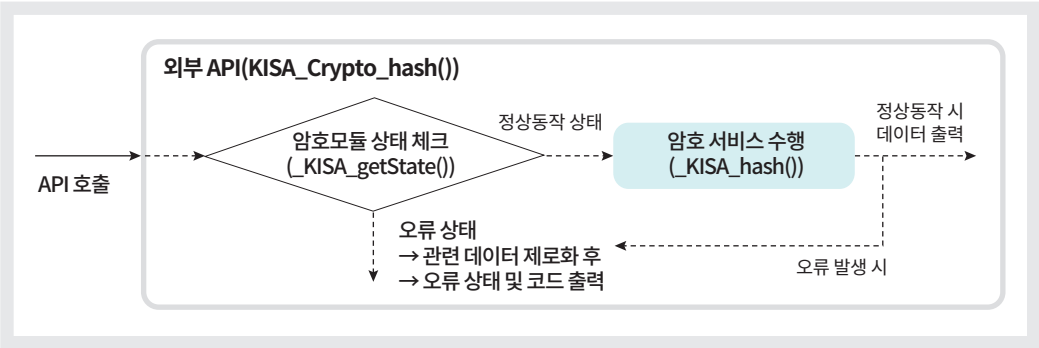
항목	AS03.07
보안요구사항 개요	암호모듈 오류, 동작 전 자가시험, 제로화, 수동 키 주입 등의 상태에서 데이터 출력 금지 방법 필요
VE03.07.01	데이터 출력 금지 방법과 이를 보장하는 방법에 대한 명세
VE03.07.02	

작성 예시

KISACrypto V1.0은 동작 전 자가시험을 포함한 자가시험 상태, 오류 상태, 제로화 상태에서 데이터 출력을 금지하는 방법을 적용하고 있다. 상태별 데이터 출력 금지 방법은 다음과 같다.

관련 API	데이터 출력 금지 메커니즘
자가시험	– 자가시험은 KISA_Crypto_selftest()를 이용하여 수행되며, 해당 API는 데이터 입력, 데이터 출력 인터페이스가 존재하지 않음(함수 인자가 존재하지 않음). – 또한 해당 API에서는 화면 입출력 함수가 사용되지 않았으며, 자가시험의 성공·실패를 상태 출력을 통해 반환함.
암호 서비스	– 암호모듈에서 제공하는 모든 암호 서비스 API 수행 전, 암호모듈의 내부 API를 통하여 암호모듈의 상태를 체크하여 정상동작 상태에서만 서비스 수행하도록 함(오류 상태에서는 서비스를 수행하지 않음). – 암호 서비스 수행 시 오류가 발생한 경우, API 내부에서 생성된 데이터를 모두 제로화한 후, 오류코드만을 상태 출력을 통해 반환함.
제로화	암호모듈은 명시적인 제로화 상태를 가지지는 않으나, 암호모듈의 서비스는 응용 프로그램에 의해 호출되어 순차적으로 동작하기 때문에 제로화 API가 호출되는 시험에 데이터 출력은 발생하지 않음.

다음은 암호모듈에서 제공하는 암호 서비스 API 내부에서 암호모듈의 현재 상태를 확인하여 오류 상태인 경우에는 암호 서비스 및 데이터 출력을 금지하는 방법을 명세한다. 암호모듈 상태를 확인할 수 있는 내부 함수(_KISA_getState())를 이용하여 암호모듈의 상태를 확인한 후, 정상동작 상태에서만 암호 서비스를 수행한다. 오류 상태일 경우에는 즉각 오류코드를 반환하고 종료한다. 정상동작 상태에서 암호 서비스를 수행할 때 오류가 발생했다면 서비스 수행 시에 발생한 내부 데이터를 모두 제로화한 후에 오류코드를 반환한다.



[암호 서비스 수행 시 오류 상태에 대한 데이터 출력 금지 방법 명세]

해설

암호모듈의 보안과 관련된 특정 상태에서는 암호모듈의 안전성에 영향을 미칠 수 있는 데이터 출력이 금지되어야 한다. 보안과 관련된 상태는 오류 상태, 동작 전 자가시험 상태, 제로화 상태, 수동 키 주입 상태, 소프트웨어/펌웨어 로드 상태이다. 단순한 형태의 소프트웨어 암호모듈인 경우에는 최소한 오류 상태, 동작 전 자가시험 상태, 제로화 상태에서 데이터 출력 금지 방법이 적용되어야 하며, 어떤 방법을 통하여 데이터 출력을 금지하는지를 기본 및 상세설계서에 명세해야 한다.

항목	AS03.11
보안요구사항 개요	상태 출력 인터페이스 명세
VE03.11.01	논리적·물리적 표시기를 포함하는 상태 출력 인터페이스 명세
VE03.11.02	가능하다면 상태 출력을 수행하는 외부 장치 명세 필요

작성 예시

KISACrypto V1.0 암호모듈에서 정의된 상태 출력은 다음 표와 같다. 오류코드 중 단순한 오류는 사용자의 실수로 인해 발생하는 오류로서 암호 서비스에서 오류코드를 반환한 후 암호모듈은 정상동작 상태로 자동천이 된다. 반면, 심각한 오류의 경우(동작 전 자가시험 실패, 조건부 자가시험 실패 등)에는 암호모듈이 오류 상태로 유지되어 이후의 서비스 요청에 대해 정상적으로 서비스를 수행할 수 없다. 따라서 심각한 오류 발생 시에는 리부팅 또는 암호모듈 재설치를 통해 오류를 해결해야 한다. 또한 정상동작 이후 사용한 중요 보안매개변수, 관련 변수를 제로화해야 한다.

상태 변수	설명	값
STATUS_LOAD	초기화 전	0x000000
STATUS_INITIALIZE	초기화	0x000001
STATUS_SELFTEST	자가시험	0x000002
STATUS_APPROVED	검증대상 동작모드 상태	0x000003
STATUS_NON_APPROVED	비검증대상 동작모드 상태	0x000004
STATUS_ERROR	심각한 오류 상태	0x000005
STATUS_TERMINATE	종료 상태	0x000006

[암호모듈 상태 목록]

분류	설명	값
성공	서비스 성공	0x000000

[서비스별 상태 출력 코드 명세 및 분류(성공)]

서비스	오류명칭	오류코드	오류설명	오류구분
블록 암호	BLOCK_INVALID_KEY_LENGTH_ERROR	0x100001	블록암호에서 키 길이 오류	단순한 오류
	BLOCK_INVALID_CIPHER_ID_ERROR	0x100002	지원하지 않는 블록암호ID	단순한 오류
	BLOCK_PADDING_ERROR	0x100003	패딩 오류	단순한 오류
	BLOCK_INVALID_INPUT_ERROR	0x100004	잘못된 입력 오류	단순한 오류
	BLOCK_SELFTEST_FAIL	0x100005	블록암호 KAT 자가시험 실패	심각한 오류
공개키 암호	PUBLIC_INVALID_KEY_LENGTH_ERROR	0x200001	공개키 암호에서 키 길이 오류	단순한 오류
	PUBLIC_INVALID_CIPHER_ID_ERROR	0x200002	지원하지 않는 공개키 암호 ID	단순한 오류
	PUBLIC_PADDING_ERROR	0x200003	공개키 암호 패딩 오류	단순한 오류
	PUBLIC_INVALID_INPUT_ERROR	0x200004	잘못된 입력 오류	단순한 오류
	PUBLIC_SELFTEST_FAIL	0x200005	공개키 암호 KAT 자가시험 실패	심각한 오류
	PUBLIC_COND_SELFTEST_FAIL	0x200006	공개키 암호 조건부 자가시험 실패	심각한 오류

[서비스별 상태 출력 코드 명세 및 분류(실패)]

해설
상태 정보 확인 API의 출력이나 서비스 API의 반환값(성공, 오류코드)을 통해 상태 출력을 한다. AS03.04에서 상태 출력을 포함한 논리적 인터페이스를 명세하였으나, 이 시험항목에서는 서비스별로 명세된 상태 출력을 종합하여 하나의 표로 나타내는 것을 권고한다. 서비스별 상태 출력(성공/오류)를 명세하되, 오류의 경우에는 사용자의 사용상 잘못으로 인한 단순한 오류인지, 암호모듈의 정상동작에 영향을 미칠 수 있는 심각한 오류인지를 나타내는 것이 바람직하다. 위에서는 예시를 위해 블록암호와 공개키 암호에 대한 상태 출력만을 제시한다. 더 많은 종류의 암호 서비스를 제공하는 경우 관련된 상태 출력에 대해서 위와 동일하게 명세 가능하다.

항목	AS03.15
보안요구사항 개요	가변 길이 입력에 대한 제한을 포함한 입력 데이터와 제어 정보에 대한 형식 정의 필요
VE03.15.01	암호모듈에 데이터를 입력하기 위해 사용된 물리적·논리적 경로 명세
VE03.15.02	
VE03.15.03	가변 길이에 대한 제한 방법과 입력 데이터 형식 정의 명세
VE03.15.04	암호모듈 내에서 형식검증을 수행하는 요소에 대한 명세

작성 예시

KISACrypto V1.0는 블록암호, 해시함수, 메시지인증코드, 난수발생기, 공개키 암호알고리즘에 대한 서비스를 제공한다. 암호알고리즘별 가변 길이에 대한 제한 및 검증방법은 아래 표와 같다. 검증 요소는 크게 알고리즘별로 허용된 키 사이즈에 대한 검증과 가변 길이 데이터(평문, 암호문 등)에 대한 검증을 포함한다.

알고리즘	검증 데이터	검증 방법
블록암호 (SEED-128)	비밀키/평문/암호문/라운드키 포인터	NULL인지 확인
	비밀키 비트 길이	128비트인지 확인
해시함수 (SHA-256)	SHA256 핸들/메시지/해시 포인터	NULL인지 확인
	메시지 길이	2^64-1 비트 범위 내인지 확인
메시지인증코드 (HMAC)	HMAC 핸들/비밀키/메시지/MAC 포인터	NULL 인지 확인
	키 길이	L과 같거나 L/2보다 큰지 확인 [L:해시함수 출력 길이(바이트)]
	메시지 길이	2^B-8B 바이트 범위 내인지 확인 [B:해시함수 입력의 블록 길이(바이트)]

알고리즘	검증 데이터	검증 방법
난수발생기 (Hash_DRBG)	엔트로피/논스/개별화 문자열/출력 난수열 포인터	NULL인지 확인
	출력 난수열 길이	2^19 비트 범위 내인지 확인
	엔트로피 최대 입력 길이	2^35 비트 범위 내인지 확인
	추가 입력 최대 길이	2^35 비트 범위 내인지 확인
	시드별 출력값 생성 횟수 (reseed_interval)	2^48 범위 내인지 확인
공개키 암호 (RSAES)	공개키 파라미터/개인키 파라미터/평문/ 암호문 포인터	NULL인지 확인
	키 비트 길이	2048 비트인지 확인
	사용되는 해시함수	SHA-256인지 확인
	평문 최대 길이(RSAES의 경우)	|M| ≤ ELen-2|hashLen|-2 (ELen:인코딩된 메시지 크기)

해설

검증대상 암호알고리즘 목록에 속하는 알고리즘은 다양한 옵션 설정에 기반하여 가변 길이의 입력을 받아 가공하여 출력한다. 옵션과 가변 길이의 입력에 대해 제한하지 않는다면 내부적으로 예측할 수 없는 동작을 유발할 가능성이 있으며, 이는 취약점으로 연결될 수 있다. 또한 암호알고리즘별로 안전성 관점에서 최대 입력 길이는 제한하는 경우도 존재한다.(암호알고리즘 표준에서 안전성을 위해 입력되는 길이의 최대 비트를 n으로 제한한 경우, 암호모듈에서는 n보다 더 짧은 길이를 이용할 수 있다). 이 시험항목에는 암호모듈이 제공하는 각 서비스에 대해 안전한 동작을 보장하기 위해 입력되는 데이터의 길이와 형식을 어떻게 제한했는지에 대해 명세해야 한다.

5 역할, 서비스 및 인증

가. 목적

암호모듈은 운영자에게 인가된 역할을 지원하고 각 역할에 맞는 서비스를 제공해야 한다. 하나의 운영자가 여러 개의 역할을 할당받을 수도 있다. 또한 암호모듈이 동시 운영자(concurrent operator)를 지원하는 경우 암호모듈은 내부적으로 각 운영자의 역할과 그 역할에 할당된 서비스를 분리하여 유지해야 한다. 암호모듈을 이용하고자 하는 운영자에 대해 암호모듈 내부의 인증 과정을 수행하며, 이 과정을 통해 운영자가 요구하는 역할과 그 역할에 할당된 서비스를 사용할 수 있는지 검증한다. 인증에 대한 요구사항은 보안수준에 따라 다음과 같다.

- 보안수준 1의 경우, 암호모듈이 자체적으로 인증 메커니즘을 탑재할 것을 강제하지 않음(이 경우에는 운영 환경에서 제공해 주는 인증 메커니즘을 사용할 것을 권고함).

암호모듈은 운영자에 대해 다음의 기본 서비스를 제공해야 한다.

- 버전 정보를 제공하는 서비스(암호모듈명, 버전 등)
- 상태를 표시하는 서비스(암호모듈에 대한 현재 상태, 서비스 수행 결과)
- 자가시험 수행 서비스(동작 전 자가시험에 대한 수행)
- 승인 보안 서비스(검증대상 동작모드에서 최소 하나의 보안 서비스를 제공해야 함.)
- 제로화 서비스(중요 보안매개변수에 대한 제로화 수행)

또한 암호모듈은 선택적으로 운영자에게 우회 기능(bypass capability), 자가초기화된 암호출력 기능(self-initiated cryptographic output capability), 소프트웨어/펌웨어 로딩(software/firmware loading) 서비스를 제공할 수 있다.

이 장에서는 역할, 서비스 및 인증에 대한 보안요구사항을 만족시키기 위해 개발업체가 작성해야 하는 AS 항목별 VE에 대한 작성 가이드를 제시한다.

나. 기본사항

항목	AS04.02
보안요구사항 개요	동시 운영자의 지원 여부 및 방법 명세
VE04.02.01	동시 운영자 지원 여부, 지원 시 각 동시 운영자에 대한 역할과 서비스를 구분하여 유지하는 방법 명세
작성 예시	
KISACrypto V1.0은 소프트웨어 라이브러리 형태의 암호모듈로서 여러 응용에게 범용적인 암호 서비스를 제공하는 것을 목표로 한다. 또한 보안수준 1이기 때문에 자체적인 인증 메커니즘을 탑재하지 않고 운영환경에서 제공하는 메커니즘을 활용한다. 따라서 암호모듈 자체적으로는 동시 운영자 기능을 제공하지 않는다. 또한 내부적으로 암호모듈을 사용하는 운영자에 대한 정보를 관리하는 기능을 포함하지 않기 때문에 사용되는 암호 서비스에 따라 암호 관리자 역할과 사용자 역할을 논리적으로 지원한다.	
해설	
범용 소프트웨어 라이브러리 형태의 암호모듈의 경우 응용 프로그램에서 사용 가능한 암호 서비스를 제공하는 것이 목적이며, 암호모듈의 운영자와 관련된 정보를 유지하는 기능을 포함하기 어렵다. 따라서 범용 소프트웨어 라이브러리 형태의 암호모듈은 동시 운영자 지원이 어렵다고 할 수 있다. 그러나 라이브러리형 암호모듈을 이용하는 제품형 암호모듈의 경우에는 자체적인 인증 메커니즘을 탑재하여 인증을 통하여 운영자에게 정해진 역할과 그 역할에 대한 서비스를 사용하도록 하는 것이 가능하다. 따라서 제품형 암호모듈의 경우에는 동시 운영자 지원이 가능하다.	

항목	AS04.03
보안요구사항 개요	[KS X ISO/IEC 19790 부속서] A.2.4가 충족되는 개발문제 제출
VE04.03.01	A.2.4에 대한 개괄적인 만족 여부 명세
작성 예시	
KISACrypto V1.0은 보안수준 1 소프트웨어 라이브러리 암호모듈이다. 이 암호모듈은 암호모듈 관리자와 사용자를 지원하며, 각 운영자는 논리적으로 구분된다. 암호모듈은 검증대상 서비스만을 포함하고 있으며, 일반 서비스인 검증대상 암호알고리즘을 이용하는 보안 서비스, 버전 출력 서비스, 자가시험 수행 서비스, 상태 출력 서비스, 제로화 서비스를 제공하며, 우회기능, 자가초기화된 출력 기능, 소프트웨어/펌웨어 로딩과 같은 서비스는 제공하지 않는다.	
해설	
A.2.4에서는 암호모듈이 지원하는 모든 인가받은 역할, 검증대상 및 비검증대상 서비스, 그리고 각 서비스에 대한 입력, 출력, 사용 인가받은 역할 등에 대한 내용을 설계서에 명세할 것을 요구한다. 이 시험항목이 요구하는 사항에 대해 AS04.05, AS04.06, AS04.11, AS04.13, AS04.14, AS04.15 항목별 VE 항목을 기술하는 것으로 대체 가능하다.	

다. 역할

항목	AS04.05
보안요구사항 개요	암호모듈 관리자 역할에 대한 명세
VE04.05.01	암호모듈 관리자와 연관된 서비스 명세

작성 예시

KISACrypto V1.0은 암호 관리자와 일반 사용자 역할을 지원하며 논리적으로 구분된다. 일반 사용자는 암호모듈이 제공하는 일반 보안 서비스를 사용할 수 있으며, 관리자는 암호모듈의 설치, 초기화, 자가시험 수행, 종료 등의 관리적인 서비스도 사용 가능하다.

구분	인가된 역할
일반 사용자	- 암호모듈에서 제공하는 보안 서비스 수행
암호 관리자	- 암호모듈 설치, 초기화, 자가시험 수행, 종료, 삭제 - 암호모듈에서 제공하는 보안 서비스 수행

해설

암호모듈은 기본적으로 암호 관리자에 대한 역할을 포함해야 한다. 암호모듈 관리자는 암호모듈 사용자가 사용할 수 있는 서비스에 추가로 암호모듈에 대한 설치, 초기화, 중요 보안매개변수에 대한 관리, 종료, 삭제 등의 서비스를 수행할 수 있다. 암호모듈은 암호 관리자 역할과 사용자 역할을 명시적으로만 구분한다.

항목	AS04.06
보안요구사항 개요	암호모듈 사용자 역할에 대한 명세
VE04.06.01	암호모듈 사용자와 연관된 서비스 명세

작성 예시

KISACrypto V1.0은 일반 사용자 역할을 지원한다. 암호 사용자는 암호모듈에서 제공하는 기본 보안 서비스를 사용할 수 있다. KISACrypto V1.0이 일반 사용자에게 제공하는 일반 보안 서비스는 블록암호, 해시함수, 메시지인증코드, 난수발생기, 공개키 암호이다. 또한 암호모듈에 대한 현재 상태 확인, 버전 정보 출력도 일반 보안 서비스에 포함된다.

해설

AS04.05와 AS04.06에서는 암호모듈 관리자와 일반 사용자 역할에 대한 개괄적인 서비스 명세를 요구한다. 암호모듈이 제공하는 서비스에 대한 명확한 명세는 서비스를 수행하는 역할과 인터페이스(제어 입력, 데이터 입력, 데이터 출력, 상태 출력)와 연관하여 명세하는 것이 필요하다. 해당 부분은 AS04.11에서 명세되어야 한다.

라. 서비스

항목	AS04.11
보안요구사항 개요	서비스별 입력과 출력 명세
VE04.11.01	서비스별 입력과 출력, 인가된 역할 명세

작성 예시

KISACrypto V1.0 암호모듈이 제공하는 서비스의 목적, 기능, 연관된 API, 입력, 출력, 인가된 역할에 대한 정보는 아래와 같다. 서비스와 연관된 상세한 입력과 출력은 암호모듈 인터페이스(VE03.04.01)에서 확인 가능하다.

서비스 분류	API 명칭	목적/기능	입력	출력	인가된 역할
초기화	KISA_Crypto_initialize()	암호모듈 초기화	-	오류코드 (성공/실패)	관리자
상태확인	KISA_Crypto_getState()	암호모듈 현재 상태 확인	-	암호모듈 상태값	관리자/ 사용자
버전확인	KISA_Crypto_getVersion()	암호모듈 이름 및 버전 확인	-	암호모듈 버전값	관리자/ 사용자
자가시험	KISA_Crypto_selftest()	(주기적) 자가시험 (암호 알고리즘 시험, 소프트웨어 무결성 시험)	-	오류코드 (성공/실패)	관리자
블록암호	KISA_Crypto_encryptInit(); KISA_Crypto_encryptUpdate(); KISA_Crypto_encryptFinal(); KISA_Crypto_encrypt();	블록암호 암호화	블록암호 설정 정보, 평문, 키, IV	암호문, 오류코드	관리자/ 사용자
	KISA_Crypto_decryptInit(); KISA_Crypto_decryptUpdate(); KISA_Crypto_decryptFinal(); KISA_Crypto_decrypt();	블록암호 복호화	블록암호 설정 정보, 암호문, 키, IV	평문, 오류코드	
해시함수	KISA_Crypto_hashInit(); KISA_Crypto_hashUpdate(); KISA_Crypto_hashFinal(); KISA_Crypto_hash();	해시함수 수행	해시함수 설정 정보, 메시지	해시값, 오류코드	관리자/ 사용자

서비스 분류	API 명칭	목적/기능	입력	출력	인가된 역할
메시지 인증코드	KISA_Crypto_hmacInit(); KISA_Crypto_hmacUpdate(); KISA_Crypto_hmacFinal(); KISA_Crypto_hmac();	HMAC 연산 수행	HMAC 설정 정보, 메시지, MAC 키	HMAC값, 오류코드	관리자/ 사용자
난수 발생기	KISA_Crypto_drbgInit(); KISA_Crypto_drbgReseed(); KISA_Crypto_drbgGenerate(); KISA_Crypto_drbgClose(); KISA_Crypto_drbgRand();	난수 생성	난수발생기 설정 정보, 엔트로피, 난수출력 길이	난수비트열, 오류코드	관리자/ 사용자
공개키 암호	KISA_Crypto_genKeyPair();	키쌍 생성	키 길이	공개키 및 개인키 파라미터	관리자/ 사용자
	KISA_Crypto_publicEncrypt();	공개키 암호 암호화	평문, 공개키 파라미터	암호문	
	KISA_Crypto_privateDecrypt();	공개키 암호 복호화	암호문, 개인키 파라미터	평문	
전자서명	KISA_Crypto_genKeyPair();	키쌍 생성	키 길이	공개키 및 개인키 파라미터	관리자/ 사용자
	KISA_Crypto_privateSign();	서명키로 서명 생성	메시지, 개인키 파라미터	서명	
	KISA_Crypto_publicVerify();	검증키로 서명 검증	서명, 메시지, 공개키 파라미터	오류코드	
키 설정	KISA_Crypto_DH_genParameter();	도메인 변수 생성	키길이	도메인 파라미터, 오류코드	관리자/ 사용자
	KISA_Crypto_genKeyPair();	키쌍 생성	키길이	공개키 및 개인키 파라미터, 오류코드	
	KISA_Crypto_compute_Key();	키 설정	도메인, 개인키 및 공개키 파라미터	공유키, 오류코드	

[암호 서비스 목록에 대한 명세]

해설

AS04.11은 암호모듈이 제공하는 각 서비스에 대해 목적, 기능, 입력·출력, 인가된 역할을 명세할 것을 요구한다. 따라서 암호모듈이 제공하는 서비스에 대해 암호모듈 인터페이스의 시험항목인 AS03.04와 연관하여 서술하는 것이 바람직하다.

항목	AS04.13
보안요구사항 개요	버전 출력 서비스 명세
VE04.13.01	암호모듈 명칭과 버전 정보를 출력하는 서비스 명세

작성 예시

KISACrypto V1.0은 KISA_Crypto_getVersion() API를 통해 아래와 같이 암호모듈의 명칭과 버전 정보를 출력하는 서비스를 제공한다.

```
KCMVP_Initialize()
KISACryptoV1.0
Developed by KISA
```

[KISA_Crypto_getVersion() 호출을 통한 암호모듈 명칭과 버전 확인]

해설

AS04.13에서 요구하는 버전 출력 서비스 수행 및 동작에 대한 결과는 암호모듈 시험서에서도 포함한다.

항목	AS04.14
보안요구사항 개요	상태표시 서비스 명세
VE04.14.01	암호모듈의 상태와 서비스 수행에 대한 결과를 출력하는 서비스 명세

작성 예시

KISACrypto V1.0은 KISA_Crypto_getState() API를 통해 암호모듈의 현재 상태를 출력 가능하다. 또한 암호 서비스 API를 수행하면 API로부터 반환되는 값을 통하여 서비스의 성공/실패 여부를 확인할 수 있다.
KISACrypto V1.0은 암호모듈의 동작 중에 가질 수 있는 내부상태를 다음과 같이 정의하였다. 암호모듈의 내부 상태를 관리하는 변수는 암호모듈 동작 중 아래에 정의되어 있는 상태값 중 하나를 가진다.

```
enum KCMVP_CRYPTO_MODULE_STATE
{
    KCMVP_CM_LOAD = KCMVP_CM_MODULE_STATE_BASE, // 암호모듈 로드(기본 상태)
    KCMVP_CM_PRE_SELFTEST,                      // 동작 전 자가시험
    KCMVP_CM_NORMAL,                            // 정상동작 상태(검증대상 동작모드)
    KCMVP_CM_COND_SELFTEST,                     // 조건부 자가시험 상태
    KCMVP_CM_NORMAL_ERROR,                      // 단순 오류 상태
    KCMVP_CM_CRITICAL_ERROR,                    // 심각한 오류 상태
    KCMVP_CM_EXIT                               // 암호모듈 종료
};
```

[암호모듈 상태 정의]

다음은 KISA_Crypto_getState() API를 이용하여 암호모듈의 상태를 출력한 결과이다. 암호모듈 초기화를 수행하기 전에는 암호모듈의 상태가 0x000003E8(암호모듈 로드)이었으며, 암호모듈 초기화를 수행한 후 0x000003EA(정상동작 상태)임을 확인할 수 있다.

```
암호모듈 상태: 000003E8
KCMVP_Initialize()
KCMVP 암호모듈 초기화 결과: 1
암호모듈 상태: 000003EA
```

[암호모듈 초기화를 통한 상태 변화 확인(성공의 경우)]

암호모듈 초기화가 실패한 경우(동작 전 자가시험 실패), 다음과 같이 암호모듈의 상태가 0x000003ED(심각한 오류 상태)임을 확인할 수 있다.

```
암호모듈 상태: 000003E8
KCMVP_Initialize()
KCMVP 암호모듈 초기화 결과: -1
암호모듈 상태: 000003ED
```

[암호모듈 초기화를 통한 상태 변화 확인(실패의 경우)]

해설

AS04.14에서 요구하는 상태 출력 서비스 수행 및 동작에 대한 결과는 암호모듈 시험서에도 포함한다.

항목	AS04.15
보안요구사항 개요	동작 전 자가시험 서비스 명세
VE04.15.01	사용자가 호출할 수 있는 자가시험 서비스 명세

작성 예시

KISACrypto V1.0은 KISA_Crypto_integritytest()와 KISA_Crypto_selftest() API를 통하여 사용자가 동작 전 자가시험을 수행할 수 있도록 한다. 암호모듈이 초기화될 때, 해당 API는 자동으로 호출되어 동작 전 자가시험을 수행한다. 동작 전 자가시험은 동작 전 소프트웨어 무결성 시험과 동작 전 핵심 기능 시험으로 구성된다. 동작 전 소프트웨어 무결성 시험에서는 암호모듈이 변조되지 않았는지의 여부를 검사하고, 동작 전 핵심 기능 시험에서는 암호모듈에 탑재되어 있는 검증대상 암호알고리즘에 대해 KAT(Known Answer Test) 형태로 정상동작 여부를 검사한다. 동작 전 자가시험이 성공적으로 끝나면 암호모듈은 사용자에게 서비스를 제공할 수 있는 검증대상 동작모드로 자동천이된다. 만약 실패하면 암호모듈은 심각한 오류 상태로 천이되어 서비스를 제공할 수 없다. 이 경우, 암호모듈 관리자는 암호모듈을 재시작하거나 재설치를 통해 암호모듈의 기능을 정상적으로 복구해야 한다.
다음은 암호모듈에서 수행되는 암호모듈 초기화 과정에서 동작 전 자가시험의 수행 결과를 출력한 것이다(편의를 위해 출력함수를 이용하여 시험 수행의 결과를 출력하도록 하였음). 동작 전 무결성 시험을 수행하기 전에 무결성 검증에 사용되는 HMAC 알고리즘에 대한 KAT 시험을 수행하였다. 또한 내부적으로 탑재된 SEED, Hash_DRBG에 대한 KAT 시험을 수행하였다. 결과값으로 반환된 '1'은 성공을 의미한다. 동작 전 자가시험을 성공적으로 완료한 후, 암호모듈은 검증대상 동작모드로 천이됨을 확인할 수 있다.

```
암호모듈 상태: 000003E8
KCMVP_Initialize()
[HMAC KAT Test] 1
[Software Integrity Test] 1
[ARIA KAT Test] 1
[DRBG KAT Test] 1
KCMVP 암호모듈 초기화 결과: 1
암호모듈 상태: 000003EA
```

[초기화 시 수행되는 동작 전 자가시험 수행 결과 출력]

또한 암호모듈이 검증대상 동작모드 상태에도 사용자가 필요 시에 KISA_Crypto_integritytest()와 KISA_Crypto_selftest()를 호출하여 동작 전 자가시험을 아래와 같이 호출할 수 있다.

[사용자에 의해 호출된 자가시험 수행 결과 출력]

해설

AS04.15에서 요구하는 상태 출력 서비스 수행 및 동작에 대한 결과는 암호모듈 시험서에 포함한다. 위의 예시는 HMAC, SEED, DRBG에 대한 KAT를 수행하는 동작 전 핵심기능 시험(예시를 위해 몇 개의 알고리즘만을 포함함.)과 동작 전 소프트웨어 무결성 시험을 수행한 결과를 보인다. 암호모듈에 탑재된 검증대상 암호알고리즘은 사용되기 전에 조건부 암호알고리즘 시험을 수행해야 하며, 이는 동작 전 핵심기능 시험에서 일괄적으로 KAT 시험 형태로 수행될 수 있다.

항목	AS04.16
보안요구사항 개요	검증대상 동작모드에서 제공하는 암호 서비스에 대한 동작 명세

작성 예시

KISACrypto V1.0은 블록암호, 해시함수, 메시지인증코드, 난수발생기, 공개키 암호에 대한 암호 서비스를 제공한다. 아래는 블록암호 SEED에 대한 동작 시험 수행 결과를 보인다. 각 알고리즘에 대한 상세한 동작 시험 결과는 시험서에 명세되어 있다.

[블록암호 SEED-128에 대한 암복호화 시험 결과 출력]

해설

AS04.16에서 요구하는 검증대상 동작모드에서 제공하는 암호 서비스에 대한 상세한 동작 시험은 시험서에서 수행한다.

항목	AS04.17
보안요구사항 개요	검증대상 동작모드에서 제공하는 제로화 서비스에 대한 동작 명세

작성 예시

KISACrypto V1.0은 사용이 완료된 중요 보안매개변수에 대해 사용 후, 메모리상에서 안전하게 삭제할 수 있는 제로화 API인 KISA_Crypto_Zeroize()를 제공한다. 아래는 128-bit 블록암호키에 대해 제로화 API를 호출하였을 때 메모리상에서 기존값이 제로화된 것을 보인 것이다.

```
[암호키]
 [16byte] :
 29 23 BE 84 E1 6C D6 AE 52 90 49 F1 F1 BB E9 EB

[제로화 수행]

[암호키]
 [16byte] :
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

[블록암호 키에 대한 제로화 수행 결과 출력]

해설

AS04.17에서 요구하는 제로화 서비스에 대한 상세한 동작 시험은 시험서에서 수행한다.

마. 인증

항목	AS04.43
보안요구사항 개요	리셋, 재부팅, 전원 재인가 후 암호모듈의 운영자 인증 방법 명세
VE04.43.01	암호모듈의 전원 종료 시 이전 인증을 소멸하는 방법 명세

작성 예시

KISACrypto V1.0은 보안수준 1 소프트웨어 라이브러리 형태의 암호모듈이기 때문에 자체적인 인증 메커니즘을 포함하지 않는다. 즉 암호모듈은 자체적인 인증방법을 제공하지 않으며, 운영체제가 제공하는 인증에 의존하기 때문에 운영체제 인증이 초기화되기 이전에 이 모듈에 접근하는 것은 불가능하다. 이 암호모듈은 수행되는 서비스에 따라 논리적으로 분리되며, 그 역할에 따라 암묵적으로 인증된다. 이 때 운영체제의 인증에 실패한 암호 관리자는 이 암호모듈의 서비스를 수행할 수 없다. 이 암호모듈이 동작하는 범용 운영환경(윈도우, 리눅스)에서는 사용자가 로그오프를 하거나 운영체제 종료 시 이전에 인증된 정보를 적절히 제거하는 기능이 탑재되어 있다.

해설

라이브러리 형태의 암호모듈의 경우에는 운영환경의 인증방법에 의존적이기 때문에 위와 같이 명세가 가능하다. 그러나 제품형 암호모듈의 경우에는 자체적인 인증 메커니즘을 포함할 수 있기 때문에 암호모듈을 종료할 때 암호모듈에 의해 메모리상에서 관리되던 사용자의 인증 정보를 적절히 제로화해야 한다.

항목	AS04.44
보안요구사항 개요	암호모듈 내의 인증 데이터에 대한 보호
VE04.44.01	암호모듈 내의 모든 인증 데이터를 보호하는 방법 명세
작성 예시	
KISACrypto V1.0은 보안수준 1 소프트웨어 라이브러리 형태의 암호모듈이기 때문에 자체적인 인증 메커니즘을 포함하지 않는다. 따라서 내부적으로 사용자의 인증 데이터를 저장하는 메모리를 포함하지 않는다.	
해설	
라이브러리 형태의 암호모듈의 경우에는 운영환경의 인증방법에 의존적이기 때문에 위와 같이 명세가 가능하다. 그러나 제품형 암호모듈의 경우에는 자체적으로 인증 메커니즘을 포함할 수 있기 때문에 암호모듈 내에 존재하는 인증 데이터를 보호할 수 있는 방법이 필요하다. 즉 인증 데이터에 대한 노출·수정·대체에 대해 보호할 수 있는 방법이 필요하다. 인증 데이터를 보호하기 위해 검증대상 암호알고리즘을 이용하여 암호화할 수 있으며, 또한 무결성 검증 과정을 통해 변조 여부를 확인할 수 있다. 인증 데이터가 암호모듈 외부에 저장된 경우에도 노출·수정·대체로부터 안전하도록 하기 위한 조치가 필요하다.	

항목	AS04.56
보안요구사항 개요	암묵적 또는 명시적 역할 할당 필요
VE04.56.01 VE04.56.02 VE04.56.03	인증 메커니즘 부재 시 암묵적 또는 명시적 역할 할당 필요(역할과 지침에 대한 명세)
작성 예시	
KISACrypto V1.0은 보안수준 1 소프트웨어 라이브러리 형태의 암호모듈로서 자체적인 인증 메커니즘을 포함하지 않으며, 운영환경이 제공하는 인증에 의존적이다. 암호모듈의 운영자는 암호모듈이 설치된 운영환경에 인증과정을 통해 사전에 정해진 역할을 할당받아야 한다. 인증과정을 통과한 운영자는 암호모듈을 암호모듈 관리자 또는 일반 사용자로 사용할 수 있다. 암호모듈은 운영자에 의해 호출되는 서비스에 따라 암호모듈 관리자 또는 암호모듈 일반 사용자 권한으로 동작한다.	
해설	
보안수준 1 소프트웨어 암호모듈의 경우에는 자체적인 인증 메커니즘을 탑재하지 않고, 운영환경에서 제공하는 인증방법을 사용할 수 있다. 운영환경에서 제공하는 인증방법을 통하여 적절한 역할을 할당받은 운영자는 암호모듈을 암호 관리자 또는 일반 사용자로 사용할 수 있음을 명세한다.	

6 소프트웨어/펌웨어 보안

가. 목적

암호모듈을 구성하는 소프트웨어/펌웨어 구성요소는 검증대상 무결성 검증방법을 이용하여 보호되어야 하며, 동작 전 자가시험 수행 시에 무결성이 검증되어야 한다. 무결성 검증이 실패할 경우 암호모듈은 오류 상태로 천이되어야 한다. 무결성 검증 과정 중 생성된 임시값은 무결성 검증이 완료된 후 제로화되어야 한다. 검증대상 무결성 검증방법(전자서명, MAC 등)을 위한 검증용 공개키나 MAC 키는 암호모듈 내부에 존재할 수 있지만 중요 보안매개변수로는 간주하지 않는다. 다음은 보안수준별 소프트웨어/펌웨어 보안요구사항이다.

- 보안수준 1의 소프트웨어/펌웨어, 하이브리드 암호모듈의 경우, 암호모듈 경계 내의 소프트웨어/펌웨어 구성요소에 대해 암호모듈 자체로 무결성을 검증하거나 또는 다른 검증필 암호모듈의 검증대상 동작모드로 검증해야 한다(다만 하이브리드 암호모듈의 경우, 하드웨어 구성요소와 독립적인 소프트웨어/펌웨어에 대해서만 적용됨).

이 장에서는 소프트웨어/펌웨어 보안의 주요 보안요구사항을 만족시키기 위해 개발업체가 작성해야 하는 AS 항목별 VE에 대한 작성 가이드를 제시한다.

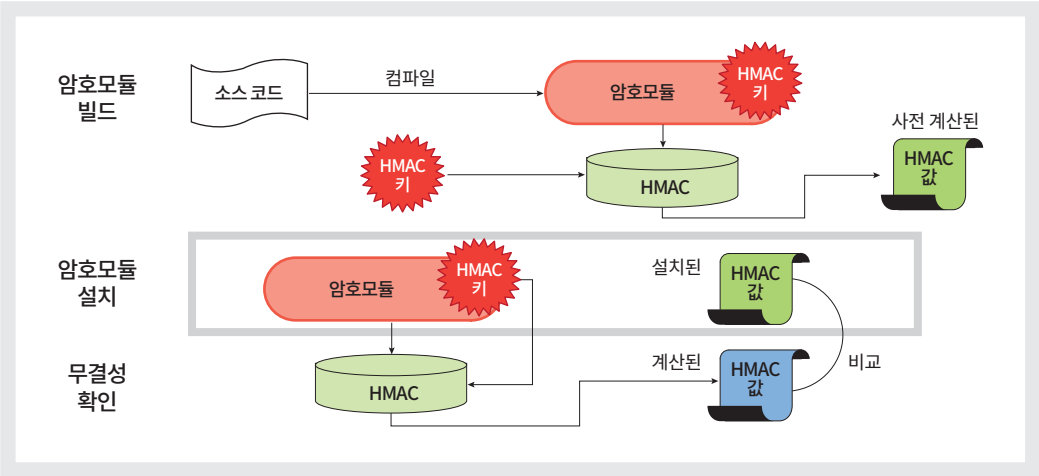
나. 기본사항

항목	AS05.02
보안요구사항 개요	무결성 기법과 시험 방법에 대한 명세
VE05.02.01	사용된 검증대상 무결성 기법과 운영자가 무결성 시험을 수행하는 방법 명세
작성 예시	
KISACrypto V1.0 암호모듈은 암호모듈에 대한 소프트웨어 무결성 시험을 위해 HMAC-SHA256을 적용한다. 무결성 검증값을 생성하는 절차는 다음과 같다. 1. 암호모듈을 빌드한 후 생성된 바이너리 파일에 대해 HMAC-SHA256 알고리즘을 적용하여 무결성 검증값 계산 2. 생성된 무결성 검증값은 바이너리 파일의 맨 뒤쪽에 연접함. 위의 과정에서 암호모듈 바이너리 파일에는 HMAC-SHA256을 이용하여 암호모듈의 무결성 검증값을 생성할 때와 동일한 HMAC 키가 분산된 형태로 하드코딩되어 저장되어 있다. 해당 키는 동작 전 자가시험에서 암호모듈의 무결성을 검증할 때 사용된다.	

암호모듈의 무결성을 검증하는 절차는 크게 암호모듈이 초기화될 때 자동적으로 수행되는 동작 전 자가시험과 사용자의 요청에 의해 수행되는 자가시험으로 구분된다. 동작 전 자가시험에서는 수행되는 암호모듈 무결성 시험 절차는 다음과 같다.

1. HMAC-SHA256 알고리즘에 대한 핵심 기능시험 수행
2. 과정 1 성공 시 암호모듈에 대한 무결성 시험 수행
   2.A 분산되어 있는 HMAC 키 복원
   2.B 메모리상에 로드된 암호모듈의 인스턴스로부터 원본 바이너리 파일의 경로를 얻어 냄(윈도우 환경에서의 GetModuleFileName 이용).
   2.C 얻어 낸 경로상의 암호모듈 바이너리 파일에 대해 HMAC-SHA256을 적용하여 무결성값 생성(이 때 암호모듈에 연접된 기존 무결성 검증값을 제외한 부분에 대해 HMAC-SHA256 적용 필요)
   2.D 생성된 무결성 검증값과 바이너리에 연접된 무결성 검증값 비교
3. 과정 2 성공 시 무결성 시험 통과

암호모듈의 무결성 검증값을 생성하여 연접하는 빌드 과정과 암호모듈의 무결성을 검증하는 설치 과정은 다음과 같다.



[암호모듈 빌드 과정과 암호모듈 설치 과정]

또한 운영자는 암호모듈이 제공하는 KISA_Crypto_selftest() API를 통해 암호모듈에 대한 무결성 시험을 포함한 동작 전 자가시험을 수행할 수 있다. 동작 전 자가시험에 대한 결과는 API의 반환값으로 반환되어 성공/실패 여부를 확인할 수 있다.

해설

암호모듈의 소프트웨어 무결성 시험을 위해 적용된 검증대상 무결성 알고리즘과 소프트웨어 무결성 시험을 수행하는 방법을 명세한다. 위의 예시에서는 HMAC을 이용한 무결성 검증방법을 제시하였으나, 전자서명 알고리즘을 이용한 방법도 가능하다.

항목	AS05.04
보안요구사항 개요	안전한 배포, 운영, 설치 전 변경되지 않아야 함.
VE05.04.01	암호모듈의 소프트웨어/펌웨어 명세 및 안전한 배포, 운영, 설치 전 변경되지 않는 근거 제시

작성 예시

KISACrypto V1.0 암호모듈은 개발이 완료된 후, 무결성 시험을 수행할 수 있는 형태로 만들어진 후 형상관리 시스템을 이용하여 형상이 관리된다. 자동화된 형상관리 시스템으로 git 서버를 구축하여 활용하고 있다. 형상관리되는 항목은 다음과 같다.

1. 문서(기본 및 상세설계서, 형상관리문서, 시험서)
2. 소스코드(*.c, *.h 파일)
3. 암호모듈(무결성 검증이 가능한 형태로 빌드된 최종 결과물)
4. 무결성 검증키
5. 암호모듈에 대한 정상동작 테스트 프로그램

암호모듈을 사용하기 위해 필요한 최종 암호모듈 라이브러리 파일과 암호모듈이 제공하는 API에 대한 원형을 정의한 헤더파일, 암호모듈의 기능을 정상적으로 검증하는 테스트 프로그램을 함께 배포한다. 암호모듈이 최종 빌드되어 무결성 검증이 수행될 수 있는 바이너리 형태로 배포되기 때문에 암호모듈 구현에 대한 소스코드를 얻어 내는 것이 현실적으로 어렵다. 도입기관에서는 배포된 암호모듈에 대해 다음의 절차를 통해서 암호모듈을 설치한다.

1. 배포된 암호모듈에 대한 해시값 비교 및 검증(암호모듈 시험기관 홈페이지에서 제공되는 검증필 암호모듈의 해시값과 암호모듈의 해시값을 비교)
2. 암호모듈의 해시값이 시험기관 홈페이지에 공시된 해시값과 일치할 경우, 암호모듈 설치 수행
3. 암호모듈에 대한 설치는 dll 또는 so 파일 형태로 제공되는 암호모듈 라이브러리 파일을 사용하고자 하는 위치에 복사함으로써 수행됨.
4. Makefile을 이용한 암호모듈 테스트 프로그램 설치
5. 암호모듈 테스트 프로그램을 통하여 암호모듈의 동작 전 자가시험 수행하여 암호모듈 정상동작 확인

해설

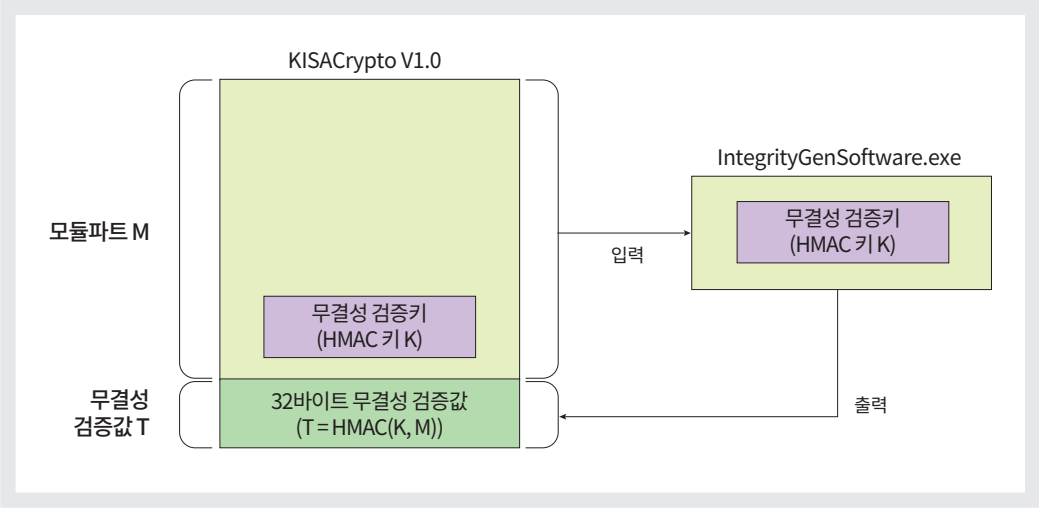
이 시험항목은 생명주기 보증 보안요구사항 영역의 내용과도 연관되며, 형상관리문서를 이용하여 증빙 가능하다.

항목	AS05.05
보안요구사항 개요	암호모듈 자체 또는 다른 검증필 암호모듈을 이용한 무결성 검증 필요
VE05.05.01 VE05.05.02 VE05.05.03 VE05.05.04	모든 소프트웨어/펌웨어 구성 요소에 대해 무결성 검증을 수행하는 근거와 방법 명세, 무결성 검증키의 위치 명세

작성 예시 1

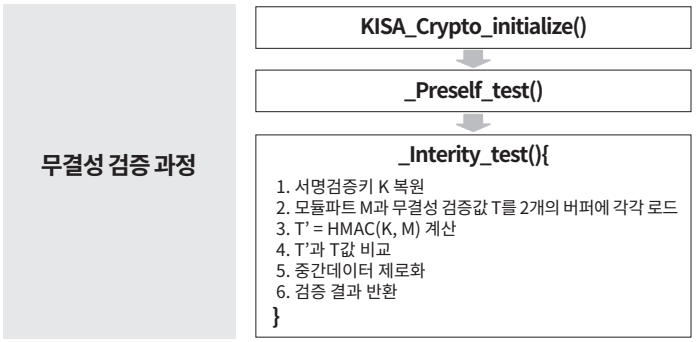
KISACrypto V1.0 암호모듈은 자체적으로 HMAC-SHA256을 이용하여 암호모듈에 포함한 소프트웨어 구성요소에 대한 무결성 검증을 수행한다. HMAC-SHA256 알고리즘에 사용되는 무결성 검증키 K의 크기는 256비트이며, 암호모듈의 무결성 검증을 위해 암호모듈 내부에 저장되어야 한다. 이 때 HMAC 알고리즘의 특성상 암호모듈에 분산 저장된다. 분산 저장함으로써 단순한 메모리 덤프를 통해 암호모듈의 무결성 검증키 값이 쉽게 노출되는 것을 방지한다. 아래 그림은 암호모듈의 무결성 검증값을 생성하는 자체 소프트웨어(IntegrityGenSoftware.exe)를 이용하여 32바이트 무결성 검증값을 생성하여 암호모듈의 끝에 연접하여 최종 암호모듈을 생성한다. 즉 IntegrityGenSoftware.exe는 HMAC-SHA256 알고리즘을 내부에 구현하고 있으며, 빌드가 완료된 모듈파트 M을 바이너리 파일로 읽어서 무결성 검증키 K를 이용하여 무결성 검증값 T를 생성한다. 생성된 T를 기존 모듈파트 M의 끝에 연접하여 저장함으로써 최종 모듈을 생성한다. 따라서 최종 빌드되어 배포되는 KISACrypto V1.0은 모듈파트 M과 무결성 검증값 T가 합쳐져 있는 형태이다.

암호모듈의 무결성 검증값을 생성하는 자체 소프트웨어는 무결성 검증키 K를 내장하고 있기 때문에 형상관리 시스템을 통하여 관리되며 인가된 사용자만 접근할 수 있다.



[HMAC을 이용한 무결성 검증값 생성 및 최종 암호모듈 생성 과정]

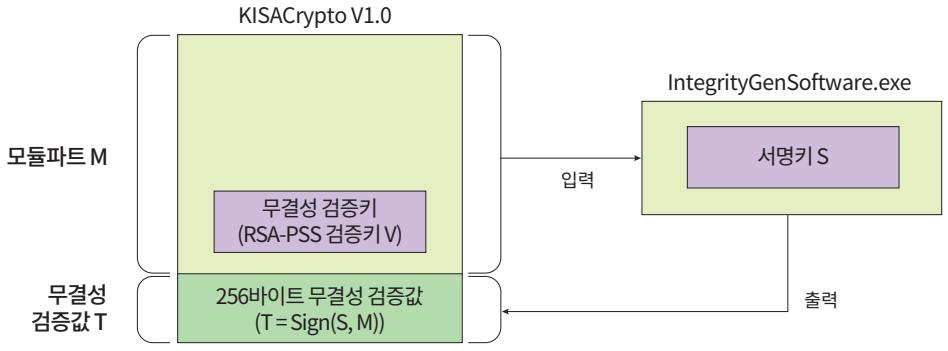
다음 그림은 암호모듈이 초기화될 때 자동으로 수행되는 동작 전 자가시험 내부에서 무결성 시험이 수행되는 절차를 나타낸다. 동작 전 자가시험을 수행하는 내부함수 _Preself_test()가 호출되면 해당 함수 안에서 HMAC에 대한 암호알고리즘 KAT 시험이 선행된 후, 무결성 시험이 _Integrity_test() 함수를 통해 수행된다. _Integrity_test()에서는 먼저 분산되어 암호모듈에 저장된 무결성 검증키 K를 복원한다. 다음으로 암호모듈 인스턴스로부터 암호모듈의 원본 파일을 바이너리 형태로 읽어서 모듈파트 M과 무결성 검증값 T를 각각 독립된 버퍼에 저장한다. 정상동작 확인이 완료된 HMAC-SHA256을 이용하여 M과 K를 입력으로 T'를 계산한다. T'과 T를 비교하여 암호모듈의 무결성이 보존되었는지를 확인한다. 무결성 검증 과정에서 생성된 중간 데이터를 모두 제로화한 후 검증결과를 반환한다. 여기에서 생성된 중간 데이터는 무결성 검증키, 무결성 검증값 등이 복사된 버퍼를 뜻한다.



[HMAC 기반의 암호모듈 무결성 검증 과정]

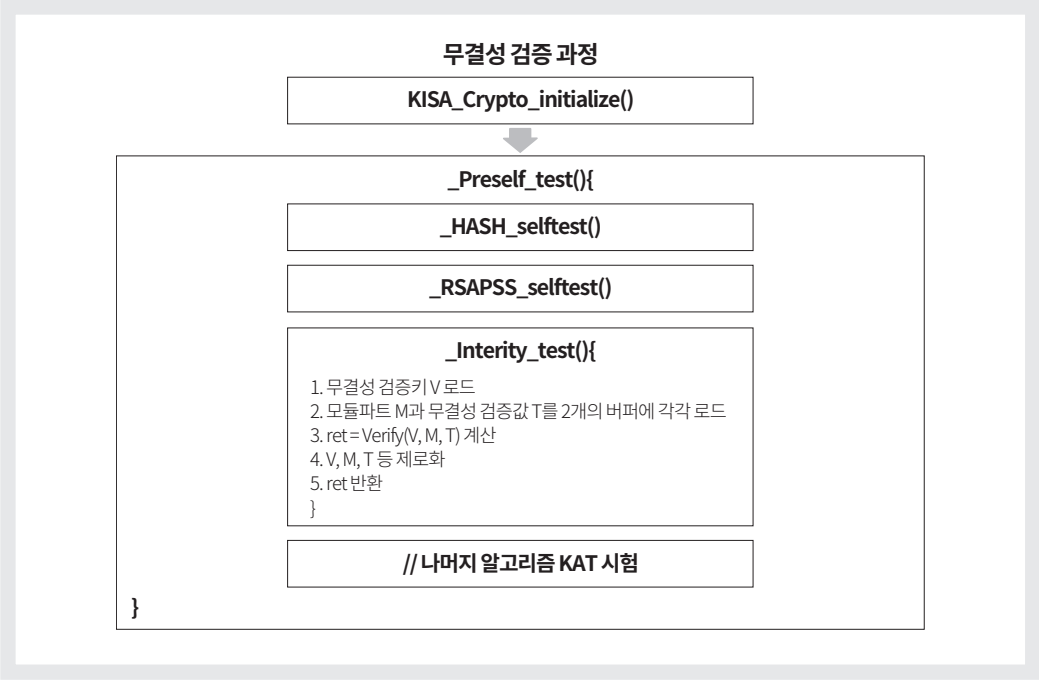
작성 예시 2

KISACrypto V1.0 암호모듈은 자체적으로 RSA-PSS(2048-bit)를 이용하여 암호모듈에 포함한 소프트웨어 구성요소에 대한 무결성 검증을 수행한다. RSA-PSS 2048-bit를 이용하여 무결성 검증을 위한 암호모듈의 서명값을 생성한다. 전자서명 알고리즘의 특징으로 암호모듈의 무결성 검증값을 생성할 때는 서명키 S가 사용되고, 암호모듈의 검증을 위해서는 서명검증키 V가 사용된다. 따라서 암호모듈은 서명검증키 V를 암호모듈 내부에 저장한다(실제로는 V키뿐 아니라 RSA-PSS 연산을 수행하기 위한 모듈러스 값도 저장된다). 서명검증키 V로부터 서명키 S를 도출하는 것은 불가능하기 때문에 서명검증키 V는 분산 저장될 필요 없다. 아래 그림은 암호모듈의 무결성 검증값을 생성하는 자체 소프트웨어(IntegrityGenSoftware.exe)를 이용하여 256바이트 무결성 검증값을 생성하여 암호모듈의 끝에 연접하여 최종 암호모듈을 생성한다. 즉 IntegrityGenSoftware.exe는 RSA-PSS 2048-bit 알고리즘을 내부에 구현하고 있으며, 빌드가 완료된 모듈파트 M을 바이너리 파일로 읽어서 서명 생성키 S를 이용하여 무결성 검증값 T를 생성한다. 생성된 T를 기존 모듈파트 M의 끝에 연접하여 저장함으로써 최종 모듈을 생성한다. 따라서, 최종 빌드되어 배포되는 KISACrypto V1.0은 모듈파트 M과 무결성 검증값 T가 합쳐져 있는 형태이다. 암호모듈의 무결성 검증값을 생성하는 자체 소프트웨어는 서명키 S를 내장하고 있기 때문에 형상관리 시스템을 통하여 관리되며 인가된 사용자만 접근할 수 있다.



[RSA-PSS를 이용한 무결성 검증값 생성 및 최종 암호모듈 생성 과정]

아래 그림은 암호모듈이 초기화될 때 자동으로 수행되는 동작 전 자가시험 내부에서 무결성 시험이 수행되는 절차를 나타낸다. 동작 전 자가시험을 수행하는 내부함수 _Preself_test()가 호출되면 해당 함수 안에서 RSA-PSS에 사용되는 해시함수와 RSA-PSS 대한 암호알고리즘 KAT 시험이 선행된 후, 무결성 시험이 _Integrity_test() 함수를 통해 수행된다. _Integrity_test()에서는 무결성 검증을 위한 서명검증키 V와 관련 파라미터를 준비한다. 다음으로 암호모듈 인스턴스로부터 암호모듈의 원본 파일을 바이너리 형태로 읽어서 모듈파트 M과 무결성 검증값 T를 각각 독립된 버퍼에 저장한다. 정상동작 확인이 완료된 RSA-PSS 알고리즘을 이용하여, 서명검증을 수행하는 Verify() 함수에 서명검증키, 모듈파트 M, 연접된 무결성 검증값 T를 입력으로 전달하여 수행한다. Verify() 함수는 전달받은 인자를 이용하여 M과 T에 대해서 무결성 검증을 수행한 후 성공/실패 결과값을 반환한다. 다음으로 무결성 검증 과정에서 생성된 중간 데이터 (V, M, T 등)를 모두 제로화한 후 검증 결과를 반환한다.



[RSA-PSS 기반의 암호모듈 무결성 검증 과정]

해설

위의 예시에서는 HMAC과 전자서명 RSA-PSS를 이용한 암호모듈 무결성 검증 방법에 대해 각각 설명한다.

항목	AS05.06
보안요구사항 개요	무결성 시험 실패 시 암호모듈은 오류 상태로 전환되어야 함.
VE05.06.01	암호모듈에 적용된 소프트웨어/펌웨어 무결성 시험에 대한 명세

작성 예시

KISACrypto V1.0 암호모듈은 RSA-PSS 2048-bit 전자서명을 이용하여 소프트웨어 구성요소에 대한 무결성 검증을 수행한다. 초기화 과정에서 수행되는 동작 전 자가시험 중에 무결성 검증이 수행되며, 실패할 경우 암호모듈은 심각한 오류 상태로 자동천이된다. 심각한 오류 상태에서는 암호모듈의 서비스를 사용할 수 없으며, 재시작 또는 재설치를 통해 오류를 해결해야 한다.

아래는 무결성 시험을 수행하는 _Integrity_test() 함수의 수도코드이다. 무결성 검증을 수행하는 아래 코드는 KISA_integrity.c 파일에 구현되어 있다. 암호모듈의 인스턴스로부터 암호모듈의 실행파일이 저장된 경로를 얻는 API를 호출하여 경로를 얻는다(A.1 과정). 획득한 경로를 이용하여 암호모듈의 모듈 파트와 무결성 검증값 파트를 각각 module_buf와 sig_buf 메모리에 로드한다(A.2 과정). module_buf와 sig_buf를 이용하여 _LibraryVerify()에서는 무결성 검증을 수행한다(A.3 과정). 이 때 암호모듈에 저장되어 있는 검증키를 로드해야 한다(B.1 과정). 검증키를 이용하여 모듈파트와 서명에 대해 무결성 검증을 수행한다(B.2 과정). 검증이 완료된 후에는 임시적으로 생성된 데이터에 대해 제로화를 수행한다(A.5, B.3 과정). 무결성 검증이 실패할 경우 오류코드를 반환하여 암호모듈이 심각한 오류 상태로 천이될 수 있도록 한다(A.4 과정).

```
int _Integrity_test(void) {
  // A.1 라이브러리 경로 얻기
  _GetModuleFileName(modulename, MODULE_NAME_LEN);
  // A.2 버퍼에 암호모듈 파일 내용 로드
  _Load_library(module_buf, sig_buf, modulename);
  // A.3 무결성 검증을 위한 내부 함수 호출
  rv =  _LibraryVerify(sig_buf, SIGN_LEN, module_buf, len);
  if (rv != 0) { // A.4 실패 시 오류코드를 반환하여 암호모듈을 심각한 오류 상태로 천이
    rv = -1;
    goto end;
  }
// A.5 제로화
Crypto_Zeroize(modulename, MODULE_NAME_LEN);
   ...
}
int _LibraryVerify(const u8 *signature, const u32 signatureLength, const u8 *message, const u32
messageLength) {
  ...
  // B.1 모듈 내부에 저장된 공개키 로드
  rv = PublicKey_Decode(pubKey, empty_o, sizeof(empty_o));
  if (rv != 0) {
    goto error;
  }
```

```
  // B.2 전자서명 기반으로 무결성 검증 수행
  rv = PKCS1_Verify(signature, signatureLength, PKCS1_MSGID_MESSAGE,
message, messageLength, PKCS1_SIGNID_RSA_PSS_SHA256, pubKey, &signParam);
  if (rv != 0) {
    goto error;
  }
  // B.3 제로화 수행(사용된 임시 변수에 대해 모두 제로화 필요)
  RSA_PublicKey_Free(pubKey);
  pubKey = NULL;
  ...
}
```

[암호모듈 무결성 시험 수도코드]

해설

이 시험항목은 AS05.05와 유사하며, AS05.05 항목별 VE와 함께 작성할 수 있다. 이 시험항목에서는 수도코드를 이용하여 무결성 검증의 과정을 설명한다. 수도코드를 이용하여 무결성 시험 과정을 설명할 때는 주석을 이용하여 코드의 내용을 설명하면 이해도를 높일 수 있다. 또한 위의 예시는 AS05.07(무결성 검증기술에 대한 요구사항)과 AS05.08(무결성 과정 중 임시로 생성된 데이터에 대한 제로화)의 시험항목에 대한 내용도 담고 있다.

항목	AS05.09
보안요구사항 개요	운영자의 요구에 따른 무결성 시험 수행 필요
VE05.09.01	운영자의 요구에 따라 인터페이스를 통해 무결성 시험을 수행할 수 있는 근거 제시

작성 예시

KISACrypto V1.0 암호모듈은 운영자의 요구에 따라 무결성 시험을 수행하는 KISA_Crypto_integritytest() API를 제공한다. 이 암호모듈은 소프트웨어 유형의 암호모듈이기 때문에 SFMI(Software or Firmware Module Interface)에 해당하는 인터페이스를 제공하며, 제어입력 인터페이스인 KISA_Crypto_integritytest()를 호출하여 암호모듈에 대한 무결성 시험을 수행할 수 있다. 무결성 검증을 수행하는 KISA_Crypto_integritytest()에 대한 데이터 입력, 데이터 출력, 상태 출력, 제어 입력에 대한 인터페이스는 암호모듈 인터페이스 보안요구사항 영역에서 명세하였으며, 명세된 것 이외의 인터페이스를 사용하지 않는다.

해설

동작 전 자가시험에서 수행되는 무결성 시험과 별개로 운영자가 필요 시에 수행 가능하도록 무결성 시험을 포함한 자가시험 API를 제공할 수 있다(무결성 시험만을 위한 API 형태 또는 무결성 시험을 포함한 자가시험 API의 두 가지 형태 모두 가능). AS05.10은 무결성 시험에 사용되는 정보는 명세된 인터페이스를 통해서만 전달되어야 함을 요구한다. 이 시험항목에서 관련 내용을 함께 명세할 수 있다.

7 운영환경

가. 목적

암호모듈에 대한 운영환경은 암호모듈이 운영되기 위해 요구되는 소프트웨어, 펌웨어, 하드웨어에 대한 관리를 의미한다. 소프트웨어, 펌웨어, 하이브리드 암호모듈에 대해서는 최소한으로 암호모듈이 운영될 컴퓨팅 플랫폼과 컴퓨팅 플랫폼에서 암호모듈을 제어하고 실행을 허용하는 운영체제를 포함한다. 하드웨어 암호모듈의 경우 모듈 안에 소프트웨어 요소 또는 펌웨어 요소의 실행을 허용하는 운영 시스템을 포함할 수 있다. 이 보안요구사항 영역에서는 암호모듈이 동작하는 다양한 운영환경에 대한 다음과 관련된 시험항목을 제시한다. 아래 시험항목 중 운영환경에 대한 암호모듈 시험은 암호모듈 시험서를 통해 증빙할 수 있다.

- 암호모듈 스스로 SSP를 제어하는지에 대한 검증
- 인가되지 않은 CSP 접근 및 제어되지 않는 보안매개변수 변경에 대한 방지
- 운영환경 설정 제한 사항 검증
- 암호모듈 운영환경에 대한 암호모듈 시험(정상동작에 대한 시험, 난수발생기 잡음원에 대한 엔트로피 시험)

암호모듈 기준에서는 다음 세 가지 형태의 운영환경을 정의한다.

- 변경 불가능한 운영환경: 운영자 또는 프로세스에 의해 암호모듈 구성요소, 컴퓨팅 플랫폼, 운영환경에 대한 변경을 방지하는 운영환경. 프로그래밍이 불가능한 컴퓨팅 플랫폼에서 하드웨어 모듈 내 펌웨어 암호모듈의 운영환경이 될 수 있음.
- 제한적 운영환경: 운영자 또는 프로세스에게 암호모듈 구성요소, 컴퓨팅 플랫폼, 운영환경에 제한적으로 변경을 할 수 있도록 허용하는 운영환경. 프로그래밍이 가능한 컴퓨팅 플랫폼에서의 하드웨어 모듈 내 펌웨어 암호모듈의 운영환경이 될 수 있음.
- 변경 가능한 운영환경: 범용 운영체제의 기능을 포함하고 기능을 추가·삭제·변경할 수 있는 운영환경. 소프트웨어/펌웨어/하이브리드 모듈의 일부가 아닌 소프트웨어를 로드하고 실행할 수 있는 운영체제는 변경 가능한 운영환경으로 간주함.

이 장에서는 운영환경의 주요 보안요구사항을 만족시키기 위해 개발업체가 작성해야 하는 AS 항목별 VE에 대한 작성 가이드를 제시한다.

나. 기본사항

항목	AS06.02
보안요구사항 개요	변경 가능한 운영환경의 경우, KS X ISO/IEC 19790 7.6.2의 운영체제 요구사항을 적용해야 함.

해설

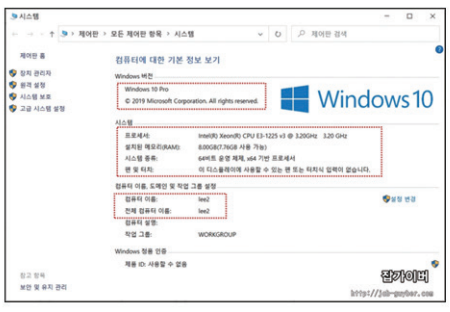
이 시험항목은 단독으로 시험되지 않고 AS06.05~AS06.29의 일부분으로 함께 시험된다. 변경 가능한 운영환경에서 보안수준 1 암호모듈에 대해서는 다음 사항을 만족해야 한다.

- 암호모듈의 각 인스턴스가 자신의 SSP를 제어해야 함.
- 운영환경은 응용 프로그램 프로세스를 독립적으로 유지할 수 있도록 함으로써 외부 프로세스로부터의 암호모듈의 CSP에 대한 인가되지 않은 접근과 제어되지 않은 보안매개변수 변경을 방지해야 함(이를 위한 설정이 필요하다면 보안정책문서상에 기술되어야 함).
- 암호모듈로부터 생성된 프로세스는 해당 암호모듈에 의해서만 소유되어야 함.

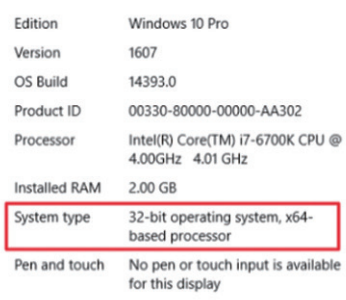
항목	AS06.03
보안요구사항 개요	암호모듈을 구동하기 위한 운영환경 정보 명세
VE06.03.01	암호모듈을 동작시키기 위한 운영환경 정보, 운영환경의 설정, 관리자 안내서에 대한 명세 필요

작성 예시

KISACrypto V1.0의 운영환경은 윈도우 10 운영체제 32/64비트와 우분투 리눅스 운영체제 20.04 32/64비트이다(운영환경에 명시한 것과 동일하게 OS, 버전, 비트 정보를 명세해야 함). 암호모듈을 사용하기 위해서는 운영체제에 사전에 암호모듈 운영자를 위한 계정 생성과 인증 절차가 마련되어 있어야 한다. 계정 생성과 인증 절차는 각 운영체제가 제공하는 일반적인 방법을 이용하여 구성 가능하다.



[윈도우 10 64bit]



[윈도우 10 32bit]

[리눅스 예시]

해설

암호모듈을 사용하기 위한 운영환경 정보와 운영환경 설정에 대한 설명을 기술한다. 암호모듈이 시험되는 모든 운영환경에 대한 정보를 모두 포함해야 한다(운영체제 버전, 비트 정보 등).

다. 변경 가능한 운영환경의 요구사항

항목	AS06.05
보안요구사항 개요	암호모듈이 독립적으로 SSP 제어 수행 필요
VE06.05.01	암호모듈 인스턴스가 스스로의 SSP를 제어하는 운영체제 메커니즘 명세

작성 예시

KISACrypto V1.0 암호모듈은 윈도우와 리눅스 운영체제에서 동작하는 암호모듈이며, 윈도우와 리눅스 운영체제는 암호모듈이 스스로의 SSP를 제어하고 다른 프로세스로부터 간섭받지 않도록 하기 위해 다음의 프로세스 보안 메커니즘을 제공한다.

보안 메커니즘	내용
단일 운영자 메커니즘	암호모듈은 단일 사용자 동작모드만 지원하는 운영체제에 설치되어 한 번에 한 명의 운영자만이 암호모듈을 사용할 수 있으며, 자신 이외의 어떤 프로세스와도 통신하지 않는다. 운영체제는 프로세스를 가상 메모리 영역상에서 분리하며, 이 가상 메모리는 운영체제와 CPU에 의해 다른 프로세스와 논리적으로 분리된다.
접근 방지 메커니즘	암호모듈은 자신을 사용하는 응용 프로그램 내 포함되어 독립적으로 운영되며, 암호모듈에 의해 생성된 프로세스는 외부 프로세스나 외부 운영자가 아닌 암호모듈에 의해서만 소유된다. 암호모듈이 실행 중이거나 운영 중인 동안 다른 프로세스가 평문으로 된 비밀키 및 핵심 보안매개변수 및 키 생성을 위한 중간값 등에 접근하는 것을 방지한다.
간섭 방지 메커니즘	암호모듈의 메모리 사용은 자신을 호출하는 프로그램이 할당한 영역 내의 주소로 국한된다. 암호모듈에 할당된 주소로 다른 응용 프로그램이 침범하는 경우 Access Violation 오류를 발생시키며 접근을 제한한다.

[운영환경 및 암호모듈 메모리 보호 메커니즘]

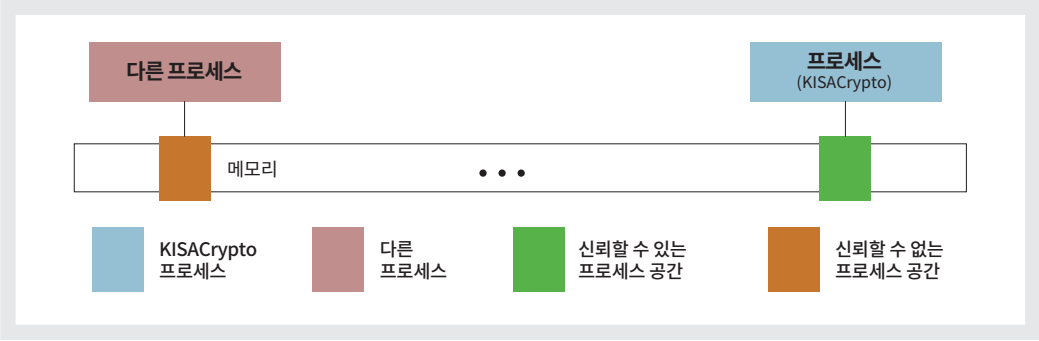
윈도우 운영체제에서 사용 가능한 프로세스 보안 및 메모리 보호 기법은 다음과 같다. 아래 방법은 운영체제 자체 또는 개발환경에서 설정을 통해 적용 가능하다.

1. GS
2. SEH(Structured Exception Handling)
3. ASLR(Address Space Layout Randomization)
4. DEP(Data Excution Prevention)

윈도우에서 제공하는 프로세스 보안 및 메모리 보호 기법은 다음과 같다. 아래 방법은 운영체제 자체 또는 개발환경에서 설정을 통해 적용 가능하다.

1. ASLR(Address Space Layout Randomization)
2. DEP(Data Excution Prevention)
3. ASCII-Armor
4. Stack Canary
5. RELRO(Relocation Read-Only)
6. PIE(Position Independent Executable)

다음은 운영체제가 암호모듈의 메모리를 다른 프로세스로부터 보호하는 방법을 도식화한 것이다.



[운영체제에 의한 암호모듈 메모리 관리]

해설

소프트웨어 암호모듈의 경우 위의 예시와 같이 암호모듈이 동작하는 범용 운영체제가 암호모듈의 SSP가 다른 프로세스로부터 간섭받지 않으며, 또한 SSP를 스스로 제어할 수 있도록 제공하는 메커니즘을 제시한다.

항목	AS06.06
보안요구사항 개요	운영환경의 암호모듈 프로세스 보호 기능
VE06.06.01	운영환경이 인가되지 않은 CSP 접근과 제어되지 않는 보안매개변수의 변경을 방지하는 방법 명세

해설

이 시험항목은 AS06.05에서 요구하는 VE 항목과 함께 명세될 수 있다.

항목	AS06.07
보안요구사항 개요	보안정책문서에 운영환경 설정 사항 명세
VE06.07.01	운영환경 설정과 규제 사항에 대해 보안정책문서에 명세 필요
작성 예시	
KISACrypto V1.0 암호모듈은 범용적인 운영체제인 윈도우 10과 우분투 리눅스 20.04에서 동작한다. 윈도우와 리눅스가 제공하는 프로세스 및 메모리 보호 방법을 이용하기 때문에 암호모듈의 인스턴스와 암호모듈이 유지하는 중요 보안매개변수가 다른 프로세스에 의해서 간접받지 않는다. 암호모듈의 인스턴스를 보고하기 위한 특별한 설정은 필요하지 않으며, 암호모듈을 운영하기 위해서는 운영환경에서 암호모듈 관리자와 사용자 계정을 생성한 후 인증과정을 통하여 역할을 부여받은 후에 암호모듈을 사용하도록 보안정책문서 상에서 권고한다.	
해설	
암호모듈이 구동하는 운영환경에 대한 정보, 암호모듈의 안전한 동작과 관련하여 필요한 설정과 규제 사항에 대해 보안정책문서에 명세하여 암호모듈 사용자가 참고하여 설정할 수 있도록 한다.	

항목	AS06.08
보안요구사항 개요	암호모듈에 의해 생성된 프로세스에 대한 제한
VE06.08.01	생성된 프로세스의 소유가 생성한 암호모듈로 국한시키는 방법 명세
작성 예시	
KISACrypto V1.0 암호모듈은 범용 소프트웨어 라이브러리형 암호모듈이며, 암호모듈 동작 중에 추가 프로세스를 생성하지 않는다. 암호모듈이 난수발생기의 잡음원 수집을 위해 사용하는 시스템 API는 운영체제가 제공하는 기능이며, 암호모듈이 생성과 소멸 과정에 참여하지 않는다.	
해설	
암호모듈이 추가 프로세스를 생성하는 경우, 생성되는 자식 프로세스의 소유권을 자식 프로세스를 생성한 부모 프로세스로 국한하는 방법을 명세해야 한다. 프로세스를 생성하는 경우가 존재하지 않는다면 이 시험항목은 프로세스를 생성하지 않는다고 명세하면 된다.	

8 | 중요 보안매개변수 관리

가. 목적

중요 보안매개변수(SSP: Sensitive Security Parameter)는 핵심 보안매개변수(CSP: Critical Security Parameter)와 공개 보안매개변수(PSP: Public Security Parameter)로 구성된다. CSP는 노출되거나 변경되면 암호모듈의 보안을 손상시킬 수 있는 정보로서 비밀키, 개인키, 패스워드, 난수발생기 상태정보 등을 포함한다. PSP는 변경되면 암호모듈의 보안을 손상시킬 수 있는 정보로서 공개키, 인증서 등을 포함한다. 이 시험 요구사항의 영역은 SSP의 생성·설정·주입·출력·저장·제로화를 포함한 SSP의 전체 생명주기에 대해 암호모듈을 안전하게 설계하고 구현했는지에 대한 시험을 수행한다. 암호모듈 내부의 CSP는 인가되지 않은 접근·사용·노출·수정·대체로부터 보호되어야 하며, PSP는 인가되지 않은 수정·대체로부터 보호되어야 한다. SSP의 생성·설정·주입·출력·저장·제로화는 각각 다음을 의미한다.

- SSP 생성:암호모듈에서 제공하는 암호 서비스를 위해 사용되는 암호키·난수 등의 생성을 의미하며, 검증대상 난수발생기를 이용하여 생성되어야 한다.
- SSP 주입/출력:암호모듈 경계를 기준으로 외부에서 암호모듈 내부로 SSP가 들어오는 경우를 주입이라고 하고, 내부에서 외부로 SSP가 나가는 경우를 출력이라고 한다. 수동 주입과 출력의 경우 직접적인 방법과 전자적인 방법으로 나뉜다. 직접적인 주입의 예는 키보드를 이용한 입력이 대표적이며, 전자적인 주입의 예로는 스마트카드 등이 존재한다. 자동화된 주입과 출력은 자동화된 SSP를 사용할 경우에 적용된다. 암호모듈은 보안수준과 주입·출력되는 방법에 따라 평문 또는 암호화된 형태로 주입·출력이 요구된다.
- SSP 설정:SSP 설정은 자동화된 SSP 설정과 수동 SSP 설정으로 나뉜다. 자동화된 SSP 설정은 자동화된 SSP 전송방법과 SSP 합의방법으로 구분되며, 검증대상 암호알고리즘을 이용하여 수행될 수 있다. 자동화된 SSP 설정의 경우에는 모든 보안수준에서 암호화된 형태로 SSP가 주입·출력되어야 한다. 수동 SSP 설정은 직접적인 주입·출력 또는 전자적인 주입·출력을 통해 수행되며, 보안수준에 따라 암호화된 형태 또는 평문 형태로 주입·출력될 수 있다.

이 보안요구사항 영역의 주된 시험항목은 크게 다음과 같이 분류된다.

- SSP 목록 및 인가되지 않은 접근·사용·노출·수정·대체로부터의 보호 방법
- 안전한 SSP 생성을 위한 검증대상 난수발생기(CTR_DRBG, HASH_DRBG, HMAC_DRBG) 구현 및 사용
- 검증대상 난수발생기에서 사용되는 잡음원과 잡음원의 엔트로피 추정값 명세
- SSP 생성, 설정, 주입·출력, 저장, 제로화에 대한 명세

이 장에서는 중요 보안매개변수 관리의 주요 보안요구사항을 만족시키기 위해 개발업체가 작성해야 하는 AS 항목별 VE에 대한 작성 가이드를 제시한다.

나. 기본사항

항목	AS09.01
보안요구사항 개요	암호모듈 내의 CSP에 대한 보호
VE09.01.01	암호모듈 내부의 모든 CSP에 대한 보호방법 명세 필요
작성 예시	

다음은 KISACrypto V1.0 암호모듈 내에서 생성될 수 있는 핵심 보안매개변수이다. KISACrypto V1.0 암호모듈은 소프트웨어 라이브러리 형태로서 응용 프로그램에게 암호 서비스를 제공한다. 따라서 암호모듈은 핵심 보안매개변수를 생성하고 제로화할 수 있는 기능을 응용 프로그램에게 제공하고, 암호모듈을 통해 생성한 핵심 보안매개변수는 응용 프로그램에서 관리한다. KISACrypto V1.0에서 기본적으로 유지하고 있는 원칙은 핵심 보안매개변수를 생성할 때 검증대상 난수발생기와 알고리즘별 표준에 의거한 키생성 방법을 준수하는 것이다. 핵심 보안매개변수가 암호모듈 내부의 암호 서비스 API에 의해 사용되는 경우, 핵심 보안매개변수를 통해 생성된 임시값 또는 데이터는 API가 종료되기 전에 제로화 과정을 통해 메모리에서 삭제된다. 인가되지 않은 접근·사용, 노출, 변경 및 대체를 막기 위한 방법은 다음과 같다.

- [인가되지 않은 접근·사용 방지] KISACrypto V1.0 암호모듈은 동적 라이브러리 형태로 빌드되어 응용 프로그램에 의해 사용되기 때문에 다수의 응용 프로그램이 암호모듈을 동시에 사용하더라도 서로 다른 독립된 메모리 공간을 할당받아 암호모듈을 사용하게 된다. 그뿐 아니라 암호모듈이 적재된 메모리를 포함한 하나의 응용 프로그램이 사용하는 메모리 공간은 가상 메모리 형태로 유지되기 때문에 실제 핵심 보안매개변수가 저장되는 물리적인 메모리의 주소가 다른 프로세스에게 노출되지 않는다. KISACrypto V1.0이 동작하는 운영환경은 윈도우, 리눅스의 범용 운영환경이기 때문에 운영체제가 이러한 프로세스별 메모리 보호 기능을 제공한다. 따라서 임의의 하나의 프로세스는 다른 프로세스와 그 프로세스가 사용하는 암호모듈의 메모리에 인가되지 않은 접근 또는 사용을 할 수 없다.
- [노출 방지] 암호모듈과 암호모듈을 이용하는 응용 프로그램은 메모리 공간을 공유하지만, 프로세스별로는 독립된 메모리 공간을 유지하며 서로 침범하는 것이 운영체제 레벨에서 방지된다. 따라서 암호모듈의 인스턴스에게 할당된 메모리는 다른 프로세스로부터 접근이 불가하기 때문에 암호모듈이 메모리상에서 유지하는 CSP가 다른 프로세스에게 노출되는 것이 방지된다. 그러나 암호모듈 내에서 사용이 해제된 메모리상에 기존에 존재했던 CSP가 남아 있어 노출될 가능성이 있기 때문에 KISACrypto V1.0 암호모듈은 API 내부에서 임시로 사용된 값에 대해서는 제로화(KISA_Crypto_Zeroize() 등을 이용)를 수행한 후에 메모리를 반환한다.
- [변경·대체 방지] 암호모듈 인스턴스에 의해 소유된 메모리상에 존재하는 CSP를 변경·대체하기 위해서는 해당 메모리 영역에 접근할 수 있어야 한다. 그러나 위에서 설명한 것과 같이 운영체제의 프로세스 보안 기능을 통해 프로세스 간 메모리 영역을 침범하여 접근하는 것은 불가하다.

메모리상에 존재하는 CSP를 이후의 사용을 위해 파일 형태로 HDD/SSD에 저장하는 경우가 존재할 수 있다. 영구적인 저장장치에 CSP가 저장되면 해당 파일을 생성한 주체뿐 아니라 다른 프로세스도 접근이 가능하다. 즉 CSP를 저장한 파일이 검증대상 암호알고리즘을 이용하여 적절히 보호되지 않을 경우, 인가되지 않은 접근·사용, 노출, 변경·대체가 가능해진다. 암호모듈에 사용되는 CSP의 경우에는 암호모듈이 관리해야 하기 때문에 CSP를 보호해야 한다. 그러나 CSP를 저장하는 주체가 응용 프로그램이고 암호모듈의 CSP가 아닌 경우에는 소프트웨어 라이브러리 형태의 KISACrypto V1.0은 HDD/SSD에 저장되는 CSP에 대해 보호할 의무를 갖지 않는다. 따라서 이 암호모듈을 이용하는 응용 프로그램(예: 제품형 암호모듈)에서 암호모듈을 통해 생성한 CSP를 파일 형태로 저장할 경우 검증대상 암호알고리즘을 이용하여 적절히 보호해야 한다.

분류	암호알고리즘	핵심 보안매개변수(CSP)
블록암호	SEED	– 비밀키(128비트) – 중간값(라운드 키 등)
메시지인증코드	HMAC	– 비밀키(224/256/384/512 비트 키)
난수발생기	HMAC_DRBG	– 시드(엔트로피 입력, 논스, 시드, 출력 난수) – 내부 상태(V, Key, C, Reseed_Counter)
공개키 암호	RSAES	– 개인키(2048 비트 d) 및 개인키 파라미터(p, q, dP, dQ, qInv)
전자서명	RSA-PSS	– 개인키 – 내부 난수값
키 설정	DH	– 개인키(224/256비트) – 공유키(2048비트)

[핵심 보안매개변수 상세 목록]

해설

이 시험항목은 AS09.02, AS09.04와 관련된다. AS09.01에서는 암호모듈의 CSP에 대한 명세와 보호방법에 대해 명세하고, AS09.02에서는 PSP에 대한 명세와 보호방법을 제시한다. AS09.04는 패스워드 해시값, 난수발생기의 상태 정보와 같은 중간값이 사용되는 경우 CSP로 정의하여 안전하게 관리하는지에 대한 명세를 제시한다. 위의 예시에서는 KISACrypto V1.0 암호모듈이 검증대상 블록암호, 메시지인증코드, 난수발생기, 공개키 암호, 전자서명, 키 설정 알고리즘을 탑재하였다고 가정하였을 때 세부 알고리즘별 핵심 보안매개변수(CSP)에 대해 명세한다. 명세된 CSP에 대해 인가되지 않은 접근, 사용, 노출, 변경 및 대체를 막기 위해 적용된 메커니즘에 대해 증빙해야 한다. 소프트웨어 라이브러리 형태의 암호모듈의 경우에는 일반적으로 메모리상에서의 인가되지 않은 접근, 사용, 노출, 변경 및 대체에 대한 방지 방법을 명세한다. 그러나 제품형 소프트웨어 암호모듈의 경우에는 핵심 보안매개변수를 HDD/SDD에 저장하는 경우가 존재하기 때문에 영구적인 저장소에 저장되는 핵심 보안매개변수를 검증대상 암호알고리즘을 이용하여 어떻게 보호하는지에 대한 명세가 추가로 필요하다.

항목	AS09.02
보안요구사항 개요	암호모듈 내의 PSP에 대한 보호
VE09.02.01	암호모듈 내부의 모든 PSP에 대한 보호방법 명세 필요

작성 예시

다음은 KISACrypto V1.0 암호모듈 내에서 생성될 수 있는 공개 보안매개변수(PSP)이다. CSP와 마찬가지로 PSP가 암호모듈 인스턴스의 메모리상에서 유지되는 동안에는 운영체제가 제공하는 프로세스 메모리 보호 기능을 통해 다른 프로세스에 의한 변경과 대체로부터 보호된다. 암호모듈 인스턴스가 할당된 메모리에 대해 반환할 경우, CSP와 마찬가지로 메모리상에 저장되었던 PSP에 대해 제로화를 수행한다. 파일 형태로 HDD/SDD에 저장되는 경우, 변경과 대체로부터 PSP를 보호하기 위해 암호모듈을 이용하는 응용 프로그램은 검증대상 암호알고리즘을 이용하여 파일에 저장된 PSP를 적절히 보호해야 한다.

분류	암호알고리즘	공개 보안매개변수(PSP)
블록암호	SEED	- CBC 운영모드(IV 값) - CTR 운영모드(CTR 값)
난수발생기	HMAC_DRBG	- 개별화문자열
공개키 암호	RSAES	- 공개키 파라미터(e, N)
전자서명	RSA-PSS	- 전자서명 검증키 파라미터(e, N)
키 설정	DH	- 공개키 - 도메인 파라미터(p, q, G)

[공개 보안매개변수 상세 목록]

해설

AS09.01 시험항목과 비교하여 매개변수의 종류가 CSP에서 PSP로 바뀌었을 뿐이며, 동일한 해설을 적용할 수 있다.

항목	AS09.04
보안요구사항 개요	키 생성 중간값에 대해 CSP로 간주 필요
VE09.04.01	패스워드의 해시값, 난수발생기 상태 등과 같은 암호키 생성 중간값에 대해 CSP로서 안전하게 관리하는 지에 대한 명세

작성 예시

KISACrypto V1.0 암호모듈은 패스워드의 해시값을 저장하는 경우는 없으나 검증대상 난수발생기를 구현한다. 검증대상 난수발생기와 관련된 CSP는 엔트로피 시드값, 난수발생기 내부 상태, 출력되는 난수열 등이다. CSP로 인식된 데이터는 사용이 완료된 후에 제로화를 통해 인가되지 않는 접근으로부터 보호한다. 예를 들어 난수발생기를 초기화하는 단계에서는 수집된 잡음원으로부터 난수발생기 내부 상태를 생성한 후에는 잡음원에 대한 정보를 제로화한다. 또한 난수발생기의 사용이 완료된 후에는 제로화 API를 통하여 난수발생기의 내부 상태를 모두 제로화한다.

해설
패스워드의 해시값, 난수발생기의 내부 상태, 엔트로피 시드값 등의 정보는 CSP로 자주 누락되는 부분이다. 암호모듈에 구현된 알고리즘별로 동작 중에 난수발생기를 이용하여 생성되는 난수값도 상황에 따라 CSP로 인식될 수 있다. 해당 CSP는 사용이 끝난 후에 적절히 제로화되는 것을 필요로 한다.

항목	AS09.05
보안요구사항 개요	KS X ISO/IEC 19790:2015의 A.2.9에 대한 명세
VE09.05.01	KS X ISO/IEC 19790:2015의 A.2.9에 명세된 요구사항을 충족하는 설계서 제공 필요

작성 예시

KISACrypto V1.0에 대한 개발문서는 KS X ISO/IEC 19790:2015의 A.2.9의 사항에 대해 다음과 같이 적절히 명세한다.

□ **암호모듈에 의해 사용되는 모든 CSP와 PSP 명세:** KISACrypto V1.0의 CSP와 PSP는 핵심 보안매개변수 상세 목록과 공개 보안매개변수 상세 목록에 각각 명세되었다.

□ **암호모듈에 포함된 난수발생기와 그 사용에 대한 명세:** KISACrypto V1.0 암호모듈은 HMAC_DRBG (HMAC-SHA-256)을 포함한다. 아래 표는 난수발생기를 사용하기 위한 API에 대한 설명이다. KISACrypto V1.0의 HMAC_DRBG는 예측내성을 지원한다. 예측내성이 설정된 경우, 난수를 생성하는 KISA_Crypto_drbgGenerate() API에서 난수를 생성하기 전에 리시드 과정을 수행하여 난수발생기의 내부 상태를 갱신한다. 예측내성이 설정되지 않은 경우, KISA_Crypto_drbgGenerate() API에서 리시드 카운터가 갱신 주기에 도달했는지를 판단하여 도달한 경우에만 리시드 과정을 수행한다. 리시드 카운터의 값은 고정값을 사용하며, KISA_Crypto_drbgInit() 호출 시에 설정된다.

API명	설명
KISA_Crypto_drbgInit();	- 논스, 잡음원, 개별화 문자열을 입력으로 받아 이를 통해 난수발생기 시드값을 생성한다. - 매개변수를 통하여 예측내성을 설정할 수 있다. - 난수발생기 시드값을 이용하여 난수발생기 내부 상태를 생성한다.
KISA_Crypto_drbgReseed();	- 추가 잡음원을 수집하여, HMAC_DRBG의 내부 상태를 갱신한다. - 선택적으로 추가 입력을 받을 수 있다. - 리시드 카운터의 값이 갱신 주기에 도달하거나, 예측내성이 활성화된 경우에 수행된다. 또는 사용자가 명시적으로 해당 API를 호출하여 난수발생기 내부 상태를 갱신할 수 있다.
KISA_Crypto_drbgGenerate();	- 사용자가 원하는 길이의 난수비트열을 생성하여 반환한다. - 선택적으로 추가 입력을 받을 수 있다. - 난수비트열을 생성한 후, 내부 상태를 자동적으로 갱신한다. - 이 API를 사용하기 전에 난수발생기가 KISA_Crypto_drbgInit()을 통해 먼저 초기화 되어 있어야 한다.
KISA_Crypto_drbgClose();	- 난수발생기의 사용이 완료된 후, 난수발생기가 사용하던 메모리를 제로화한다(내부 상태 포함).
KISA_Crypto_drbgRand();	- 초기화 과정, 난수 생성 과정, 제로화 과정을 한 번에 수행하는 API로서, 1회성으로 난수발생기를 사용한 후 바로 제로화하는 경우에 사용할 수 있는 API이다.

[난수발생기 API 설명]

다음은 난수발생기를 사용하는 방법에 대한 예시이다.

```
int                 rv = 0;
u32                 length = 256;
u8                  rand_data[4096] = { 0, };    // 난수 버퍼

// HMAC_DRBG(HMAC-SHA-256) 난수발생기 초기화
rv = KISA_Crypto_drbgInit(KISA_RAND_ID_DRBG_HMAC_SHA256, PR_ON);
if (rv != 0) {
        fprintf(stderr, "KISA_Crypto_drbgInit failed: %08x\n", rv);
        goto error;
}
// HMAC_DRBG(HMAC-SHA-256) 난수발생기 난수 생성
rv = KISA_Crypto_drbgGenerate(rand_data, length);
if (rv != 0) {
        fprintf(stderr, "KISA_Crypto_drbgGenerate failed: %08x\n", rv);
        goto error;
}
// 생성된 난수값 출력
printHex("rand_data: ", rand_data, length);

// HMAC_DRBG(HMAC-SHA-256) 난수발생기 리시딩
rv = KISA_Crypto_drbgReseed();
if (rv != 0) {
        fprintf(stderr, “KISA_Crypto_drbgReseed failed: %08x\n", rv);
        goto error;
}

// HMAC_DRBG(HMAC-SHA-256) 난수발생기 난수 생성
rv = KISA_Crypto_drbgGenerate(rand_data, length);
if (rv != 0) {
        fprintf(stderr, "KISA_Crypto_drbgGenerate failed: %08x\n", rv);
        goto error;
}

// 생성된 난수값 출력
printHex("rand_data: ", rand_data, length);

error:
// HMAC_DRBG(HMAC-SHA-256) 난수발생기 제로화
KISA_Crypto_drbgClose();
```

[난수발생기 사용 예시]

□ **엔트로피 입력 각각에 대한 최소 엔트로피 명세:** KISACrypto V1.0 암호모듈은 난수발생기의 초기화 과정 또는 리시드 과정에서 잡음원을 수집하여 이를 이용하여 난수발생기 내부 상태를 구성하거나 갱신한다. TTAK.KO-12.0306/R1 (소프트웨어 환경에서의 잡음원 엔트로피 검증 알고리즘) 표준에서 제시하는 통계적인 테스트에 의거하여 증명한다. 통계적인 테스트를 통하여 적절한 잡음원을 선별하여 암호모듈의 난수발생기가 사용하도록 한다. 암호모듈의 동작 중 수집되는 잡음원의 엔트로피에 대해서는 정확한 추정이 어렵기 때문에 반복 횟수 테스트와 적응성 비율 테스트를 실시간 건전성 테스트로 적용한다.

□ **난수발생기를 이용하는 모든 SSP 생성 방법과 SSP 생성 알고리즘에 대한 명세:** 난수발생기는 아래에 제시된 SSP 생성 시에 사용되며, SSP별로 명시된 표준에 의거하여 SSP를 생성한다. HMAC_DRBG는 'TTAK.KO-12.0332-Part1: HMAC 기반 결정론적 난수발생기' 표준에 의거하여 설계하고 구현하였다.

서비스	용도	관련 API	비고
블록암호	마스터키 생성	KISA_Crypto_drbgGenerate()	난수발생기 난수 생성 API를 그대로 이용
	IV, CTR 생성		
메시지인증코드 (HMAC)	HMAC 키	KISA_Crypto_drbgGenerate()	난수발생기 난수 생성 API를 그대로 이용
공개키 암호 (RSAES)	키쌍 생성	KISA_Crypto_genKeyPair()	소수 p, q 생성 시 난수발생기 난수 생성 API 이용 [KS X ISO/IEC 18033-2(2017) 표준 기반]
	암호화	KISA_Crypto_publicEncrypt()	OAEP 인코딩 수행 시 랜덤 시드값 생성 시, 난수발생기 난수 생성 API 이용
전자서명 (RSA-PSS)	키쌍 생성	KISA_Crypto_genKeyPair()	소수 p, q 생성 시 난수발생기 난수 생성 API 이용 [KS X ISO/IEC 14888-2(2011) 표준 기반]
	서명 생성	KISA_Crypto_sign()	PSS 인코딩 수행 시, 랜덤 솔트값 생성 시, 난수발생기 난수 생성 API 이용
키 설정	파라미터 생성	KISA_Crypto_DH_genParameter()	소수 p와 q, 생성원 G 생성 시, 난수발생기 난수 생성 API 이용 [KS X ISO/IEC 11770-3(2018) 표준 기반]
	키쌍 생성	KISA_Crypto_DH_genKeyPair()	개인키 x 생성 시 난수발생기 난수 생성 API 이용 [KS X ISO/IEC 11770-3(2018) 표준 기반]

[SSP 생성 방법 및 난수발생기 사용 명세]

□ **모든 SSP 설정 방법에 대한 명세:** KISACrypto V1.0 암호모듈은 자동화된 키 설정 방법으로서 DH 알고리즘을 이용한 키 설정 방법을 제공한다. 암호모듈은 DH 알고리즘 파라미터로서 공개키 2048-bit, 개인키 256-bit 파라미터를 이용한다. DH는 상대방과 키를 공유하기 위한 키 합의 알고리즘으로서 먼저 파라미터를 생성한 후(KISA_Crypto_DH_genParameter()), 키를 공유하고자 하는 상대방과 파라미터를 공유해야 한다. 동일한 파라미터 기반으로 키쌍을 생성한 후(KISA_Crypto_DH_genKeyPair()), 상대방의 공개키와 자신의 개인키를 이용하여 공통의 키를 계산하는 것이 가능하다(KISA_Crypto_DH_computeSharedSecret()).

API명	설명
KISA_Crypto_DH_genParam()	- 공개키와 개인키의 크기를 전달하여 조건에 맞는 DH 파라미터를 생성하여 저장
KISA_Crypto_DH_genKeyPair()	- 파라미터를 이용하여 공개키와 개인키 쌍 생성
KISA_Crypto_DH_computeSharedSecret()	- 파라미터, (자신의) 개인키와 (상대방의) 공개키를 이용하여 공유 비밀키 값 생성

[키 설정 API 설명]

□ **암호모듈에 적용된 SSP 주입과 출력 방법 명세:** KISACrypto V1.0 암호모듈은 주입과 출력을 위한 API를 제공하지만 실제로 주입과 출력을 수행하는 주체는 암호모듈을 이용하는 응용 프로그램이다. 따라서, KISACrypto V1.0 암호모듈 자체로는 주입과 출력을 대한 시험항목을 직접적으로 적용받지 않는다. 근거는 다음과 같다. KISACrypto V1.0 암호모듈은 소프트웨어 라이브러리 형태로서 독립적으로는 동작할 수 없으며, 응용 프로그램에 링크되어서만 실행될 수 있다. 따라서 응용 프로그램과 암호모듈은 메모리 공간을 공유한다. 또한 암호모듈은 SSP 생성을 위한 API를 제공하며 생성된 SSP는 응용 프로그램에서 사용되며 관리된다. 외부로부터 주입된 SSP는 응용 프로그램을 거쳐 암호모듈의 관련 API에 전달된다. 예를 들어 DH 암호 서비스를 통한 키 합의 과정 수행 시, 물리적으로 떨어져 있는 상대방으로부터 공개키를 전달받는 주체는 응용 프로그램이 되고 응용 프로그램은 키 합의를 통한 공유키 생성을 위해 암호모듈의 관련 API에 SSP인 공개키를 전달한다. 이와 같이 암호모듈은 응용 프로그램의 경계 안에 속해 있으며, 실제 주입과 출력을 수행하는 주체는 응용 프로그램이 된다.

□ **암호모듈 내부에 저장되는 SSP에 대해 인가되지 않은 접근·사용, 노출, 변경·대체를 방지하는 방법 명세:** 해당 부분에 대한 사항은 VE09.01.01과 VE09.02.01에 대한 증빙을 통해 제시하였다.

□ **암호모듈 내부에 저장된 PSP에 대해 개체(운영자, 역할, 프로세스)와 연관시키는 방법 명세:** 해당 부분에 대한 사항은 VE09.03.01에 대한 증빙을 통해 제시하였다.

□ **암호모듈에 적용된 제로화 방법:** KISACrypto V1.0 암호모듈은 절차적 제로화와 운영적 제로화를 적용한다. 절차적 제로화는 API 내부에서 사용이 완료된 SSP를 제로화하는 것이다. 즉 매개변수를 통해 API 내부의 임시 변수에 복사된 데이터나 SSP로부터 도출된 데이터를 API를 반환하기 전에 메모리 공간에서 제로화하는 것이다. 제로화한다는 의미는 SSP가 저장되어 있던 메모리 공간을 특정값('0' 또는 의미 없는 난수)으로 덮어 쓴다는 것이다. 이 암호모듈에서는 '0'으로 메모리 공간을 덮어 쓰는 제로화 방법을 적용한다. long-term으로 사용되는 SSP에 대해서는 즉각적인 제로화를 적용하기 어렵다. 즉 암호모듈이 동작하는 동안에 사용되는 SSP나 API 단위로 수행되는 것이 아닌, 특정 작업 단위로 수행되는 경우에는 SSP를 제로화할 경우 작업을 연속적으로 진행하는 것이 어렵다. 이는 운영적 제로화 방법을 적용한다. 운영적 제로화는 암호모듈이 특정 SSP를 제로화할 수 있도록 API를 제공하는 것이고, SSP의 사용이 완료된 후에는 응용 프로그램이 해당 API를 이용하여 SSP를 제로화한다.

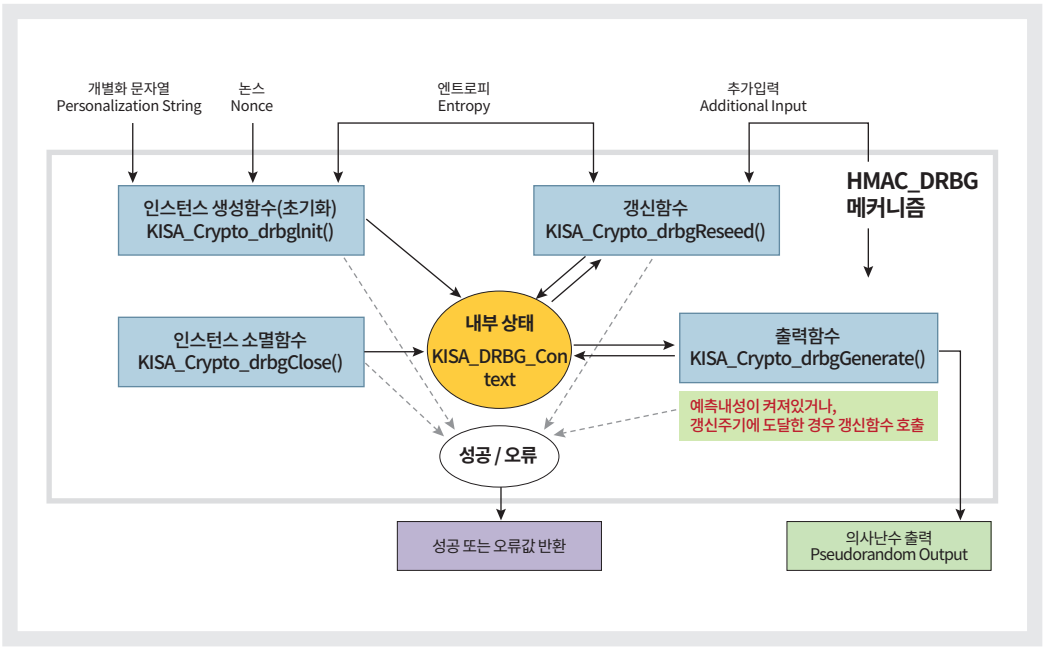
해설

KS X ISO/IEC 19790:2015의 A.2.9 (중요 보안매개변수 관리)에서는 SSP 명세, 난수발생기 설계 및 사용 방법 명세, 최소 엔트로피 분석, SSP 생성·주입·출력·저장·제로화에 대해 개발 문서 관점에서 적절히 명세하도록 요구한다. 따라서 이 시험항목에서 요구되는 사항은 다른 시험항목에서 부분적으로 요구된 것도 존재하기 때문에 실제 개발 문서 작성 시에는 중복된 사항은 상호 참조를 통해 명세하는 것이 가능하다.

다. 난수발생기

항목	AS09.06
보안요구사항 개요	검증대상 난수발생기 사용
VE09.06.01 VE09.06.02	난수발생기의 목록과 사용방법 및 난수 생성과 설정 시에 검증대상 난수발생기를 사용했다는 증빙 명세
작성 예시	

KISACrypto V1.0은 HMAC_DRBG(HMAC-SHA256)을 제공한다. 아래는 암호모듈에 구현된 난수발생기에 대한 도식이다. 암호모듈은 KISA_Crypto_drbgInit()을 이용하여 난수발생기의 인스턴스를 생성할 수 있으며, 매개변수를 통하여 예측내성을 설정하거나 끌 수 있다. KISA_Crypto_drbgInit() 내부에서는 난수발생기 인스턴스 생성을 위하여 운영환경에 제공하는 잡음원을 수집하여 난수발생기 시드값을 구성한다. 시드값을 이용하여 난수발생기의 내부 상태를 구성한다. KISA_Crypto_drbgGenerate()에서는 사용자가 원하는 비트 길이만큼의 난수를 출력한다. 예측내성이 설정되어 있는 경우 난수를 생성하기 전에 운영환경으로부터 잡음원을 수집하여 내부 상태를 갱신한 후에 난수를 생성하여 출력한다. 난수를 출력한 후에는 추가로 내부 상태를 갱신한다. 예측내성이 설정되어 있지 않다면, 리시드 카운터의 값이 난수발생기에 설정된 갱신 주기에 도달한 경우에 추가 리시드 과정을 수행한다. 리시드 과정을 수행한 후에는 리시드 카운터의 값을 다시 1로 초기화시킨다. 사용이 완료된 난수발생기에 대해 KISA_Crypto_drbgClose()를 호출하면 난수발생기의 내부 상태를 메모리에서 삭제할 수 있다.



[암호모듈에 구현된 난수발생기 구조]

'[난수발생기 API 설명]', '[난수발생기 사용 예시]'를 통하여 암호모듈이 탑재하고 있는 난수발생기 목록과 사용 방법에 대해 명세하였다. 또한 '[SSP 생성 방법 및 난수발생기 사용 명세]'를 통하여 SSP 생성 및 설정 시에 검증대상 난수발생기를 적용하고 있음을 명세하였다. 추가로 SSP 생성 및 설정 시 검증대상 난수발생기가 사용되고 있음을 다음의 표를 통하여 확인 가능하다.

서비스	용도	관련 API	소스코드 정보(파일 및 관련 코드 정보)
블록암호	마스터키 생성	KISA_Crypto_ drbgGenerate()	KISA_drbg.c KISA_Crypto_drbgGenerate() → hmac_drbgGenerate()
	IV, CTR 생성		
HMAC	HMAC 키	KISA_Crypto_ drbgGenerate()	KISA_drbg.c
공개키 암호 (RSAES)	키쌍 생성	KISA_Crypto_ genKeyPair()	KISA_rsa.c KISA_Crypto_genKeyPair() → BN_gen_prime() → hmac_drbgGenerate()
	암호화	KISA_Crypto_ publicEncrypt()	KISA_rsa.c KISA_Crypto_publicEncrypt() → BN_gen_seed() → hmac_drbgGenerate()
전자서명 (RSA-PSS)	키쌍 생성	KISA_Crypto_ genKeyPair()	KISA_rsa.c KISA_Crypto_genKeyPair() → BN_gen_prime() → hmac_drbgGenerate()
	서명 생성	KISA_Crypto_sign()	KISA_rsa.c KISA_Crypto_publicEncrypt() → BN_gen_salt() → hmac_drbgGenerate()
키 설정	파라미터 생성	KISA_Crypto_DH_ genParameter()	KISA_dh.c KISA_Crypto_DH_genParameter() → BN_gen_prime() → hmac_drbgGenerate()
	키쌍 생성	KISA_Crypto_DH_ genKeyPair()	KISA_dh.c KISA_Crypto_DH_genKeyPair() → BN_gen_prime() → hmac_drbgGenerate()

[SSP 생성 및 설정 방법에서 난수발생기 적용 증빙]

해설
이 시험항목은 AS09.05 항목별 VE09.05.01과 함께 증빙 가능하다. 난수발생기의 설계와 구현이 표준에 의거했음을 증빙하기 위해 '암호모듈에 구현된 난수발생기 구조'와 같이 난수발생기 구조에 대한 그림을 실제 구현된 난수발생기 API와 대응시켜 설명하는 것이 바람직하다. 또한 SSP 생성과 SSP 설정 단계에서 난수발생기가 사용되는 것을 명확하게 증빙하기 위해 소스코드 파일명과 관련 코드를 제시하는 것이 바람직하다. 관련 코드를 제시할 때는 모든 코드를 제시하기보다는 SSP 생성과 SSP 설정 API 내부에서 난수발생기를 호출하는 부분을 명시하면 된다.

항목	AS09.07
보안요구사항 개요	외부 엔트로피 입력으로부터 생성된 데이터에 대한 CSP 간주
VE09.07.01	암호 경계 외부에서 수집된 엔트로피로부터 생성된 데이터에 대해 CSP로 명세 필요

작성 예시
KISACrypto V1.0 암호모듈은 윈도우와 리눅스와 같이 신뢰된 범용 운영체제에서 동작하는 소프트웨어 라이브러리형 암호모듈이다. 암호모듈은 운영체제가 제공하는 시스템 API를 이용하여 난수발생기에서 사용되는 잡음원을 수집한다. 즉 암호모듈에서 구현된 난수발생기는 초기화 과정에서 운영체제가 제공하는 시스템 API를 이용하여 잡음원을 수집한 후 이를 난수발생기 시드로 하여 내부 상태를 구성한다. 난수발생기 내부 상태를 통해 생성되어 출력된 난수열은 암호모듈 내부에서 CSP로 간주되어 처리된다. 즉 출력된 난수열은 사용이 완료된 후 제로화된다. 그뿐 아니라 난수발생기 시드로 활용된 수집된 잡음원도 CSP로 간주되어 난수발생기 내부 상태를 구성한 후 제로화된다. 최종적으로 난수발생기 사용이 완료될 경우, 잡음원으로부터 구성된 난수발생기 내부 상태로 제로화되는 과정을 거친다.
해설
엔트로피 잡음원은 난수발생기 내부 상태를 구성하는 비결정론적인 데이터이다. 따라서 사용이 완료된 이후에는 제로화가 필요하다. 또한 엔트로피 잡음원으로부터 생성된 난수발생기 출력값도 난수 또는 비밀키로 사용되기 때문에 CSP로 간주되는 것이 필요하다.

라. 중요 보안매개변수 생성

항목	AS09.08
보안요구사항 개요	SSP 생성을 손상하는 행위는 전수조사와 동일한 연산량 필요
VE09.08.01	난수발생기 시드값을 추측하는 행위는 SSP값을 전수조사하는 만큼의 연산량을 필요로 한다는 근거 명세
작성 예시	

KISACrypto V1.0 암호모듈의 운영환경은 윈도우 10 (64-bit)과 우분투 리눅스 20.04이다. 암호모듈이 동작하는 모든 각 운영환경에서 난수발생기의 씨드값을 구성하기 위해 수집되는 잡음원을 명세해야 한다. 즉, 잡음원의 이름, 출력 바이트 길이, 출력되는 바이트에 대한 정보, 그리고 추정된 엔트로피를 제시해야한다.

※ 잡음원, 엔트로피 소스, 결정론적 난수발생기 운영방법은 TTAK.KO-12.0235(운영체제별 잡음원 수집 및 운용지침) 참고하여 작성한다.

CryptoGenRandom()은 윈도우 환경에서 대표적으로 사용되는 난수 생성 API로서 암호모듈에서는 해당 API의 출력을 잡음원으로 활용한다. 윈도우 환경에서는 NtQuerySystemInformation() API를 제공하며 파라미터 설정을 통해 시스템의 다양한 정보를 얻을 수 있다. 그뿐 아니라 해당 API를 통해 얻은 정보는 난수발생기의 시드값으로 활용될 수 있음이 명세되어 있다(참고: https://docs.microsoft.com/en-us/windows/win32/api/winternl/nf-winternl-ntquerysysteminformation). 시드값으로 활용될 수 있는 대표적인 파라미터는 SystemExceptionInformation, SystemInterruptInformation, SystemLookasideInformation, SystemPerformanceInformation이다.

No.	잡음원 이름 (잡음원 설명)	수집 길이 (단위:바이트)	분석 결과 (단위:비트)
1	CryptGenRandom (윈도우 운영체제가 제공하는 의사난수 생성 API. 가변 길이 출력 지원)	-	-
2	GetPerformanceInfo (시스템의 실제 및 가상 메모리 사용 현황 대한 정보. PERFORMANCE_INFORMATION 구조체를 통한 정보 반환)	-	-
3	GetProcessIoCounters (지정된 프로세스에서 수행한 모든 I / O 작업에 대한 계정 정보. IO_COUNTERS 구조체를 통한 정보 반환)	-	-
4	GlobalMemoryStatusEx (시스템의 현재 물리 메모리와 가상 메모리에 대한 사용 정보 반환. MEMORYSTATUSEX 구조체를 통한 정보 반환)	-	-
5	QueryPerformanceCounter(high resolution의 time stamp값 반환)	-	-

No.	잡음원 이름 (잡음원 설명)	수집 길이 (단위:바이트)	분석 결과 (단위:비트)
6	NtQuerySystemInformation(SystemInterruptInformation) (시스템 인터럽트 정보)	-	-
7	NtQuerySystemInformation(SystemExceptionInformation) (시스템 예외 정보)	-	-
8	NtQuerySystemInformation(SystemLookasideInformation) (시스템 메모리 변환 색인 정보)	-	-
9	NtQuerySystemInformation(SystemPerformanceInformation) (시스템 성능 정보)	-	-
	합계		

[Windows 10 64bit 엔트로피 잡음원]

우분투 리눅스 20.04 64-bit 운영환경에서 사용가능한 잡음원은 다음과 같다. 리눅스 환경의 대표적인 잡음원으로 /dev/urandom이 있다. /dev/urandom은 내부적으로 랜덤풀을 관리하는 잡음원으로서 충분한 엔트로피가 보장되지 않더라도 사용자의 요청에 대해 즉각적으로 난수값을 반환한다(/dev/random은 충분한 엔트로피가 보장될 때까지 출력이 blocking되기 때문에 이 암호모듈에서는 사용하지 않는다). 나머지 잡음원은 리눅스 운영체제가 관리하는 시스템에 대한 정보를 /proc 폴더의 관련 파일로부터 읽어들여 사용한다.

No.	잡음원 이름 (잡음원 설명)	수집 길이 (단위:바이트)	분석 결과 (단위:비트)
1	/proc/stat(시스템 사용 통계)	-	-
2	/proc/interrupts[장치에서 발생된 인터럽트(IRQ) 정보]	-	-
3	/proc/meminfo(메모리 사용량 정보)	-	-
4	/proc/net/dev(네트워크 인터페이스에 대한 통계)	-	-
5	/proc/uptime[시스템의 얼라이브(alive) 시간]	-	-
6	/proc/vmstat(가상메모리 통계)	-	-
7	clock(사용된 프로세스 시간)	-	-
8	getrusage(프로세스의 자원 사용량)	-	-
9	gettimeofday(시스템의 현재 시간)	-	-
10	/dev/urandom(유닉스 계열 운영 체제에서 차단 방식의 유사난수발생기)	-	-
	합계		

[우분투 리눅스 20.04 64-bit 엔트로피 잡음원]

위의 예시에서 제시한 [Windows 10 64bit 엔트로피 잡음원]와 [우분투 리눅스 20.04 64-bit 엔트로피 잡음원] 표의 잡음원들은 소프트웨어 라이브러리 형태의 암호모듈 내부에서 난수발생기의 씨드값을 구성하기 위한 잡음원으로서 사용가능하다. 하지만, 운영환경의 시스템 구성에 따라서 잡음원의 엔트로피 값이 가변적일 수 있기 때문에 정확한 출력값과 엔트로피 값은 제시하지 않는다. 잡음원에 대한 엔트로피 분석은 암호모듈 개발업체에서 통과하기 어려운 시험항목 중 하나이기 때문에 잡음원의 수집과 분석 과정을 시험기관의 안내에 따라 정확히 수행하는 것이 필요하다.

※ 2022.6.1.부터 난수발생기 신규 시험방법론을 의무적용하며, 적용 대상은 신규 신청 시험, 검증효력 유효기간이 연장되는 재검증 신청 시험 등이 해당된다. 관련 표준 및 FAQ는 국가정보원 홈페이지 또는 한국인터넷진흥원 암호이용활성화 홈페이지를 통해 확인할 수 있다.

해설

난수발생기 신규 시험방법은 다음과 같다.

STEP 1. (잡음원 엔트로피 평가) 신청기관이 잡음원별 최소 25만개의 샘플 데이터를 수집하여 제출

STEP 2. (잡음원 반복 사용 횟수 확인 및 보안강도 1차 평가) 잡음원별 엔트로피를 평가한 후, 통계적 특성 시험을 통과한 주요 잡음원 1종을 선정하고, 사용된 출력 수를 고려하여 잡음원 출력열의 엔트로피 평가

※ 잡음원 엔트로피 평가는 TTAK.KO-12.0341 기반 표준 평가 방법을 기반으로 시험하며, 통계적 테스트를 통과한 경우에 한해 잡음원 반복사용을 허용한다.

STEP 3. (건전성 시험 적절성 확인) 주요 잡음원의 엔트로피를 고려하여 건전성 시험의 컷오프 결정하고, 구현된 건전성 시험 코드가 표준과 일치한지 확인(반복 횟수 테스트, 적응성 비율 테스트)

※ TTAK.KO-12.0235/R2 부록 참고

STEP 4. (컨디셔닝 후 엔트로피 계산)

$$h_{out}=OutputEntropy(len_{in}, n_{out}, nw, entropy_{in})$$

len_{in}, : 입력 데이터의 크기
n_{out}, : 컨디셔닝 함수의 출력 크기
nw : 컨디셔닝 함수의 내부 연산 크기
$entropy_{in}$: 입력 데이터의 엔트로피

STEP 5. (난수발생기 씨드 구성) 엔트로피 소스로 보안강도에 맞게 난수발생기 씨드 구성

항목	AS09.09
보안요구사항 개요	SSP 생성 또는 주입된 SSP로부터 유도 시 KS X ISO/IEC 19790 부속서 D의 목록에 제시된 SSP 생성 방법 사용
VE09.09.01 VE09.09.02	모듈에서 SSP를 생성하거나 주입된 SSP로부터 SSP를 유도하는 경우, 부속서 D의 목록에 제시된 SSP 생성 방법을 준수함을 명세 필요(상세 사용법도 추가 필요)

작성 예시

KISACrypto V1.0 암호모듈은 HMAC_DRBG 난수발생기를 제공하며 암호키 또는 난수생성 시에 해당 난수발생기를 이용한다. 또한 DH 기반의 키 설정의 경우, 외부에서 주입된 공개키를 이용하여 공통의 공유키를 생성한다. 암호모듈에 구현된 RSAES 기반의 키 전송 방법(Key Transport)과 DH 기반의 키 합의 방법(Key Agreement)은 ISO/IEC 19790 표준의 부속서 D에 명시된 'ISO/IEC 11770-3 Information technology - Security techniques - Key Management - Part 3: Mechanisms using asymmetric techniques.'의 'G.3 RSA key transfer'과 'D.6 Diffie-Hellman key agreement'를 준수하여 구현되었다.

난수발생기를 이용하여 SSP를 생성하는 방법에 대해서는 '난수발생기 사용 예시'를 통하여 명세하였다.

해설

KS X ISO/IEC 19790:2015의 부속서 D에는 중요 보안매개변수 생성과 설정 방법에 대해 '검증기관이 별로도 제공한다'라고 명시되어 있다. 국제표준 ISO/IEC 19790:2012에는 부속서 D가 존재하며 키 설정 방법으로 'ISO/IEC 11770-3 Information technology - Security techniques - Key Management - Part 3: Mechanisms using asymmetric techniques.'를 참조하도록 명시되어 있다. 암호모듈이 키 설정 기능을 구현할 경우에는 부속서 D에 명시된 표준 방법을 준용해야 한다.

키 설정은 통신 또는 대화에 참여하는 개체 간에 안전하게 키를 공유하기 위한 방법을 의미하며, 대표적으로 키 전송 방법과 키 합의 방법이 존재한다. 키 설정은 하나의 개체가 키를 생성한 후에 공개키 기반 알고리즘을 이용하여 나머지 개체에게 키를 전송하는 방법이며, 키 합의 방법은 공유키를 도출하는 과정에 모든 개체가 참여하는 것이다. KCMVP에서 허용된 키 설정 방법은 RSA 기반의 키 전송 방법과 DH 기반의 키 합의 방법이다. 즉 키 전송 방법으로 RSAES를 프리미티브로 사용할 수 있고, 키 합의 방법으로 DH와 ECDH 프리미티브로 사용할 수 있다.

마. 중요 보안매개변수 설정

항목	AS09.10
보안요구사항 개요	자동화된 SSP 설정 사용 시 부속서 D의 방법 준수
VE09.10.1	모든 자동화된 SSP 설정 방법에 대한 목록과 사용 방법 제시

작성 예시

KISACrypto V1.0 암호모듈은 자동화된 SSP 설정 방법으로서 RSAES 기반의 키 전달 방법과 DH 기반의 키 합의 방법을 서비스로 제공한다. 다음은 RSAES 기반의 키 전달 방법에 대한 예시이다. 먼저 개체 A에서 개체 B로 세션 암호화 키를 전송하기 위해서는 난수발생기를 이용하여 랜덤한 키를 생성해야 한다. 세션 암호를 위해 SEED-128이 사용된다고 가정하고 16바이트의 랜덤한 값을 생성한다. 다음으로 개체 B의 공개키가 요구되나, 예시를 위해 개체 B의 개인키와 공개키를 생성한다. 16바이트 세션 암호화 키를 개체 B의 공개키를 이용하여 암호화한 후 ciphertext 배열에 저장한다. 이를 네트워크를 통하여 개체 B에게 전송한다고 가정한다. 수신된 ciphertext에 대해 개체 B는 자신의 개인키를 이용하여 복호화를 수행하며 이는 recovered 배열에 저장된다. recovered 배열에 저장된 16바이트 난수값은 개체 A가 전송한 것으로서 향후 개체 A와 개체 B가 통신 세션을 암호화하는 데 사용되는 키이다.

```
int rv = 0;
int lenCipher, lenPlain;
unsigned char secretKey[16];
unsigned char ciphertext[256];
unsigned char recovered[256];

KISA_RSA_publicKey publicKey;
KISA_RSA_privateKey privateKey;

// 난수발생기로 세션 암호화키 생성
KISA_Crypto_drbgRand(secretKey, 16);

// RSAES 키 쌍생성(개체 B)
KISA_Crypto_genKeyPair(&publicKey, &privateKey, 256);

// 개체 B의 공개키로 세션 암호화 키 암호화 수행
KISA_Crypto_publicEncrypt(ciphertext, &lenCipher, secretKey, 16, &publicKey);

// ciphertext가 네트워크를 통해 전송되었다고 가정
// 개체 B의 개인키로 복호화
KISA_Crypto_privateDecrypt(recovered, &lenPlain, ciphertext, 256, &privateKey);
// recovered에 들어있는 lenPlain 바이트의 정보를 세션 암호화 키로 사용
```

```
// 사용된 메모리 해제 및 제로화
KISA_Crypto_DH_freeParam(params);
KISA_clear(&priKeyA);
KISA_clear(&pubKeyA);
KISA_clear(&priKeyB);
KISA_clear(&pubKeyB);
KISA_clear(&secretAB);
KISA_clear(&secretBA);
```

[RSAES 기반 키 설정 방법 예시]

다음은 DH 기반의 키 합의 방법의 사용 예시이다(|p|=2048-bit, |q|=256-bit). KISA_Crypto_DH_newParam() API를 이용하여 DH 키 합의 연산을 동작하기 위한 파라미터를 생성한다. 생성된 파라미터를 이용하여 KISA_Crypto_DH_genKeyPair() API를 이용하여 개체 A의 키쌍과 B의 키쌍을 생성한다. 다음으로 공유키를 계산하는 KISA_Crypto_DH_computeSharedSecret() API를 이용하여 공유키를 계산한다. 즉 개체 A에서는 자신의 개인키 A와 B의 공개키를 이용하여 공유키 secretAB를 계산하고, 개체 B는 자신의 개인키 B와 A의 공개키를 이용하여 공유키 secretBA를 계산한다. 생성된 secretAB와 secretBA를 비교하여 DH 키 합의가 정상적으로 동작했음을 확인할 수 있다.

```
// DH 파라미터 및 중요 보안매개변수 버퍼 선언
int rv = 0;

KISA_DH_Parameters *params = NULL;
BN priKeyA, priKeyB;
BN pubKeyA, pubKeyB;
BN secretAB;
BN secretBA;

BN_init(&priKeyA, 32);
BN_init(&pubKeyA, 256);
BN_init(&priKeyB, 32);
BN_init(&pubKeyB, 256);
BN_init(&secretAB, 256);
BN_init(&secretBA, 256);

// DH 파라미터 생성을 위한 메모리 할당
params = KISA_Crypto_DH_newParam();

// DH 파라미터 생성
rv = KISA_Crypto_DH_genParam(params, primePLen/8, primeQLen/8);
```

```
// DH 키쌍 생성(entity A)
rv = KISA_Crypto_DH_genKeyPair(&priKeyA, &pubKeyA, params);

// DH 키쌍 생성(entity B)
rv = KISA_Crypto_DH_genKeyPair(&priKeyB, &pubKeyB, params);

// 공유키 계산(entity A의 개인키와 entity B의 공개키 이용)
rv = KISA_Crypto_DH_computeSharedSecret(&secretAB, &priKeyA, &pubKeyB, params);

// 공유키 계산(entity B의 개인키와 entity A의 공개키 이용)
rv = KISA_Crypto_DH_computeSharedSecret(&secretBA, &priKeyB, &pubKeyA, params);

// 사용된 메모리 해제 및 제로화
KISA_Crypto_DH_freeParam(params);
KISA_clear(&priKeyA);
KISA_clear(&pubKeyA);
KISA_clear(&priKeyB);
KISA_clear(&pubKeyB);
KISA_clear(&secretAB);
KISA_clear(&secretBA);
```

[DH 기반 키 설정 중 공유된 비밀값 생성 예시]

해설

RSAES 기반의 키 전달 방법의 예시와 DH 기반의 키 합의 방법 예시를 제공한다. 위의 예시는 RSAES 기반의 키 전달 방법과 DH 기반의 키 합의 방법의 구현 형식에 따라 달라질 수 있다. 즉 API 형태, 매개변수 전달 방법에 따라 예제의 형식이 달라질 수 있으며, 개발업체가 개발한 암호 API 형식과 데이터 타입에 따라 적절히 작성하면 된다.

바. 중요 보안매개변수의 주입과 출력

항목	AS09.19
보안요구사항 개요	소프트웨어 암호모듈에 대한 주입·출력
VE09.19.01	소프트웨어 암호모듈의 경우, 평문 또는 암호화된 형태의 주입·출력방법 명세

작성 예시

KISACrypto V1.0 암호모듈은 소프트웨어 라이브러리 형태의 암호모듈이며, 응용 프로그램으로부터 API를 통해서 CSP, PSP를 평문 형식으로 입력받는다.

해설

보안수준 1과 2에 대해, 소프트웨어 암호모듈 또는 하이브리드 소프트웨어 암호모듈의 경우에는 CSP, 키 요소 및 인증 데이터가 KS X ISO/IEC 19790의 7.6.3(변경 가능한 운영환경의 요구사항)의 요구사항을 만족할 경우, 평문 또는 암호화된 형태로 주입·출력이 가능하다. 따라서 이 시험항목에서 암호모듈이 주체적으로 주입·출력하는 CSP, 키 요소 및 인증 데이터에 대해 평문 또는 암호화된 형태로 주입·출력하는 방법을 명세한다. 소프트웨어 라이브러리 형태의 암호모듈은 이 시험항목을 적용받지 않지만, 소프트웨어 하이브리드 형태의 암호모듈은 소프트웨어 구성요소와 하드웨어 구성요소 간에 CSP를 포함하는 SSP가 이동할 수 있기 때문에 이 시험항목을 적용받는다. 또한 제품형 암호모듈의 경우에도 이 시험항목을 적용받는다.

사. 중요 보안매개변수의 제로화

항목	AS09.29
보안요구사항 개요	제로화된 SSP는 복구될 수 없어야 함.
VE09.29.01	제로화된 SSP가 어떻게 복구되거나 재사용될 수 없는지 명세

작성 예시

KISACrypto V1.0 암호모듈은 보호되지 않는 SSP를 제로화하기 위해 절차적 제로화와 운영적 제로화를 적용한다. 제로화를 위해 SSP가 저장된 메모리 공간을 SSP 크기에 해당하는 바이트만큼 '0'으로 덧쓰는 방법을 통해 제로화를 수행한다. 제로화를 수행하는 기본 API는 KISA_Crypto_Zeroize()로서 컴파일러 최적화 옵션으로 인한 생략을 방지하기 위해 다음과 같이 volatile 키워드를 이용하여 최적화에 대해 예외처리될 수 있도록 한다.

```
void KISA_Crypto_Zeroize(void * ptr, int size)
{
    volatile unsigned char* p = ptr;
    while(size--)
    {
        *p++ = 0;
    }
}
```

[KISA_Crypto_Zeroize() API]

절차적 제로화의 방법은 하나의 API를 기준으로 API 내부에서 생성된 중요데이터, SSP와 관련된 매개변수를 복사하거나 SSP로부터 도출된 데이터를 API를 반환하기 전에 제로화하는 것을 의미한다. 또한 암호연산 수행 시 오류가 발생한 경우 관련된 데이터를 제로화하는 것도 절차적 제로화에 해당한다. 절차적 제로화의 기본 형태는 다음과 같다.

```
RET func_name(param1, param2, ..., paramN)
{
    var1;
    var2;
    ...
    varN;

    // 매개변수를 var1~varN에 복사
    // 또는 매개변수로부터 도출된 값이 var1~varN에 저장
    // 암호 서비스 수행
    // var1~varN에 대한 제로화 수행

    return;
}
```

[절차적 제로화 기본 형태]

이 암호모듈이 각 암호 서비스에 대해 수행하는 절차적 제로화 대상은 다음과 같다.

서비스	SSP	절차적 제로화 대상
블록암호	라운드 키 및 암호연산 관련 임시 데이터	– KISA_Crypto_encryptFinal() 내에서 사용이 완료된 라운드 키에 대한 제로화 수행 – KISA_Crypto_decryptFinal() 내에서 사용이 완료된 라운드 키에 대한 제로화 수행 – KISA_Crypto_encrypt() 내에서 사용이 완료된 라운드 키에 대한 제로화 수행 – KISA_Crypto_decrypt() 내에서 사용이 완료된 라운드 키에 대한 제로화 수행 – 위에 명시된 API 내부에서 임시로 생성되거나 복사된 데이터에 대해 API 반환 전 제로화 수행 – 오류 발생 시 연관된 데이터 제로화

서비스	SSP	절차적 제로화 대상
HMAC	keyed ipad/opad 및 암호연산 관련 임시 데이터	– KISA_Crypto_hmacFinal() 내에서 HMAC 키와 XOR된 ipad와 opad에 대한 제로화 – KISA_Crypto_hmac() 내에서 HMAC 키와 XOR된 ipad와 opad에 대한 제로화 – 위에 명시된 API 내부에서 임시로 생성되거나 복사된 데이터에 대해 API 반환 전 제로화 수행 – 오류 발생 시 연관된 데이터 제로화
HMAC_DRBG	엔트로피 잡음원 및 암호연산 관련 임시 데이터	– KISA_Crypto_drbgInit() 내에서 난수발생기 내부 상태를 구성한 후, 수집된 잡음원 정보 제로화 – KISA_Crypto_drbgReseed() 내에서 난수발생기 내부 상태를 구성한 후, 수집된 잡음원 정보 제로화 – 오류 발생 시 연관된 데이터 제로화
RSAES	난수 및 암호연산 관련 임시 데이터	– KISA_Crypto_genKeyPair() 내에서 키쌍 생성 시 임시로 생성되어 사용된 난수값 및 데이터 제로화 – KISA_Crypto_publicEncrypt() 내에서 랜덤 시드 및 암호화 관련 임시 데이터 제로화 – KISA_Crypto_privateDecrypt() 내에서 복호화 관련 임시 데이터 – 오류 발생 시 연관된 데이터 제로화
RSA-PSS	난수 및 암호연산 관련 임시 데이터	– KISA_Crypto_genKeyPair() 내에서 키쌍 생성 시 임시로 생성되어 사용된 난수값 및 데이터 제로 – KISA_Crypto_sign() 내에서 랜덤 솔트 값 및 서명 생성 시 생성된 임시 데이터 제로화 – KISA_Crypto_verify() 내에서 서명 검증 시 생성된 임시 데이터 제로화
DH	난수 및 암호연산 관련 임시 데이터	– KISA_Crypto_DH_genParameter() 내에서 파라미터 생성을 위해 사용되는 난수 및 임시 데이터 제로화 (J, Seed, Count, U 등) – KISA_Crypto_DH_genKeyPair() 내에서 키쌍 생성을 위해 사용되는 임시 데이터 제로화 – KISA_Crypto_DH_generateSecret() 내에서 공유키 생성을 위해 사용되는 임시 데이터 제로화

[절차적 제로화 대상]

운영적 제로화는 long-term으로 사용되는 SSP에 대해 응용 프로그램에서 사용이 완료된 후 제로화할 수 있도록 하는 API를 제공하는 것을 의미한다. long-term으로 사용되는 SSP에 대해 제로화할 수 있도록 하는 API의 목록은 다음과 같다. 6번의 바이트 배열에 저장된 난수를 제로화할 수 있는 있는 KISA_Crypto_Zeroize()가 기본이며, 다수의 바이트 배열로 구성된 공개키/개인키, 공개키 파라미터 등은 이를 확장한 형태의 제로화 API이다.

1.RSAES/RSA-PSS 공개키에 대한 제로화: KISA_Crypto_RSA_publicKey_Zeroize()
2.RSAES/RSA-PSS 개인키에 대한 제로화: KISA_Crypto_RSA_privateKey_Zeroize()
3.ECDSA 공개키에 대한 제로화: KISA_Crypto_EC_publicKey_Zeroize()
4.ECDSA 개인키에 대한 제로화: KISA_Crypto_EC_privateKey_Zeroize()
5.DH 파라미터에 대한 제로화: KISA_Crypto_DH_param_Zeroize()
6.DH 공개키에 대한 제로화: KISA_Crypto_DH_publicKey_Zeroize()
7.DH 개인키에 대한 제로화: KISA_Crypto_DH_privateKey_Zeroize()
8.난수에 대한 제로화: KISA_Crypto_Zeroize()

해설

이 시험항목에서는 암호모듈에서 적용하고 있는 제로화 방법에 대해 명세할 것을 요구한다. 암호모듈에서 적용하고 있는 절차적 제로화 방법과 운영적 제로화 방법에 대해 명세한다. 추가로 제로화 방법이 적용된 소스코드의 위치까지 제공하면 명확하게 제로화에 대한 증빙이 가능하다.

9 자가시험

가. 목적

자가시험(selftest)은 운영자에게 암호모듈의 정상적인 동작을 방해하는 오류가 발생하지 않았음을 보증할 수 있는 수단이며 동작 전 자가시험(pre-operational selftest)과 조건부 자가시험(conditional selftest)으로 구성된다. 모든 자가시험은 외부의 제어 없이 암호모듈 내부에 의해서만 성공/실패 여부가 결정되어야 한다. 동작 전 자가시험은 암호모듈이 데이터 출력 인터페이스를 통해 데이터 출력을 수행하기 전에 수행되어야 하며, 조건부 자가시험은 보안 함수 또는 프로세스가 실행될 때 수행되어야 한다(즉 사용되는 암호알고리즘에 대해 조건부 자가시험이 요구된다).

검증대상 암호알고리즘에 대한 자가시험은 모두 암호모듈 내부에 구현되어야 한다. 암호모듈은 KS X ISO/IEC 19790:2015, KS X ISO/IEC 24759:2015에서 명시된 것 이외의 다른 형태의 동작 전 자가시험과 조건부 핵심 기능 시험도 수행이 가능하다. 자가시험에 실패할 경우 암호모듈은 오류 상태로 진입해야 하며, 표시기를 통해 오류 상태임을 알려야 한다. 오류 상태에서 암호모듈은 암호 서비스를 제공할 수 없다. 자가시험을 구성하는 동작 전 자가시험과 조건부 자가시험에 대해, 동작 전 자가시험은 암호모듈이 정상적인 서비스를 수행하기 위해 암호모듈 초기화 과정에서 수행되어야 하며, 조건부 자가시험은 조건이 충족된 경우에 수행되어야 한다(예: 특정 알고리즘이 처음 호출되어 사용되는 경우, 공개키 키쌍을 생성하는 경우 등).

다음은 동작 전 자가시험과 조건부 자가시험의 종류를 제시한다. 소프트웨어/펌웨어 암호모듈의 경우 동작 전 자가시험의 '동작 전 소프트웨어/펌웨어 무결성 시험'을 반드시 수행해야 한다. 암호알고리즘이 최초 호출되어 사용되는 경우, 알고리즘의 정상동작을 '조건부 암호알고리즘 자가시험'을 통해 확인해야 한다. 그러나 해당 시험은 '동작 전 핵심기능 시험'에서 모든 알고리즘에 대해 일괄적으로 수행 가능하다. 공개키 기반 암호알고리즘의 경우 키쌍을 생성하는 경우에는 '조건부 키쌍 일치 시험'을 수행하여 생성된 키쌍의 유효성을 검증해야 한다. 난수발생기가 엔트로피 잡음원을 수집하는 경우 (초기화 단계, 리씨드 단계)에는 수집된 잡음원에 대해 조건부 핵심 기능 시험의 일부로 "조건부 엔트로피 건전성 시험"을 수행해야 한다.

자가시험 종류	세부 자가시험 종류
동작 전 자가시험	- 동작 전 소프트웨어/펌웨어 무결성 시험 - 동작 전 핵심 기능 시험 - 동작 전 우회 시험
조건부 자가시험	- 조건부 암호알고리즘 자가시험 - 조건부 키쌍 일치 시험 - 조건부 소프트웨어/펌웨어 로드 시험 - 조건부 수동 주입 시험 - 조건부 우회 시험 - 조건부 핵심 기능 시험(조건부 엔트로피 건전성 시험) - 주기적 자가시험

[자가시험 종류]

이 장에서는 자가시험의 주요 보안요구사항을 만족시키기 위해 개발업체가 작성해야 하는 AS 항목별 VE에 대한 작성 가이드를 제시한다.

나. 기본사항

항목	AS10.07
보안요구사항 개요	자가시험 실패 시 오류 처리
VE10.07.01	오류 발생 조건, 명칭, 해제 방법에 대한 명세
작성 예시	

KISACrypto V1.0 암호모듈은 블록암호, 해시함수, 메시지인증코드, 난수발생기, 공개키 암호, 전자서명, 키설정 알고리즘을 탑재하고 있으며, 암호모듈이 초기화되는 과정에서 수행하는 동작 전 자가시험과 특정 조건이 발생했을 때 수행하는 조건부 자가시험을 구현하고 있다.

KISACrypto V1.0 암호모듈은 검증대상 암호알고리즘으로서 블록암호 SEED, 해시함수 SHA-256, HMAC-SHA-256, HMAC_DRBG(SHA-256), RSAES(2048, SHA-256), RSA-PSS (2048, SHA-256), DH(2048-bit, 256-bit)를 탑재했다고 가정한다. 암호모듈에서 구현하고 있는 동작 전 자가시험은 다음과 같다. 동작 전 자가시험에서 암호모듈에 대한 무결성 시험과 탑재한 검증대상 암호알고리즘에 대한 조건부 암호알고리즘 시험을 핵심 기능 시험으로서 수행한다. 핵심 기능 시험은 사전에 암호모듈에 하드코딩되어 있는 테스트벡터를 이용한 KAT(Known Answer Test) 형태로 수행된다.

동작 전 자가시험 세부 시험	자가시험 세부 내용	오류 발생 조건, 오류코드, 해제 방법
동작 전 소프트웨어/펌웨어 무결성 시험	– 소프트웨어/펌웨어 암호모듈에 대한 무결성 시험 수행(사용되는 검증대상 무결성 알고리즘에 대한 알고리즘 시험 선행) – 사전에 계산되어 암호모듈에 연접되어 있는 무결성 검증값과 암호모듈이 동작할 때 새롭게 계산한 무결성 검증값을 비교하는 형태로 무결성 시험 수행 – 오류 발생 시 심각한 오류 상태로 자동 천이	– 암호모듈의 바이너리 또는 무결성 검증값이 변조될 경우 오류 발생 – KISA_PreSelfTest_Integrity_Fail – 암호모듈 재시작 또는 재설치
동작 전 핵심 기능 시험	– SEED 운영모드에 대한 KAT 시험 수행(지원하는 모든 키사이즈, 운영모드에 대해 알고리즘 시험 수행) – 운영모드에 대한 알고리즘 시험을 수행하기 위해 다중블록 형태의 테스트벡터 이용 – 암호화 KAT 시험 수행(저장된 평문, 키, IV/CTR을 이용하여 암호문을 계산한 후 저장된 암호문과 비교) – 복호화 KAT 시험 수행(계산된 암호문에 대해 다시 복호화를 수행한 후 저장된 평문과 비교)	– 암호화 KAT 또는 복호화 KAT 실패 – KISA_PreSelfTest_BlockCipher_SEED_Fail – 암호모듈 재시작 또는 재설치
	– 해시함수에 대한 KAT 시험 수행 – 가변 길이의 메시지에 대해 해시값을 적절히 계산하는지 확인하기 위해 기본 블록 길이보다 짧은 메시지와 블록 길이보다 긴 메시지의 테스트벡터 이용 – 해시 KAT 수행(저장된 메시지에 대해 해시값을 계산한 후 저장된 해시값과 비교)	– 해시함수 KAT 실패 – KISA_PreSelfTest_Hash_SHA_Fail – 암호모듈 재시작 또는 재설치
	– HMAC에 대한 KAT 시험 수행 – 가변 길이의 메시지에 대해 해시값을 적절히 계산하는지 확인하기 위해 기본 블록 길이보다 짧은 메시지와 블록 길이보다 긴 메시지의 테스트벡터 이용 – HMAC KAT 수행(저장된 메시지와 MAC 키를 이용하여 MAC값을 계산한 후 저장된 MAC값과 비교)	– HMAC KAT 실패 – KISA_PreSelfTest_MAC_HMAC_Fail – 암호모듈 재시작 또는 재설치
	– HMAC_DRBG (SHA-256)에 대한 KAT 시험 수행 – 난수발생기에 대한 KAT이기 때문에 잡음원을 수집하지 않고 사전에 저장된 엔트로피 잡음원 이용 – DRBG KAT 수행(저장된 엔트로피 잡음원, 논스, 개별화 문자열을 이용하여 난수값 출력 후 사전에 저장된 난수값과 비교)	– HMAC_DRBG KAT 실패 – KISA_PreSelfTest_DRBG_HMACDRBG_Fail – 암호모듈 재시작 또는 재설치

동작 전 자가시험 세부 시험	자가시험 세부 내용	오류 발생 조건, 오류코드, 해제 방법
동작 전 핵심기능 시험	- RSAES에 대한 KAT 시험 수행 - RSAES에 대한 KAT이기 때문에 OAEP 인코딩 과정 중에 랜덤 시드값을 생성하는 것이 아닌 사전에 저장된 랜덤 시드값 이용 - 암호화 KAT 시험 수행(사전에 저장된 모듈러스, 공개키, 시드값, 평문에 대해 암호화를 수행한 후 사전에 저장된 암호문과 비교) - 복호화 KAT 시험 수행(생성된 암호문에 대해 사전에 저장된 개인키를 이용하여 복호화 수행한 후 평문과 비교)	- RSAES 암호화 KAT 또는 복호화 KAT 실패 - KISA_PreSelfTest_Public_RSAES_Fail - 암호모듈 재시작 또는 재설치
	- RSA-PSS에 대한 KAT 시험 수행 - RSA-PSS에 대한 KAT이기 때문에 메시지 인코딩 과정 중에 랜덤 솔트값을 생성하는 것이 아닌 사전에 저장된 랜덤 솔트값 이용 - 서명 KAT 시험 수행(사전에 저장된 모듈러스, 개인키, 솔트값, 평문에 대해 서명을 생성한 후 사전에 저장된 서명값과 비교) - 서명검증 KAT 시험 수행(생성된 서명에 대해 사전에 저장된 공개키와 평문을 이용하여 서명 검증 수행)	- RSA-PSS 서명 KAT 또는 검증 KAT 실패 - KISA_PreSelfTest_Digital_RSAPSS_Fail - 암호모듈 재시작 또는 재설치
	- DH에 대한 KAT 시험 수행 - DH에 대한 KAT이기 때문에 하나의 도메인 파라미터에 대해 개체 A의 개인키와 개체 B의 공개키를 사전에 저장 - DH 키 합의 KAT 시험 수행(도메인 파라미터, 개체 A의 개인키, 개체 B의 공개키를 이용하여 공유키를 계산하여 사전에 저장된 공유키와 비교)	- DH 키 합의 KAT 실패 - KISA_PreSelfTest_KA_DH_Fail - 암호모듈 재시작 또는 재설치

[KISACrypto V1.0 동작 전 자가시험 목록]

암호모듈에서 구현하고 있는 조건부 자가시험은 다음과 같다. 공개키 기반 암호알고리즘 (공개키 암호, 전자서명, 키설정)의 키 쌍을 생성하는 경우, 생성된 키 쌍에 대해 다음에 명세된 시험을 수행한다. 또한, 난수발생기에서 엔트로피 잡음원을 수집하는 경우, 수집된 잡음원에 대해 조건부 핵심 기능 시험(조건부 엔트로피 건전성 시험)을 수행한다. 즉,조건부 핵심 기능 시험(조건부 엔트로피 건전성 시험)의 경우 잡음원 별로 반복 횟수 테스트와 적응성 비율 테스트를 적용한다.

조건부 자가시험 세부 시험	자가시험 세부 내용	오류 발생 조건, 오류코드, 해제 방법
조건부 키쌍 일치 시험	– RSAES에 대한 키쌍 일치 시험 수행 – 생성된 키쌍에 대해 사전에 정의된 메시지를 이용하여 암호화를 수행하여 원본 메시지와 암호문 비교 – 암호문을 복호화하여 원본 메시지와 비교	– 암호문이 원본 메시지와 동일하거나 복호화된 결과가 원본 메시지와 다를 경우 오류 발생 – KISA_CondTest_RSAES_PairKey_Fail – 암호모듈 재시작 또는 재설치
	– RSA-PSS에 대한 키쌍 일치 시험 수행 – 생성된 키쌍에 대해 사전에 정의된 메시지를 이용하여 서명을 생성. 생성된 서명과 원본 메시지 비교 – 생성된 서명에 대한 검증 수행	– 서명이 원본 메시지와 동일하거나 서명 검증이 실패할 경우 오류 발생 – KISA_CondTest_RSAPSS_PairKey_Fail – 암호모듈 재시작 또는 재설치
	– DH에 대한 키쌍 일치 시험 수행 – 생성된 키쌍에 대해 추가로 하나의 키쌍을 더 생성하여 키 합의 과정 수행(추가 키쌍의 경우에는 작은 지수값을 이용) – 키 합의 과정을 통해 생성된 공유키를 비교	– 두 개의 공유키가 일치하지 않을 경우 오류 발생 – KISA_CondTest_DH_PairKey_Fail – 암호모듈 재시작 또는 재설치
조건부 핵심기능 시험의 일부 (조건부 엔트로피 건전성 시험)	– HMAC_DRBG 난수발생기 조건부 엔트로피 – 잡음원을 수집하는 초기화 함수, 리시드 함수에서 수집된 잡음원에 대한 건전성 시험(반복 횟수 테스트, 적응성 비율 테스트) 수행(잡음원별 건전성 함수 적용)	– 건전성 시험 실패 시 오류 발생 – KISA_CondTest_DRBG_Health_Fail – 암호모듈 재시작 또는 재설치

[KISACrypto V1.0 조건부 자가시험 목록]

해설

암호모듈이 수행하고 있는 동작 전 자가시험과 조건부 자가시험에 대한 명세를 한다. 소프트웨어/펌웨어 암호모듈의 경우에는 동작 전 자가시험에서 반드시 동작 전 소프트웨어/펌웨어 무결성 시험을 수행해야 한다. 조건부 자가시험의 경우 특정 알고리즘이 실행되거나 특정 조건이 발생한 경우에 수행되는 시험이다. 조건부 암호알고리즘 시험의 경우, 암호모듈에 탑재된 검증대상 암호알고리즘이 최초 수행되는 경우 KAT 형태로 수행된다. 그러나 동작 전 핵심 기능 시험에서 탑재한 알고리즘에 대해 KAT 시험을 일괄적으로 수행할 수 있다. 동작 전 핵심 기능 시험에서 암호알고리즘 시험을 수행한 경우, 조건부 암호알고리즘 시험을 생략할 수 있다. 공개키 기반 암호알고리즘에서 새로운 키쌍을 생성한 경우 조건부 키쌍 일치시험을 수행한다. 또한 난수발생기에서 엔트로피 잡음원을 새롭게 수집한 경우, 건전성 시험을 수행하여 수집된 잡음원의 유효성을 검증한다. 이 시험항목에서는 위에서 설명한 동작 전 자가시험과 조건부 자가시험의 종류, 수행 방법, 오류 발생 조건, 오류 해제 조건을 명세하는 것을 요구한다.

항목	AS10.08
보안요구사항 개요	자가시험 실패 시 표시기를 통한 오류 표시
VE10.08.01	각 자가시험에 따른 오류 상태 명세와 오류 표시기 명세
작성 예시	
KISACrypto V1.0 암호모듈은 'KISACrypto V1.0 동작 전 자가시험 목록'과 'KISACrypto V1.0 조건부 자가시험 목록'을 통하여 세부적인 자가시험에 대한 내용과 실패 조건, 실패에 따른 오류코드를 정의하였다. 자가시험 실패에 따른 상태 표시기가 설계서에 명세된 오류코드와 일치하는지의 여부는 암호모듈 시험서를 통해 제시한다.	
해설	
이 시험항목은 AS10.07의 VE10.07에서 함께 명세될 수 있다. 자가시험이 실패할 경우, 실패한 자가시험의 종류를 확인할 수 있는 오류코드가 설계서에 명세되어 있는지와 실제 자가시험 실패 시 암호모듈로부터 반환되는 상태코드가 설계서에 명세된 오류코드와 일치하는지의 여부를 요구하는 시험항목이다.	

항목	AS10.09
보안요구사항 개요	오류 상태 시 암호연산 및 데이터 출력 금지
VE10.09.01	오류 상태 시 암호연산, 제어 출력 및 데이터 출력 인터페이스를 통한 출력 금지 방법 명세
작성 예시	
KISACrypto V1.0 암호모듈은 오류 상태를 심각한 오류와 단순한 오류로 구분한다. 단순한 오류는 사용자의 암호 서비스 사용의 실수로 발생하는 오류로서 오류의 종류를 명시하는 상태출력을 수행한 후, 정상동작 상태(검증대상 동작 모드 상태)로 자동천이된다. 심각한 오류는 자가시험 실패로 발생하는 오류이며, 암호모듈의 재시작 또는 재설치를 통해 해결될 수 있다. 심각한 오류 상태에서는 암호모듈은 암호 서비스를 운영자에게 제공할 수 없다. '암호 서비스 수행 시 오류 상태에 대한 데이터 출력 금지 방법 명세'를 통해, 오류 상태에서는 암호모듈이 암호 서비스를 수행하지 않으며, 또한 데이터 출력을 금지하고 있음을 명세하였다. 즉, 암호 서비스를 수행하는 외부 API가 호출되면 내부에서 암호모듈의 현재 상태를 체크한 후 정상동작 상태에서만 암호 서비스를 수행한다. 암호 서비스를 수행하던 중 오류가 발생하면 관련된 데이터를 모두 제로화한 후 오류코드만을 반환한다.	
해설	
이 시험항목의 VE10.09.01은 VE03.07.01, VE03.07.02, VE03.10.01, VE03.10.02와 함께 명세할 수 있다.	

항목	AS10.10
보안요구사항 개요	자가시험 오류 시 암호 서비스 제한
VE10.10.01	자가시험 오류가 해결되기 전, 암호 서비스를 제한하는 방법 명세
작성 예시	
KISACrypto V1.0 암호모듈은 자가시험(동작 전 자가시험, 조건부 자가시험) 실패 시 심각한 오류로 자동 천이된다. 암호모듈은 현재 상태를 저장할 수 있는 내부 변수를 유지하고 있으며, 암호모듈의 서비스 연산 결과에 따라서만 내부 상태가 변경된다. 자가시험 실패 시 내부 상태를 심각한 오류 상태로 변경하며, 이후의 어떤 암호 서비스도 수행하지 않는다. 즉 암호 서비스 수행 시에 내부 변수를 이용하여 암호모듈의 상태를 파악하여 심각한 오류 상태의 경우에는 즉각 반환 되도록 한다. 암호모듈의 재구동 또는 재설치를 통해 암호모듈의 상태가 정상동작 상태인 경우에만 암호 서비스가 수행되도록 한다.	
해설	
이 시험항목은 AS10.09의 VE10.09.01과 함께 명세할 수 있다.	

항목	AS10.11
보안요구사항 개요	자가시험 실패 시 오류 상태 미출력에 대한 개발문서에 명세
VE10.11.01	자가시험 실패 시 오류 상태 미출력에 대한 암호모듈이 오류 상태에 진입했는지를 결정할 수 있는 명백한 절차를 명세
작성 예시	
KISACryptop V1.0 암호모듈은 자가시험 실패에 대해 별도로 오류 상태를 출력하지 않는다. 각각의 자가시험을 실행하여 암호모듈이 모든 오류 상태에 진입하게 하여 암시적으로 오류 상태에 진입했음을 확인할 수 있도록 하는 절차와, 그 때 암호 서비스가 수행 불가함을 확인할 수 있는 소스코드를 명세해야 한다.	
해설	
암호모듈이 자가시험 실패에 대하여 오류 상태를 출력하지 않는다면, 암호모듈이 오류 상태에 진입했는지를 결정할 수 있는 명백한 절차를 명세한다.	

다. 동작 전 자가시험

항목	AS10.15
보안요구사항 개요	동작 전 자가시험 수행 필요
VE10.15.01 VE10.15.02	설계서에 동작 전 자가시험에 대한 정보 명세
작성 예시	
이 암호모듈에서 수행하는 동작 전 자가시험은 설계서의 'KISACrypto V1.0 동작 전 자가시험 목록'에서 확인할 수 있다. 크게 동작 전 소프트웨어 무결성 시험과 동작 전 핵심 기능 시험을 수행한다. 동작 전 자가시험은 암호모듈의 초기화 과정에서 수행되며, 순서대로 소프트웨어 무결성 시험과 핵심 기능 시험을 수행한다. 모든 동작 전 자가시험이 성공으로 끝난 경우에만 암호모듈은 정상동작 상태로 운영자에게 암호 서비스를 제공할 수 있다.	
해설	
이 시험항목은 AS10.07의 VE10.07.01에서 암호모듈의 동작 전 자가시험과 조건부 자가시험에 대해 명세해야 한다.	

항목	AS10.17
보안요구사항 개요	동작 전 소프트웨어/펌웨어 무결성 시험
VE10.17.01 VE10.17.02 VE10.17.03 VE10.17.04	소프트웨어/펌웨어 무결성 시험에 대한 상세 명세
작성 예시	
KISACrypto V1.0 암호모듈은 소프트웨어 암호모듈로서 자체적인 무결성 검증 기술을 이용하여 모든 소프트웨어 구성요소에 대해 무결성 검증을 수행한다. 동작 전 소프트웨어 무결성 시험에 대한 상세한 내용은 설계서의 소프트웨어/펌웨어 보안에서 명세하였으며, 세부적인 과정은 'HMAC을 이용한 무결성 검증값 생성 및 최종 암호모듈 생성 과정', 'HMAC 기반의 암호모듈 무결성 검증 과정', '암호모듈 무결성 시험 수도코드'를 이용하여 명세하였다.	
해설	
이 시험항목은 AS05.05의 VE05.05.01~VE05.05.04, AS05.06의 VE05.06.01로서 명세될 수 있다.	

항목	AS10.20
보안요구사항 개요	소프트웨어/펌웨어 시험 전 무결성 기술에 대한 자가시험 수행
VE10.20.01	소프트웨어/펌웨어 시험에 사용된 검증대상 무결성 기술에 대해 자가시험 선행에 대한 명세

작성 예시

KISACrypto V1.0 암호모듈은 동작 전 소프트웨어 시험 수행 전, 사용된 검증대상 무결성 기술에 대한 암호알고리즘 시험을 핵심 기능 시험으로서 수행한다. 해당 내용은 설계서의 'KISACrypto V1.0 동작 전 자가시험 목록'에 명세되어 있다.

해설

동작 전 소프트웨어/펌웨어 무결성 시험을 수행하기 전, 사용된 무결성 검증기술에 대해 자가시험을 수행하여 알고리즘의 정상동작 여부를 확인해야 한다. 즉 동작 전 자가시험의 구성은 다음과 같이 할 수 있다.

1. 동작 전 소프트웨어/펌웨어 무결성 시험에 사용된 무결성 기술에 대한 핵심기능시험을 포함한 모든 암호알고리즘 자가시험(KAT 시험 등)
2. 동작 전 소프트웨어/펌웨어 무결성 시험

또는

1. 동작 전 소프트웨어/펌웨어 무결성 시험에 사용된 무결성 기술에 대한 핵심 기능 시험(KAT 시험)
2. 동작 전 소프트웨어/펌웨어 무결성 시험
3. 나머지 알고리즘에 대한 핵심 기능 시험(KAT 시험)

항목	AS10.24
보안요구사항 개요	동작 전 핵심 기능 시험 명세
VE10.24.01	동작 전 핵심 기능 시험에 대한 명세

작성 예시

KISACrypto V1.0 암호모듈은 동작 전 핵심 기능 시험으로서 조건부 암호알고리즘 시험을 수행한다. 즉 탑재하고 있는 검증대상 암호알고리즘에 대해 KAT 형태의 자가시험을 수행한다. 암호모듈이 수행하고 있는 동작 전 핵심 기능 시험의 목록은 설계서 'KISACrypto V1.0 동작 전 자가시험 목록'에서 확인 가능하다. 다음은 동작 전 핵심 기능 시험으로 수행되는 알고리즘별 KAT 시험항목, 방법, 참조 표준, 관련 소스파일과 함수에 대한 명세이다. 알고리즘 KAT 시험에서 사용되는 테스트벡터 모두 관련표준을 참조한 것이며, 소스코드에 하드코딩되어 있다.

조건부 알고리즘 시험	수행 방법	참조 표준, 관련 소스파일/함수
SEED	– SEED-128 ECB, CBC, CTR, CFB128, OFB 운영모드 – 참조 표준에 명시되어 있는 키, 평문, 암호화 KAT, 복호화 KAT 수행	– KS X 3254(2016), KS X 3276(2019) – kisa_blockcipher_selftest.c (seed_kat_test())
SHA-256	– SHA-256의 경우, 참조 표준 (p.42)에 명시되어 있는 56바이트/62바이트 2개의 메시지에 대해 해시 KAT 수행	– ISO/IEC 10118-3(2018) – kisa_hash_selftest.c (sha_kat_test())
HMAC-SHA-256	– HMAC-SHA-256의 경우, 참조 표준(p.5)에 명시되어 있는 암호키와 메시지에 대해 HMAC KAT 수행	– TTAK.KO-12.0330-Part 2 – kisa_mac_selftest.c (hmac_kat_test())
HMAC_DRBG (SHA-256)	– 참조 표준(p.6)에 명시되어 있는 엔트로피, 논스, 개별화 문자열, 추가 입력 등의 데이터에 대해 시나리오 2-1 (개별화 문자열 사용, 추가 입력 사용), 시나리오 2-2 (개별화 문자열 미사용, 추가 입력 사용), 시나리오 2-3 (개별화 문자열 사용, 추가 입력 미사용), 시나리오 2-4 (개별화 문자열 미사용, 추가 입력 미사용) – 참조 표준에 명시된 대로 갱신주기를 2로 하여 KAT 수행	– TTAK.KO-12.0332-Part 2 – kisa_drbg_selftest.c (hmacdrbg_kat_test())
RSAES-2048 (SHA-256)	– CAVP 과정에서 통과한 공개키/개인키, 메시지, 랜덤 시드값을 이용한 암호화 KAT, 복호화 KAT 수행	– kisa_public_selftest.c (rsaes_kat_test())
RSA-PSS-2048 (SHA-256)	– CAVP 과정에서 통과한 공개키/개인키, 메시지, 랜덤 솔트값을 이용한 서명 생성 KAT, 검증 KAT 수행	– kisa_public_selftest.c (rsapss_kat_test())
DH (|p|=2048-bit, |q|=256-bit)	– CAVP 과정에서 통과한 DH 파라미터, 개인키/공개키 쌍에 대해 키 합의 KAT 수행	– kisa_ka_selftest.c (dh_kat_test())

[KISACrypto V1.0 암호알고리즘 시험 상세 명세]

알고리즘별 KAT 시험에 대한 상세코드는 위 표에서 제시한 소스코드의 해당 함수를 통해 파악 가능하다. 다음은 대표적으로 SEED에 대한 암호화 KAT와 복호화 KAT에 대한 코드 예시이다.

```
// 테스트벡터 구조체 정의
typedef struct _SEED_TV_ {
        int mode;
        unsigned char MK[32];
        int lenOfMK;
        unsigned char PT[128];
        int lenOfPT;
        unsigned char CT[144];
        int lenOfCT;
        unsigned char IV[16];
}SEED_TV;

const SEED_TV seedTestVectors[] = { { // 운영모드(예: SEED_CBC_MODE),
                        { // 마스터키 }, //마스터키 길이,
                                { // 평문 }, // 평문 길이,
                                { // 암호문 }, // 암호문 길이,
                                { // IV 또는 CTR } } };

// SEED KAT 수행 함수
int seed_kat_test()
{
        unsigned char rk[16 * 17] = { 0x00, };
        unsigned char CT[144] = { 0x00, };
        unsigned char recovered[128] = { 0x00, };
        int ret = SUCCESS;
        int cnt_i = 0;

        // 테스트벡터 수만큼 KAT 수행
        for (cnt_i = 0; cnt_i < sizeof(seedTestVectors) / sizeof(SEED_TV); cnt_i++)
        {
       // SEED 암호화 KAT 수행
       seed_encrypt(SEED_ENCRYPT,
              seedTestVectors[cnt_i].mode,
              seedTestVectors[cnt_i].PT,
              seedTestVectors[cnt_i].lenOfPT,
              seedTestVectors[cnt_i].MK,
              seedTestVectors[cnt_i].lenOfMK * 8, CT);

                  if (memcmp(seedTestVectors[cnt_i].CT, CT, 16)) {
                        ret = FAIL;
                        goto END;
                  }

       // SEED 복호화 KAT 수행
              seed_encrypt(SEED_DECRYPT,
              seedTestVectors[cnt_i].mode,
              CT,
              seedTestVectors[cnt_i].lenOfPT,
              seedTestVectors[cnt_i].MK,
              seedTestVectors[cnt_i].lenOfMK * 8,
              recovered);
```

```
if (memcmp(seedTestVectors[cnt_i].PT, recovered, 16)) {
                    ret = FAIL;
                    goto END;
                }
        }

END:
        // 사용된 변수에 대한 제로화 수행
        memset(rk, 0, 16 * 17);
        memset(CT, 0, 144);

        return ret;
}
```

[SEED 알고리즘 KAT 시험 코드 예시]

해설

동작 전 핵심 기능 시험은 오류발생 시 CSP의 노출을 초래할 수 있는 기능이나, KCMVP에서는 동작 전 핵심 기능 시험에서 조건부 암호알고리즘 시험을 수행할 수 있다. 즉 동작 전 핵심 기능 시험에서 암호모듈이 탑재하고 있는 알고리즘에 대해 KAT 형태의 암호알고리즘 시험을 일괄적으로 수행할 경우, 조건부 자가시험을 생략할 수 있다. KAT 형태의 암호알고리즘 시험에 대한 명세 시 상세한 KAT 시험 방법(사용된 테스트벡터 출처, 소스파일/관련 함수, KAT 시험 형태, 수도코드 등)을 제시하는 것이 바람직하다.

라. 조건부 자가시험

항목	AS10.25
보안요구사항 개요	조건부 자가시험 수행 필요
VE10.25.01	조건부 자가시험에 대한 명세

작성 예시

KISACrypto V1.0 암호모듈은 조건부 암호알고리즘 시험과 조건부 엔트로피 건전성 시험은 동작 전 핵심 기능 시험으로 수행하기 때문에 실제로 수행하는 조건부 자가시험은 조건부 키쌍 일치 시험과 조건부 핵심 기능 시험(조건부 엔트로피 건전성 시험)이다. 암호모듈이 수행하는 조건부 자가시험에 대한 목록은 설계서의 'KISACrypto V1.0 조건부 자가시험 목록'에 명세되어 있다. 조건부 키쌍 일치 시험은 공개키 기반 암호(공개키 암호, 전자서명, 키 설정)에서 사용되는 키쌍을 생성하는 경우, 유효성을 확인하는 시험이다. 이 암호모듈에서 키쌍을 생성할 때는 'SSP 생성 방법 및 난수발생기 사용 명세'에 명세된 대로 표준에 의거하여 키쌍을 생성한다. 생성된 키쌍에 대해 암호화→복호화, 서명 생성→서명 검증, 키 합의 형태의 시험을 수행하여 키쌍이 유효한지를 확인한다.

조건부 키쌍 일치 시험	수행 방법	관련 소스파일/함수
RSAES 키쌍 일치 시험	– KISA_Crypto_genKeyPair() 내부에서 생성된 RSAES 키쌍에 대해 키쌍 일치 시험 수행 – 시험을 위해 생성된 키쌍을 매개변수로 하여 rsaes_pairwise_test() 호출 – rsaes_pairwise_test() 내부에서 고정된 평문에 대해 공개키로 암호문 생성(평문과 동일한 경우 오류 처리) – 생성된 암호문에 대해 개인키로 복호화 수행(복호문이 평문과 동일할 경우 성공)	– kisa_public_selftest.c (rsaes_pairwise_test())
RSA-PSS 키쌍 일치 시험	– KISA_Crypto_genKeyPair() 내부에서 생성된 RSAES 키쌍에 대해 키쌍 일치 시험 수행 – 시험을 위해 생성된 키쌍을 매개변수로 하여 rsaes_pairwise_test() 호출 – rsaes_pairwise_test() 내부에서 고정된 평문에 대해 공개키로 암호문 생성(평문과 동일한 경우 오류 처리) – 생성된 암호문에 대해 개인키로 복호화 수행(복호문이 평문과 동일할 경우 성공)	
DH 키쌍 일치 시험	– KISA_Crypto_DH_genKeyPair() 내부에서 생성된 DH 키쌍에 대해 키쌍 일치 시험 수행 – 시험을 위해 생성된 파라미터와 키쌍을 매개변수로 하여 dh_pairwise_test() 호출 – dh_pairwise_test() 내부에서 테스트를 위해 추가로 키쌍을 하나 더 생성한 후, 키 합의 과정을 통해 생성된 키쌍의 유효성 검증 수행 (테스트를 위한 추가 키쌍은 계산 효율성을 위해 작은 범위의 지수를 개인키로 하여 공개키를 계산함. 검증 완료 후 제로화 수행)	– kisa_public_selftest.c (dh_pairwise_test())

[KISACrypto V1.0 조건부 키쌍 일치 시험 상세 명세]

다음은 조건부 핵심기능 시험(조건부 엔트로피 건전성 시험)에 대한 상세 내용이다.

조건부 엔트로피 건전성 시험 시점	수행 방법	참조 표준, 관련 소스파일/함수
난수발생기 Instantiate 과정	– KISA_Crypto_drbgInit()에서 엔트로피 잡음원을 수집하는 get_entropy() 함수 내에서 수집되는 잡음원에 대해 건전성 시험 수행 – 참조 표준에 명시되어 있는 반복횟수 테스트(p.11)와 적응성 비율 테스트(p.12)를 구현	– TTAK.KO-120306-R1 – kisa_drbg_selftest.c (entropy_health_test())
난수발생기 Reseed 과정	– KISA_Crypto_drbgInit()에서 엔트로피 잡음원을 수집하는 get_entropy() 함수 내에서 수집되는 잡음원에 대해 건전성 시험 수행 – 참조 표준에 명시되어 있는 반봇횟수 테스트 (p.11)와 적응성 비율 테스트(p.12)를 구현	

[KISACrypto V1.0 조건부 엔트로피 건전성 시험 상세 명세]

해설

암호모듈이 수행하는 조건부 자가시험에 명세한다. 공개키 기반 암호알고리즘의 키쌍 생성 기능이 구현되어 있는 경우 조건부 키쌍 일치 시험이 수반되어야 하고, 난수발생기에서 자체적으로 엔트로피 잡음원을 수집하여 난수를 생성하는 경우에는 조건부 핵심 기능 시험(조건부 엔트로피 건전성 시험)이 수반되어야 한다. VE10.24.01와 동일한 수준에서 조건부 자가시험의 방법, 관련 소스코드/함수에 대한 정보를 상세하게 제시하는 것이 바람직하다. 조건부 핵심 기능 시험(조건부 엔트로피 건전성 시험)의 경우 컷오프값(C)이라고 하는 기준값에 의거하여 동작을 한다. 컷오프값 C는 실제 운영환경에서 수집된 잡음원 각각에 대해 통계적인 'TTAK.KO-120306-R1(소프트웨어 환경에서의 잡음원 엔트로피 검증 알고리즘)' 표준에 명시되어 있는 8가지 통계적 테스트를 통하여 최소 엔트로피를 계산한 후, 유의수준값과 추정 엔트로피를 바탕으로 계산된다(예: 반복횟수 테스트의 $C = \left\lceil 1 + \frac{-\log_2 \alpha}{\hat{H}} \right\rceil$). 예를 들어 추정된 엔트로피값 $\hat{H}$가 8이고, 유의수준 α가 2^{-40}일 경우, 반복횟수 테스트에서 컷오프값은 6이 된다. 즉 엔트로피값이 높을수록 컷오프값은 작아진다.

항목	AS10.27
보안요구사항 개요	조건부 암호알고리즘 시험
VE10.27.01 VE10.27.02 VE10.27.03	조건부 암호알고리즘 시험에 대한 수행 방법 명세

작성 예시

KISACrypto V1.0 암호모듈은 조건부 암호알고리즘 시험을 동작 전 핵심 기능 시험으로서 수행하며, 상세한 수행 방법에 대해서는 'KISACrypto V1.0 암호알고리즘 시험 상세 명세'에 명세하였다.

해설

암호알고리즘이 최초로 사용되기 이전에 조건부 암호알고리즘 시험이 선행되어야 한다. 이 시험항목은 두 가지 형태로 충족 가능하다. 첫 번째는 암호알고리즘이 최초 실행될 때, 조건부 암호알고리즘 시험이 수행되도록 하는 것이며 이후의 실행에서는 조건부 암호알고리즘 시험을 수행하지 않는다. 이와 같은 형태의 조건부 자가시험을 위해서는 암호알고리즘별 암호알고리즘 시험 여부를 저장하는 상태 정보가 필요하다. 두 번째는 동작 전 핵심 기능 시험에서 암호모듈에 탑재된 모든 검증대상 암호알고리즘에 대해 일괄적으로 KAT 시험을 수행하는 것이다. 두 번째 방법은 암호모듈의 초기화가 느려진다는 단점이 있으나 조건부 암호알고리즘 시험의 구조가 간편하다는 장점이 있다.

항목	AS10.28
보안요구사항 개요	KAT 형태의 조건부 암호알고리즘 시험
VE10.28.01 VE10.28.02	KAT 형태의 조건부 암호알고리즘 시험 수행 방법 명세
작성 예시	
KISACrypto V1.0 암호모듈은 조건부 암호알고리즘 시험을 동작 전 핵심 기능 시험으로서 수행한다. 알고리즘별로 KAT 형태의 암호알고리즘 시험을 수행하고 있으며, 'KISACrypto V1.0 암호알고리즘 시험 상세 명세'와 'SEED 알고리즘 KAT 시험 코드 예시'에 상세한 KAT 시험 방법에 대해 명세하였다.	
해설	
AS10.27 시험항목과 연관된 것으로서, 조건부 암호알고리즘 시험은 KAT 형태로 수행되어야 함을 의미한다. 알고리즘별로 각기 다른 KAT 시험 형태가 가능하기 때문에 적용한 KAT 시험 방법에 대한 상세한 명세가 필요하다.	

항목	AS10.29
보안요구사항 개요	검증대상 알고리즘의 자가시험 대상의 파라미터 크기
VE10.29.01	키 길이, 모듈 크기, 타원 곡선 등은 가장 작은 파라미터 자가시험 수행, 운영모드 중 최소 1개 이상 자가시험 수행 여부 명세
작성 예시	
KISACrypto V1.0 암호모듈은 조건부 암호알고리즘 시험은 동작 전 핵심 기능 시험으로 수행한다. 동작 전 자가시험에서 키 길이, 모듈 크기, 타원 곡선 등 가장 작은 파라미터 자가시험 수행, 운영모드 전체 자가시험을 수행하였다.	
해설	
이 시험항목의 VE10.29.01은 AS10.15, AS10.25와 함께 명세할 수 있다.	

항목	AS10.33
보안요구사항 개요	암호알고리즘 비교 시험
VE10.33.01 VE10.33.02	구현된 암호알고리즘의 비교 자가시험 명세
작성 예시	
KISACrypto V1.0 암호모듈은 조건부 암호알고리즘 시험을 동작 전 핵심 기능 시험으로서 수행하며, 상세한 수행 방법에 대해서는 'KISACrypto V1.0 암호알고리즘 시험 상세 명세'에 명세하였다. 그 중 비교 시험이 사용된다면 이 시험 방법의 절차와 결과를 명세해야 한다.	
해설	
비교 시험은 암호알고리즘을 두 번 이상 독립적으로 구현한 후 2개 이상 구현물의 출력값을 비교하여 출력값이 일치하지 않으면 비교 자가시험은 실패로 판정한다. 이 시험항목의 VE10.33.01은 VE10.27.03과 함께 명세할 수 있다.	

항목	AS10.34
보안요구사항 개요	오류 탐지 시험
VE10.34.01	암호알고리즘에 대한 기지 답안 시험이나 비교 시험을 보완하기 위해 오류 탐지 시험 구현 여부 명세
작성 예시	
KISACrypto V1.0 암호모듈은 조건부 암호알고리즘 시험을 동작 전 핵심 기능 시험으로서 수행하며, 상세한 수행 방법에 대해서는 'KISACrypto V1.0 암호알고리즘 시험 상세 명세'에 명세하였다. 그 중 오류 탐지 시험이 사용된다면 이 시험 방법의 절차와 결과를 명세해야 한다. a. 암호알고리즘의 명세/구현에 대한 오류 조건의 설명 b. 각 오류 조건에 대응되는 오류 표시의 명세 c. 오류 탐지 시험에서 각각의 오류 조건이 시험된다는 것을 설명한 근거	
해설	
오류 탐지 시험은 기지 답안 시험이나 비교 시험을 보완하기 위하여 오류 탐지 시험을 구현했는지 여부를 명세해야 한다. 만약 오류가 탐지되면 암호알고리즘에 대한 오류 탐지 자가시험을 실패로 판정한다. 이 시험항목의 VE10.34.01은 VE10.07.01, VE10.09.01, VE10.10.01과 함께 명세할 수 있다.	

항목	AS10.35
보안요구사항 개요	조건부 암호키 쌍 일치 시험
VE10.35.01 VE10.35.02 VE10.35.03	조건부 암호키 쌍 일치 시험에 대한 명세
작성 예시	
KISACrypto V1.0 암호모듈은 공개키 암호알고리즘 RSAES, 전자서명 RSA-PSS, 키 합의 알고리즘 DH를 탑재하고 있으며, 각 알고리즘마다 키쌍 생성 기능도 제공한다. 키쌍을 생성할 때마다 생성된 키쌍의 유효성에 대해 키쌍 일치 시험을 통해 검증하고 있으며, 상세한 시험 방법은 'KISACrypto V1.0 조건부 키쌍 일치 시험 상세 명세'에 명세하였다.	
해설	
AS10.27 시험항목과 연관된 것으로서, 조건부 암호알고리즘 시험은 KAT 형태로 수행되어야 함을 의미한다. 알고리즘별로 각기 다른 KAT 시험 형태가 가능하기 때문에 적용한 KAT 시험 방법에 대한 상세한 명세가 필요하다.	

항목	AS10.53
보안요구사항 개요	주기적 자가시험
VE10.53.01 VE10.53.02	주기적 자가시험(동작 전 자가시험, 조건부 알고리즘 시험)에 대한 명세
작성 예시	
KISACrypto V1.0 암호모듈은 주기적 자가시험 API를 제공한다. 동작 전 핵심기능 시험으로서 조건부 알고리즘 시험 및 소프트웨어 무결성 시험을 수행하는 KISA_Crypto_selftest()이다. 위의 API를 통해 운영자는 암호모듈에 주기적 자가시험을 요청할 수 있다.	
해설	
주기적 자가시험으로서 동작 전 자가시험과 조건부 알고리즘 시험 및 소프트웨어 무결성 시험을 포함해야 한다. 주기적 자가시험은 서비스 요청, 암호모듈 재시작 등을 통해 수행될 수 있다. 암호모듈이 제공하는 주기적 자가시험 방법에 대해 명세해야 한다.	

10 생명주기 보증

가. 목적

생명주기 보증 보안요구사항에서는 암호모듈 개발업체가 암호모듈 설계, 개발, 운영, 암호모듈의 수명 종료까지 최선의 방법으로 암호모듈이 적절히 설계, 개발, 테스트, 설정, 배포, 설치, 폐기되었는지를 보증하는 것과 적절한 운영자 가이드 문서가 제공되는지를 확인하는 것이다. 이 생명주기 보증의 주요 시험항목은 다음과 같다.

- 암호모듈 형상관리 자동화된 형상관리 시스템이 필수적으로 사용되어야 하며, 형상관리에 대한 사항은 형상관리문서를 통해 대체될 수 있음.
- 유한상태모델 암호모듈의 전체 동작 과정을 암호모듈이 필수적으로 가져야 할 상태와 상태 간 천이를 통해 설명함(유한상태 천이도).
- 벤더시험 암호모듈 내부에 구현된 보안기능이 정상적으로 동작하고 있음을 개발업체가 시험하는 것으로, 시험서를 통해 증빙될 수 있음.
- 배포 및 운영 암호모듈이 개발된 후, 안전하게 배포되고 운영되는 방법 명세
- 수명의 종료 암호모듈의 사용이 완료된 후, 안전하게 폐기하는 방법 명세
- 안내서 암호모듈을 사용하는 운영자를 위한 암호모듈 사용 안내서

이 장에서는 생명주기 보증의 주요 보안요구사항을 만족시키기 위해 개발업체가 작성해야 하는 AS 항목별 VE에 대한 작성 가이드를 제시한다.

나. 기본사항

항목	AS11.01
보안요구사항 개요	생명주기 보증에 관한 문서 작성
VE11.01.01	KS X ISO/IEC 19790 부속서 A.2.11의 생명주기 보증에 관한 문서 작성
작성 예시	
KISACrypto V1.0 암호모듈은 KS X ISO/IEC 19790 부속서 A.2.11의 주요 내용에 대해 추가적인 형상관리 문서를 통해 명세하고 있다.	
해설	
KS X ISO/IEC 19790 부속서 A.2.11에서는 다음과 같은 사항을 담은 문서 작성을 요구한다. 1~6번 항목은 형상관리문서를 통해 명세가 가능하고, 7번 항목은 기본 및 상세설계서에서 유한상태모델을 통해 명세한다. 1. 암호모듈 개발에 사용한 형상관리 시스템 명세 2. 형상관리 시스템을 통해 관리하는 개발문서, 암호모듈 구성요소 및 암호모듈 명세 3. 암호모듈의 안전한 설치, 생성, 시동을 위한 절차 명세 4. 소프트웨어 소스코드와 모듈과의 관련성을 분명히 설명할 수 있는 코멘트 제시 5. 소프트웨어/펌웨어의 소스코드 명세 6. 관리자 안내서 7. 유한상태모델 기술(상태천이도, 상태천이표 활용)	

다. 형상 관리

항목	AS11.03
보안요구사항 개요	형상관리 시스템의 사용
VE11.03.01	개발문서, 모듈 구성요소, 모듈에 관리 적용된 형상관리 시스템 명세
작성 예시	
KISACrypto V1.0 암호모듈은 자동화된 형상관리 시스템을 사용하여 개발문서, 암호모듈 소스코드, 암호모듈, 테스트 프로그램, 무결성 검증키 등에 대해 형상관리를 수행한다. 자세한 사항은 형상관리문서를 통해 확인 가능하다.	
해설	
이 시험항목에 대해서는 형상관리문서에 작성되어 있는 내용을 인용하여 증빙 가능하다. 형상관리 시스템으로는 SVN, GIT 등이 사용될 수 있으며, 형상변경을 위한 적절한 접근제어가 적용되어야 한다. 또한 사용되는 자동화된 형상관리 시스템의 상세 정보도 제시되어야 한다.	

항목	AS11.04
보안요구사항 개요	형상관리 항목
VE11.04.01	형상관리 시스템으로 관리되는 항목, 버전 부여 방법

작성 예시

KISACrypto V1.0 암호모듈은 다음과 같은 항목에 대해 자동화된 형상관리 시스템을 통해 형상을 관리한다.

분류	이름	설명
개발문서	기본 및 상세 설계서	KS X ISO/IEC 19790:2015, KS X ISO/IEC 24759:2015의 보안수준 1 소프트웨어 암호모듈의 요구사항을 만족하도록 암호모듈의 설계를 증빙하는 문서
	형상관리문서	암호모듈에 대한 형상관리 대상, 방법, 정책 및 절차 기술
	보안정책서	암호모듈에 대한 특징, 설정, 사용 방법에 대한 명세
	시험서	암호모듈이 제공하는 보안서비스에 대한 시험 수행 결과에 대한 보고서(시험항목에 대한 시험 목적, 시험 절차 및 시험 결과 기술)
	관리자 안내서	암호모듈 운영에 대한 안내서
소스코드	암호모듈 원천 소스코드	*.c, *.h로 이루어진 암호모듈 소스코드
암호모듈	KISACrypto32.dll, KISACrypto64.dll, libKISACrypto32.so, libKISACrypto64.so	빌드된 암호모듈 라이브러리 파일
무결성 검증키	VerificationKey	바이너리 파일 형태로 관리
무결성 검증값 생성 프로그램	IntegrityGenSoftware.exe	무결성 검증값을 생성하는 소프트웨어
암호모듈 테스트 프로그램	KISACryptoTester.exe	암호모듈의 보안서비스를 테스트할 수 있는 테스트 프로그램

[형상관리되는 항목]

형상관리되는 항목에 대해 유일한 버전을 식별하는 방법은 다음과 같다.

항목명 V(Major Version).(Minor Version)

- 항목명: 형상관리되는 파일에 대한 이름
- Major Version: 암호모듈의 메이저 버전으로서 형상관리위원회를 통해 주요한 기능 추가 및 변경이 발생할 경우 1씩 증가시키며, 1자리 숫자로 구성되며 최초 '1'부터 시작됨.
- Minor Version: 암호모듈의 마이너 버전으로서 형상관리위원회를 통해 일부 기능의 변경이 발생할 경우 1씩 증가시키며, 1자리 숫자로 구성되며 최초 '0'부터 시작됨.

해설
이 시험항목에 대해서는 형상관리문서에 작성되어 있는 내용을 인용하여 증빙 가능하다. 형상관리 시스템으로 관리되는 항목에 대한 명세와 형상관리 시스템을 이용하여 버전을 유일하게 식별하는 방법에 대해 명세한다.

항목	AS11.05
보안요구사항 개요	형상관리 시스템 유지
VE11.05.01	형상관리 시스템을 통한 식별, 등록, 개정 등 추적 필요
작성 예시	
KISACrypto V1.0 암호모듈은 형상관리 시스템을 이용하여 형상관리되는 항목에 대한 등록·변경 등에 대한 이벤트가 발생할 경우 커밋 시의 로그 기능을 이용하여 로그를 남긴다. 이를 통해 개정에 대한 이력을 추적할 수 있도록 한다.	
해설	
이 시험항목은 암호모듈 개발업체가 실제로 형상관리 시스템을 통해 개발문서, 암호모듈/소스파일 등의 등록·변경·삭제 시에 로그를 반영하고 있는지는 확인하는 것이다. 따라서 형상관리 시스템의 로그 기능을 이용하여 증빙 가능하다.	

라. 유한상태모델

항목	AS11.08
보안요구사항 개요	유한상태모델
VE11.08.01	암호모듈에 대한 유한상태모델(상태천이도, 상태천이표) 명세
작성 예시	

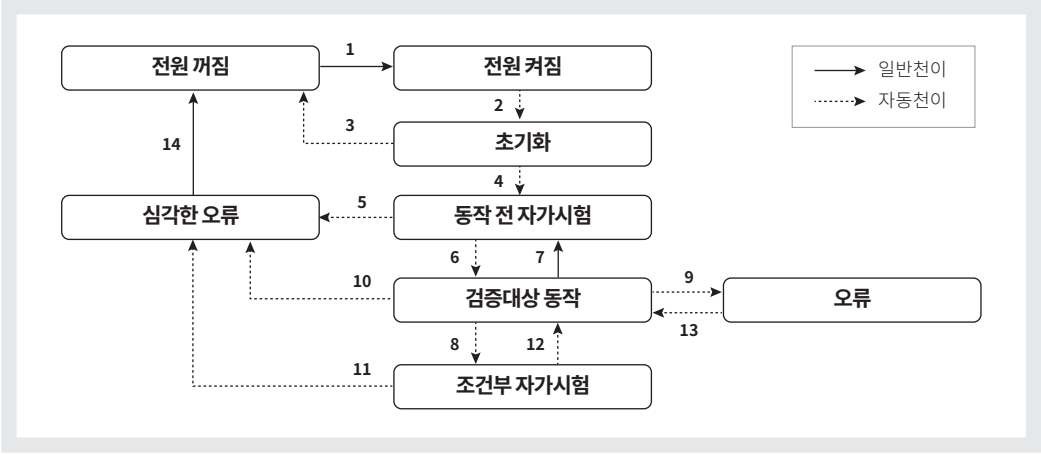
KISACrypto V1.0 암호모듈이 가질 수 있는 상태에 대해서는 다음 표를 통해 명세한다.

번호	상태	입력
1	전원 꺼짐	암호모듈이 메모리에 로딩되지 않는 상태
2	전원 켜짐	암호모듈을 이용하는 응용 프로그램이 실행되어 암호모듈의 실행 파일이 메모리에 로드된 상태
3	초기화	응용 프로그램이 암호모듈을 초기화하는 API를 호출한 상태

번호	상태	입력
4	동작 전 자가시험	암호모듈이 동작 전 자가시험을 수행하는 상태
5	검증대상 동작	동작 전 자가시험을 통과한 암호모듈이 운영자에게 암호 서비스를 수행할 수 있는 상태 또는 암호 서비스를 수행하는 상태
6	조건부 자가시험	암호모듈이 암호 서비스를 수행할 때, 관련된 조건부 자가시험을 수행하는 상태
7	오류	암호모듈이 암호 서비스를 수행할 때, 운영자의 실수로 인한 단순한 오류가 발생한 상태(오류코드를 출력한 후 자동으로 검증대상 동작으로 천이됨.)
8	심각한 오류	암호모듈이 동작 전 자가시험을 실패하거나, 조건부 자가시험을 실패하여 빠지게 되는 상태(운영자가 개입하여 오류 해결 필요)

[암호모듈 상태 설명]

위에서 정의된 상태를 바탕으로 암호모듈의 동작은 다음의 상태천이도를 통해 설명된다.



[상태천이도]

또한 암호모듈의 동작은 다음의 상태천이표를 통해 입력·출력을 포함하여 좀 더 상세하게 설명될 수 있다.

번호	현재 상태	입력	출력	다음 상태
1	전원 꺼짐	응용 프로그램 시작 또는 응용 프로그램이 암호모듈의 초기화 API 호출	출력 없음	전원 켜짐
2	전원 켜짐	자동천이	출력 없음	초기화
3	초기화	자동천이	수행 결과(실패)	전원 꺼짐
4	초기화	자동천이	출력 없음	동작 전 자가시험
5	동작 전 자가시험	자동천이	수행 결과(실패)	심각한 오류
6	동작 전 자가시험	자동천이	수행 결과(성공)	검증대상 동작
7	검증대상 동작	주기적 자가시험 API 호출	출력 없음	동작 전 자가시험

번호	현재 상태	입력	출력	다음 상태
8	검증대상 동작	자동천이(조건부 자가시험 함수 호출)	출력 없음	조건부 자가시험
9	검증대상 동작	자동천이	수행 결과(실패)	오류
10	검증대상 동작	자동천이	수행 결과(실패)	심각한 오류
11	조건부 자가시험	자동천이	수행 결과(실패)	심각한 오류
12	조건부 자가시험	자동천이	수행 결과(성공)	검증대상 동작
13	오류	자동천이	출력 없음	검증대상 동작
14	심각한 오류	응용 프로그램이 암호모듈의 종료 API 호출 또는 응용 프로그램 종료 수행	출력 없음	전원 꺼짐

[상태천이표]

해설

암호모듈은 유한상태모델에 의거하여 동작해야 한다. 유한상태모델은 암호모듈이 필수적으로 갖추어야 할 상태 간에 어떤 이벤트를 통해 천이되는지를 증빙할 수 있는 방법이다. 소프트웨어 라이브러리 형태의 암호모듈의 경우, 필수적으로 들어가야 할 상태는 초기화, 동작 전 자가시험, 검증대상 동작모드, 조건부 자가시험, 오류 등이다. 또한 천이는 운영자가 개입하는 일반천이와 자동으로 상태가 바뀌는 자동천이로 구분된다. 유한상태모델을 통하여 암호모듈이 전체적으로 어떻게 동작하는지를 증명할 수 있으며, 실제 암호모듈은 유한상태모델에 의거하여 동작해야 한다.

항목	AS11.11
보안요구사항 개요	오류 상태에 대한 복구
VE11.11.01	심각한 오류 상태가 아닐 경우, 복구가 가능하다는 명세

작성 예시

KISACrypto V1.0 암호모듈은 유지보수 기능을 제공하지 않으며, 운영자의 실수로 인한 단순한 오류(잘못된 키 사이즈 입력, 잘못된 알고리즘 ID 사용 등)는 관련 오류코드를 출력한 후 자동으로 정상동작 모드 상태로 자동천이되도록 설계하였다.

해설

암호모듈은 유지보수 상태, 심각한 오류 상태가 아닐 경우 그로부터 복구가 가능해야 한다. 소프트웨어 라이브러리 형태의 암호모듈의 경우 운영자의 실수로 발생하는 단순한 오류의 경우에는 관련 오류코드를 출력한 후 자동으로 정상동작 모드(검증대상 동작모드) 상태로 자동천이되도록 암호모듈을 설계할 수 있다.

마. 개발

항목	AS11.15
보안요구사항 개요	개발환경 명세
VE11.15.01	개발환경(컴파일러, 링커, 런타임 라이브러리 등)에 대한 상세 정보 명세
작성 예시	
KISACrypto V1.0 암호모듈의 운영환경은 윈도우 10 32/64비트와 우분투 리눅스 20.14 32/64비트이다. 윈도우 환경에서는 Visual Studio 2019(16.7.5) 버전을 이용하였다. 윈도우 환경에서는 최종적으로 release 모드를 통하여 암호모듈을 최종 빌드하여 'KISACrypto32.dll', 'KISACrypto64.dll' 파일을 생성하였다. 리눅스 환경에서는 Makefile을 정의하였으며, make 유틸리티를 이용하여 암호모듈을 빌드하였다. 컴파일러는 GCC 9.3.0 버전을 이용하였으며, fPIC 옵션을 이용하여 공유 라이브러리 형태로 빌드 하였다. 또한 최적화 옵션은 -O3를 적용하였다. 최종적으로 빌드된 암호모듈은 'libKISACrypto32.so', 'libKISACrypto64.so'이다.	
해설	
암호모듈의 운영환경별 상세한 개발옵션에 대한 정보를 제공해야 한다. 리눅스/유닉스 계열의 운영환경인 경우 Makefile을 인용하는 것도 가능하다.	

항목	AS11.16
보안요구사항 개요	소스코드 주석
VE11.16.01	소스코드 주석 작성을 통해 암호모듈의 설계와 일치함을 증명
작성 예시	
KISACrypto V1.0 암호모듈을 구성하는 *.c, *.h 파일에 대해 모두 주석을 작성하여 해당 파일의 목적, 용도, 그리고 API별로 목적, 용도, 관련 매개변수에 대한 식별이 가능하도록 하였다.	
해설	
암호모듈을 구성하는 소스코드에는 실제 설계문서와의 일치함을 증빙할 수 있도록 주석을 작성해야 한다.	

항목	AS11.19
보안요구사항 개요	무결성 결과 코드 탑재
VE11.16.01	개발 단계에서 무결성 및 인증 기법 메커니즘의 결과 코드 명세
작성 예시	
무결성 검증 코드 생성을 위해 별도 프로그램인 'IntegrityGenSoftware'을 사용하여 암호모듈 파일의 무결성 검증값을 생성하고, 해당 값을 암호모듈 하단에 결합하였다. 무결성 검증키는 검증대상 난수발생기로 생성하여 안전하게 생성되고 관리된다. 또한 무결성 검증값 생성 프로그램은 암호모듈과 같이 형상관리의 적용을 받으며, 해당 프로그램에 구현된 소스코드는 암호모듈의 소스파일에 구현된 HMAC(SHA-256)와 정확히 동일하다.	
해설	
개발 단계에서 무결성 및 인증 기법 메커니즘의 결과 코드를 계산하여 이 결과 코드를 소프트웨어 모듈에 결합해야 한다.	

항목	AS11.21
보안요구사항 개요	개발 도구(컴파일러)
VE11.16.01	소프트웨어 암호모듈 구성 요소에 대하여 암호모듈을 개발하기 위해 사용한 개발 도구 명세
작성 예시	
암호모듈 운영환경별 컴파일러, 컴파일 옵션, 빌드 도구, 빌드 방법 등 개발환경 및 개발 도구 설정 방법을 명세	
해설	
암호모듈의 개발환경 및 개발 도구 설정 방법(컴파일 옵션 등)을 각각 상세히 명세해야 한다.	

바. 벤더 시험

항목	AS11.29
보안요구사항 개요	암호모듈 기능시험
VE11.29.01	암호모듈에 대해 수행된 기능시험 명세
작성 예시	
KISACrypto V1.0 암호모듈에 대한 벤더 시험은 추가 개발문서인 '암호모듈 시험서' 를 통해 증빙한다.	
해설	
벤더 시험은 암호모듈이 제공하는 보안서비스에 대해 정상동작을 확인하기 위해 개발업체가 수행하는 시험으로서, 암호모듈 시험서를 통해 증빙 가능하다.	

항목	AS11.30
보안요구사항 개요	자동화된 보안 진단 도구 이용
VE11.15.01	자동화된 보안 진단 도구를 이용하여 소프트웨어/펌웨어의 취약점 점검
작성 예시	
KISACrypto V1.0 암호모듈에 대해 공개 취약점 진단 도구인 Yasca V2.1을 적용하여 버퍼오버플로에 활용될 수 있는 취약한 함수와 코드를 제거하였다. 대표적으로 매개변수에 버퍼의 길이를 명시하지 않는 strlen, strcmp, strcpy 등의 함수를 제거하였으며, 특정 버퍼로 데이터를 복사하는 경우, 버퍼 간 값을 비교하는 경우 등에서 반드시 사용되는 버퍼의 길이를 체크하도록 하였다.	
해설	
암호모듈의 소프트웨어/펌웨어 구성요소에 대해 자동화된 소스코드 취약점 진단 도구를 적용해야 하며, 알려져 있는 다양한 취약점(예: 버퍼오퍼플로 등)에 대해 대응해야 한다. 이 시험항목에서는 암호모듈에 적용된 취약점 진단 도구와 적용 결과에 대해 명세한다. 기존에 발생한 취약점을 어떻게 수정했는지에 대한 내용을 추가하는 것이 바람직하다.	

사. 배포 및 운영, 수명의 종료

항목	AS11.32
보안요구사항 개요	안전한 배포 및 운영
VE11.32.01	안전한 배포 및 운영을 위한 절차 명세

작성 예시

KISACrypto V1.0 암호모듈은 다음 절차를 통해 안전하게 배포, 운영된다.

구분 (주체)	절차
설치 전 검증필 암호모듈 확인 (암호모듈 관리자)	1. 보안 USB를 통해 암호모듈을 배포. 인수자는 암호모듈 파일에 대한 파일명과 시험기관 홈페이지에 등재되어 있는 해시값을 이용하여 검증필 여부 확인 – 암호모듈 파일 : KISACrypto V1.0 – 버전 : 1.0
설치 (인수자)	1. 인수자는 암호모듈 파일(KISACrypto32.dll, KISACrypto64.dll, libKISAcrypto32.so, libKISAcrypto64.so)을 설치할 경로로 이동 – 별도의 설치 프로그램 없음. – 일반 동적/공유 라이브러리를 링크하여 사용하는 것과 동일하게 사용 가능
초기화 및 시동 (인수자)	1. 테스트 프로그램을 이용하여 암호모듈에 대한 테스트 수행 가능. 암호모듈 로드 후, 동작 전 자가시험 수행 성공 여부 확인 파일에 대한 파일명과 버전 확인 – 시험 실패 시 인수자는 개발팀장과 보안관리자에게 문의 2. 암호모듈이 제공하는 서비스 API 호출 후 결과 확인 – 암호모듈명과 버전을 확인하는 API를 통해 정보 확인
소거(개발자, 인수자)	1. 사용이 완전히 완료되어 삭제하는 경우, 특정 경로에 저장된 라이브러리 파일 삭제

해설

암호모듈은 사전에 할당된 운영자에게 안전한 배포 과정을 통해 전달되어야 하고, 안전한 운영을 위한 초기화와 설치 절차도 안내되어야 한다. 안전한 설치를 위해 스크립트 형태의 설치 명령어 집합을 제공할 수도 있다.

항목	AS11.36
보안요구사항 개요	안전한 소거
VE11.36.01	안전한 소거 절차 명세

작성 예시

KISACrypto V1.0 암호모듈은 동적 라이브러리 형태로서 일반 동적 라이브러리와 동일한 형태로 응용 프로그램에 링크되어 암호 서비스를 제공한다. 응용 프로그램과 독립적인 형태로 존재하기 때문에 특정 패스에 저장된 암호모듈 라이브러리 파일을 삭제함으로써 소거가 가능하다.

해설

암호모듈의 사용이 완전히 종료된 후, 안전한 소거 절차를 통해 암호모듈을 제거해야 한다. 안전한 소거를 위해 스크립트 형태의 설치 명령어 집합을 제공할 수도 있다.

아. 안내서

항목	AS11.38
보안요구사항 개요	관리자 안내서
VE11.38.01	관리자 안내서 제공
작성 예시	
KISACrypto V1.0 암호모듈을 안전하게 관리하고 운영하기 위한 관리자 안내서를 개발하여 이 암호모듈을 사용하는 도입기관에서 활용할 수 있도록 한다.	
해설	
관리자 안내서는 암호모듈 관리자가 암호모듈의 안전한 관리와 운영에 필요한 정보를 제공하는 문서이다.	

항목	AS11.39
보안요구사항 개요	비관리자 안내서
VE11.39.01	비관리자 안내서 제공
작성 예시	
KISACrypto V1.0 암호모듈에서 제공하는 보안 서비스에 대한 API 사용 방법을 매뉴얼 형태로 제공하여 암호모듈의 일반 사용자가 암호모듈을 안전하게 사용할 수 있도록 한다. 대표적으로 매개변수에 버퍼의 길이를 명시하지 않는 strlen, strcmp, strcpy 등의 함수를 제거하였으며, 특정 버퍼로 데이터를 복사하는 경우, 버퍼 간 값을 비교하는 경우 등에서 반드시 사용되는 버퍼의 길이를 체크하도록 하였다.	
해설	
비관리자 안내서는 암호모듈의 일반 사용자가 암호모듈을 안전하게 사용할 수 있도록 안내하는 문서이다. 일반 사용자를 위한 문서로서 API 사용 매뉴얼 형태를 제공할 수 있다.	

11 기타 공격에 대한 대응

가. 목적

KS X ISO/IEC 19790:2015, KS X ISO/IEC 24759:2015 표준상에 명시되어 있지 않은 공격을 완화시킬 수 있는 방법을 암호모듈이 제공할 경우, 공격과 대응 방법에 대해 제시한다. KCMVP에서는 이 보안요구사항 영역에서 버퍼오버플로를 포함하는 소프트웨어 취약점에 대한 대응 방법과 공개키 암호알고리즘을 구현한 경우 단순 타이밍 공격에 대해 대응할 것을 요구한다.

항목	AS12.02
보안요구사항 개요	기타 공격에 대한 대응
VE12.02.01	기타 공격과 대응 방법에 대한 명세

작성 예시

이 암호모듈은 버퍼오버플로를 포함한 소스코드 취약점을 자동화된 점검도구(Yasca V2.1)을 통해 확인하여 보완하였다. 대표적으로 매개변수에 버퍼의 길이를 명시하지 않는 strlen, strcmp, strcpy 등의 함수를 제거하였으며, 특정 버퍼로 데이터를 복사하는 경우, 버퍼 간 값을 비교하는 경우 등에서 반드시 사용되는 버퍼의 길이를 체크하도록 하였다. 또한 암호모듈 시험기관에서 사용하는 소스코드 취약점 정적 분석 도구를 활용하여 잠재적인 취약점으로 활용될 수 있는 코드에 대해 모두 보완하였다.

암호모듈이 탑재하고 있는 RSAES, RSA-PSS, DH에 대해서는 큰수에 대한 지수승 연산이 핵심연산이다. 지수승 연산은 효율성을 위해 Square-and-Multiply 방법을 통해 계산된다. 그러나 일반적인 Square-and-Multiply 방법의 경우에는 지수 비트값에 따라 Multiply 연산의 수행 여부가 결정된다. 즉 지수 비트값에 따라 특정 연산의 수행 여부가 결정되기 때문에 수행되는 시간의 변동을 파악하여 사용된 지수 값을 공격자가 파악할 수 있게 된다. 이를 방지하기 위해 지수가 어떤 값을 가지더라도 항상 동일한 연산을 수행하는 지수승 방법을 사용할 수 있다. 이를 constant-time 지수승 방법이라고 하며, 대표적인 방법으로 더미연산 방법을 이용하는 Square-and-Multiply-Always 방법이 있다. 또한 Laddering 기반의 constant-time 지수승 방법이 존재한다. 이 암호모듈은 Laddering 기반의 지수승 방법을 이용하여 단순 시간차 공격에 대응한다.

해설

소프트웨어 암호모듈의 경우 알려져 있는 소프트웨어 취약점에 대한 대응 사항, 그리고 공개키 암호알고리즘을 구현한 경우 단순 타이밍 공격에 대한 대응 방법을 제시한다.

암호모듈 제출물
작성 안내서






II

형상관리문서
작성 안내서

Ⅱ 형상관리문서 작성 안내서

1 개요

KS X ISO/IEC 19790:2015, KS X ISO/IEC 24759:2015 표준문서의 생명주기 보증: 형상관리에서 암호모듈 개발 시에 형상관리 시스템을 이용하여 암호모듈에 대한 설계서, 소스코드, 암호모듈에 대한 버전을 식별하여 관리할 것을 요구하고 있다. 이 문서는 암호모듈 개발업체가 형상관리 관련 시험항목에 형상관리문서를 작성하는 데 가이드하는 것을 목표로 한다.

2 형상관리 시스템

자동화된 형상관리 시스템을 이용하여 암호모듈 설계서와 소스코드, 암호모듈에 대한 형상을 관리할 것을 요구한다. 자동화된 형상관리 시스템으로는 SVN, GIT 등 다양한 형상 및 버전 관리 소프트웨어가 존재한다. 암호모듈 개발업체가 사용하고 있는 형상관리 시스템의 서버, 클라이언트 환경에 대해 상세히 명세한다.

가. 형상관리 시스템 구성 및 설정

다음에서 설치하여 운영하고 있는 형상관리 시스템의 서버, 클라이언트 제품, 버전, 그리고 설정 방법에 대해 명세한다. 형상관리 시스템은 반드시 적절한 접근통제를 통해 정해진 사용자만이 접근하여 형상관리를 수행할 수 있도록 해야 한다.

나. 사용자 추가 및 사용 방법

형상관리 시스템을 이용할 수 있는 사용자에 대한 설정과 형상관리 시스템을 이용할 수 있는 방법에 대한 내용을 작성한다.

3 형상항목 및 형상관리 절차

형상관리의 대상이 되는 것은 설계서, 암호모듈 소스코드, 암호모듈 등이다. 따라서 형상관리 시스템을 통해 버전 관리되는 자료의 종류를 명세해야 한다. 또한 형상변경을 수행하기 위해서는 적절한 절차가 필요하며, 이는 형상관리위원회와 같은 명시적인 절차를 통해 수행되어야 한다.

가. 형상항목

다음에서 형상관리되는 설계서의 종류, 암호모듈 소스코드, 암호모듈, 테스트 프로그램, 무결성 검증키 등의 항목을 제시한다. 다음과 같이 제시 가능하다.

분류	이름	설명
개발문서	기본 및 상세 설계서	KS X ISO/IEC 19790:2015, KS X ISO/IEC 24759:2015의 보안수준 1 소프트웨어 암호모듈의 요구사항을 만족하도록 암호모듈의 설계를 증빙하는 문서
	형상관리문서	암호모듈에 대한 형상관리 대상, 방법, 정책 및 절차 기술
	보안정책서	암호모듈에 대한 특징, 설정, 사용방법에 대한 명세
	시험서	암호모듈이 제공하는 보안서비스에 대한 시험 수행결과에 대한 보고서(시험항목에 대한 시험 목적, 시험 절차 및 시험 결과 기술)
	관리자 안내서	암호모듈 운영에 대한 안내서
소스코드	암호모듈 원천 소스코드	*.c, *.h로 이루어진 암호모듈 소스코드
암호모듈	KISACrypto.dll, libKISACrypto.so	빌드된 암호모듈 라이브러리 파일

분류	이름	설명
무결성 검증키	VerificationKey	바이너리 파일 형태로 관리
무결성 검증값 생성 프로그램	IntegrityGenSoftware.exe	무결성 검증값을 생성하는 소프트웨어
암호모듈 테스트 프로그램	KISACryptoTester.exe	암호모듈의 보안서비스를 테스트할 수 있는 테스트 프로그램

[형상관리되는 항목]

나. 버전 부여 방법

형상관리 시스템을 이용하여 형상관리되는 자료에 유일한 버전을 부여하는 방법을 명세한다. 다음과 같은 형태로 명세 가능하다.

항목명 V(Major Version).(Minor Version)

- 항목명: 형상관리되는 파일에 대한 이름
- Major Version: 암호모듈의 메이저 버전으로서 형상관리위원회를 통해 주요한 기능 추가 및 변경이 발생할 경우 1씩 증가시키며, 1자리 숫자로 구성되며 최초 '1'부터 시작됨.
- Minor Version: 암호모듈의 마이너 버전으로서 형상관리위원회를 통해 일부 기능의 변경이 발생할 경우 1씩 증가시키며, 1자리 숫자로 구성되며 최초 '0'부터 시작됨.

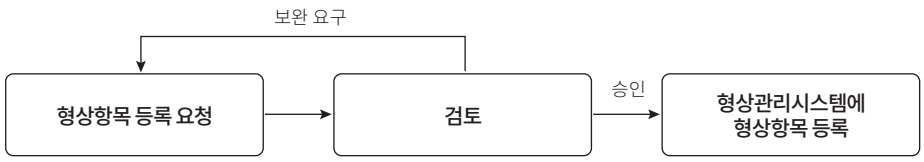
다. 형상관리 절차

형상항목 등록·변경·삭제의 경우 적절한 절차를 통하여 형상관리 시스템에 접근하여 형상관리를 수행해야 한다. 각각의 과정에 대한 명세가 필요하다. 다음은 형상관리를 수행하는 구성 조직에 대한 명세 예시이다.

조직	담당자	역할
형상관리위원회	개발연구소장, 개발팀장	담당자 임명을 총괄하는 조직으로서 형상관리위원장과 위원으로 구성
		형상항목의 등록과 삭제에 대한 승인
형상관리자	형상관리위원회에서 임명	형상관리 시스템 구축 및 관리
		형상관리 시스템의 버전 업데이트, 계정 등록·수정·삭제, 백업·복구
		형상항목의 등록·삭제·백업·복구

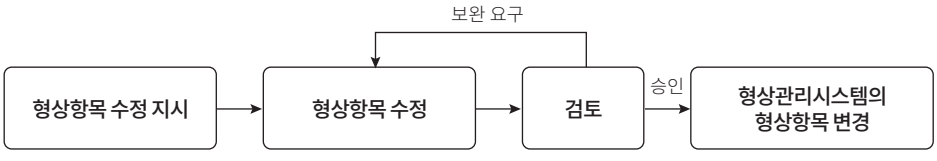
조직	담당자	역할
개발팀장	형상관리위원회에서 임명	제품 개발 총괄
		개발 시스템 구축 및 관리
		형상항목의 변경에 대한 승인
개발팀장	형상관리위원회에서 임명	형상항목의 변경
		형상항목의 반출입
개발자	형상관리위원회에서 임명	제품 개발
		개발 관련 문서 및 사용자 관련 문서 작성
		제품 시험
배포 담당자	형상관리위원회에서 임명	생성된 형상의 배포

1) 형상항목 등록 절차



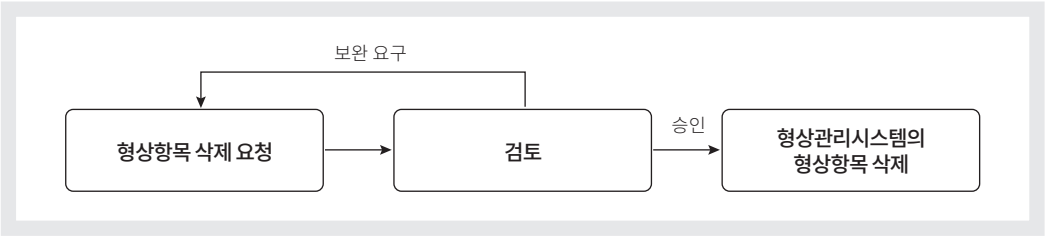
- 형상항목 등록 요청 : 새로운 프로젝트가 생성된 경우, 개발팀장은 개발 프로젝트에 대한 설명과 형상관리 시스템 저장소에 저장될 이름을 기술하여 형상관리위원회에 형상항목 등록을 요청한다. 요청 후에 형상관리위원회로부터 보완 요구가 있다면, 이를 보완하여 다시 요청한다.
- 검토 : 형상관리위원회는 요청된 형상항목 등록에 대해 검토하고, 보완이 필요하면 개발팀장에게 보완 요구를 하고, 문제가 없으면 형상항목 등록을 승인하고, 형상관리자에게 형상관리 시스템에 형상항목을 등록하라고 지시한다.
- 형상항목 등록 : 형상관리자는 형상관리 시스템의 저장소를 생성한다.

2) 형상항목 변경 절차



- 형상항목 수정 지시 : 소스코드, 문서 등 형상항목의 수정이 필요한 경우, 개발팀장은 개발자에게 형상항목 수정을 지시한다.
- 형상항목 수정 : 형상항목 수정을 지시받은 개발자는 형상항목을 수정하여 시험 결과가 성공일 경우, 개발팀장에게 검토를 요청한다. 개발팀장은 소스코드 리뷰와 테스트를 통해 변경이 적절한지 여부를 확인한다. 성공일 경우 형상관리위원회에 형상항목 변경을 요청한다.
- 검토 : 형상관리위원회의 추가 검토가 수행되며, 추가 변경이 필요한 경우 보완을 요구한다. 문제가 없으면 형상관리 변경을 승인한다.
- 형상항목 변경 : 개발팀장은 형상관리 시스템의 형상항목을 변경한다.

3) 형상항목 삭제 절차



- 형상항목 삭제 요청 : 형상항목의 유지가 더 이상 필요하지 않아 삭제하는 경우, 형상항목의 삭제 이유를 기술하여 형상관리위원회에 형상항목 삭제를 요청한다. 요청 후에 형상관리위원회로부터 보완 요구가 있다면, 이를 보완하여 다시 요청한다.
- 검토 : 형상관리위원회는 요청된 형상항목 삭제에 대해 검토하고, 보완이 필요하면 개발팀장에게 보완요구를 하고, 문제가 없으면 형상항목 삭제를 승인하고, 형상관리자에게 형상관리 시스템에서 형상항목을 삭제하라고 지시한다.
- 형상항목 삭제 : 형상관리자는 형상관리 시스템의 저장소에서 형상항목을 삭제한다.

4 배포, 설치, 삭제

암호모듈의 안전한 배포, 설치, 삭제 절차를 명세한다. 암호모듈은 정해진 사용자에게 안전하게 배포되어야 하며, 정상적으로 운영될 수 있도록 설치되어야 한다. 또한 사용이 완료된 후에는 적절한 절차를 통해 삭제되어야 한다. 다음은 배포, 설치, 삭제의 과정에 대한 예시이다.

구분(주체)	절차
설치 전 검증필 암호모듈 확인 (암호모듈 관리자)	1. 보안 USB를 통해 암호모듈을 배포. 인수자는 암호모듈 파일에 대한 파일명과 시험기관 홈페이지에 등재되어 있는 해시값을 이용하여 검증필 여부 확인 – 암호모듈 파일 : KISACrypto V1.0 – 버전 : 1.0
설치 (인수자)	1. 인수자는 암호모듈 파일(KISACrypto.dll, libKISAcrypto.so)을 설치할 경로로 이동 – 별도의 설치 프로그램 없음. – 일반 동적/공유 라이브러리를 링크하여 사용하는 것과 동일하게 사용 가능
초기화 및 시동 (인수자)	1. 테스트 프로그램을 이용하여 암호모듈에 대한 테스트 수행 가능. 암호모듈 로드 후, 동작 전 자가시험 수행 성공 여부 확인 파일에 대한 파일명과 버전 확인 – 시험 실패 시 인수자는 개발팀장과 보안관리자에게 문의 2. 암호모듈이 제공하는 서비스 API 호출 후 결과 확인 – 암호모듈명과 버전을 확인하는 API를 통해 정보 확인
소거 (개발자, 인수자)	1. 사용이 완전히 완료되어 삭제하는 경우, 특정 경로에 저장된 라이브러리 파일 삭제

암호모듈 제출물
작성 안내서






III

시험서
작성 안내서

Ⅲ 시험서 작성 안내서

1 개요

KS X ISO/IEC 19790:2015, KS X ISO/IEC 24759:2015 표준문서의 생명주기 보증: 벤더 시험에서 암호모듈 내부에 구현된 보안기능이 정상적으로 동작하고 있음을 개발업체가 시험하여 증빙할 것을 요구한다. 이 문서는 암호모듈 개발업체가 벤더 시험에 대응하는 암호모듈 시험서를 작성하는 데 가이드하는 것을 목표로 한다.

2 암호알고리즘 시험 결과

가. 시험환경

암호모듈이 시험되는 운영환경에 대한 명세를 한다. 다음과 같이 명세할 수 있다.

구분	항목	시방
하드웨어 환경	CPU	Intel Pentium 4 1GHz 이상
	RAM	1GB 이상
	HDD	10GB 이상
소프트웨어 환경 1	운영체제	윈도우 10 64비트
소프트웨어 환경 2	운영체제	우분투 리눅스 18.04 64비트

나. 검증대상 암호알고리즘

암호모듈이 탑재하고 있는 모든 검증대상 암호알고리즘에 시험을 수행해야 한다. 암호모듈에 탑재된 검증대상 암호알고리즘을 다음과 같이 명세한다.

구분	알고리즘
블록암호	SEED-128
	• 운영모드 : ECB, CBC, CFB128, OFB, CTR
해시함수	SHA-256
메시지인증코드	HMAC-SHA256
난수발생기	HASH_DRBG(SHA-256)
	• 예측내성 미지원, 추가 입력 미지원
공개키	RSAES
	• 공개키 길이: 2048, Hash: SHA-256
전자서명	RSA-PSS
	• 공개키 길이: 2048, Hash: SHA-256
키 설정	DH
	• (공개키 길이, 개인키 길이) : (2048,224), (2048,256)

[암호모듈에 탑재된 검증대상 알고리즘]

다. 시험에 사용된 테스트벡터

암호알고리즘 시험에 사용된 테스트벡터는 해당 알고리즘 표준에 명시된 테스트벡터를 활용할 수 있다.[1] 검증대상 알고리즘 표준에 대해서는 https://seed.kisa.or.kr/kisa/kcmvp/EgovVerification.do 사이트(KISA 암호이용활성화 홈페이지) 또는 https://www.nis.go.kr/AF/1_7_3_2.do(국가정보원 홈페이지)에서 참고할 수 있다. 다음과 같이 알고리즘별로 사용된 테스트벡터에 대해 명세할 수 있다. 테스트벡터 명세 시 출처를 명시하는 것이 권고된다.

1) 알고리즘별 테스트벡터 출처는 국가정보원 또는 한국인터넷진흥원 홈페이지에 명시된 표준과 CAVP 테스트벡터만 허용한다.

1) SEED 알고리즘 테스트벡터(ECB 모드)

테스트를 위한 API명	구분	값
KISA_Crypto_encryptInit() KISA_Crypto_encryptUpdate() KISA_Crypto_encryptFinal() KISA_Crypto_encrypt() KISA_Crypto_decryptInit() KISA_Crypto_decryptUpdate() KISA_Crypto_decryptFinal() KISA_Crypto_decrypt()	출처	KSX3276 (n비트 블록 암호 운영모드-제4부:SEED(2019))
	비밀키 (128비트)	F51F7233331B99AD36006711E3A99836
	평문 (32바이트)	89BC8E0C22DCAB77D620FF5EC1C423FF F61F6485C4D524DC384E7EC4E2672924
	암호문 (32바이트)	9874A63EB46CE9458AE3950115185D66 660FB64A0AB26D3531BA1C584E8FA919

2) SEED 알고리즘 테스트벡터(CTR 모드)

테스트를 위한 API명	구분	값
KISA_Crypto_encryptInit() KISA_Crypto_encryptUpdate() KISA_Crypto_encryptFinal() KISA_Crypto_encrypt() KISA_Crypto_decryptInit() KISA_Crypto_decryptUpdate() KISA_Crypto_decryptFinal() KISA_Crypto_decrypt()	출처	KSX3276 (n비트 블록 암호 운영모드-제4부:SEED(2019))
	비밀키 (128비트)	186E2BA5594710EAE9BB85529C1F2265
	카운터 (16바이트)	EBD8435847640432AD9A692D7B32DC5F
	평문 (48바이트)	8287DD995C9CF590FF5DEB3A93283B7B 818CC401CD51C16DA5B8FAB33A033CC3 CE3806690E238571A39AC21004051B7F

3) SHA-256 알고리즘 시험 테스트벡터

테스트를 위한 API명	구분	값
KISA_Crypto_hashInit() KISA_Crypto_hashUpdate() KISA_Crypto_hashFinal() KISA_Crypto_hash()	출처	[KS X ISO/IEC 10118-3:2001] 해시함수 - 제3부 전용 해시함수 (2018)
	메시지 (3바이트)	616263
	해시값 (32바이트)	BA7816BF8F01CFEA414140DE5DAE2223B00 361A396177A9CB410FF61 F20015AD

테스트를 위한 API명	구분	값
KISA_Crypto_hashInit() KISA_Crypto_hashUpdate() KISA_Crypto_hashFinal() KISA_Crypto_hash()	출처	[KS X ISO/IEC 10118-3:2001] 해시함수 - 제3부 전용 해시함수 (2018)
	메시지 (56바이트)	6162636462636465636465666465666765666768 6666768696768696a68696a6b696a6b6c6a6b6c 6d6b6c6d6e6c6d6e6f6d6e6f706e6f7071
	해시값 (32바이트)	248D6A61D20638B8E5C026930C3E6039A33 CE45964FF2167F6ECEDD419DB06C1

4) HMAC(SHA-256) 알고리즘 시험 테스트벡터

테스트를 위한 API명	구분	값
KISA_Crypto_hmacInit() KISA_Crypto_hmacUpdate() KISA_Crypto_hmacFinal() KISA_Crypto_hmac()	출처	[TTAK.KO-12.0330-Part2] 해시 함수 기반 메시지 인증코드 (HMAC) - 제2부: 해시 함수 SHA-2 (2018)
	비밀키 (64바이트)	00112233445566778899aabbccddeeff00112233445566778899aabbccddeeff0011223344556677 8899aabbccddeeff00112233445566778899aabbc cddeeff
	메시지 (64바이트)	0000111122223333444455556666777788889999 9aaaabbbbccccddddeeeeffff0000111122223333 4444555566667777888899999aaaabbbbccccdddd eeeeffff
	메시지인증코드 (32바이트)	f81f235e7c4abcf72fdf90fbcc03e38f2d352dc 757b5ee452476d1d2903ee528

5) HASH_DRBG(SHA-256) 알고리즘 시험 테스트벡터

테스트를 위한 API명	구분	값
KISA_Crypto_drbgInit() KISA_Crypto_drbgReseed() KISA_Crypto_drbgGenerate() KISA_Crypto_drbgClose() KISA_Crypto_drbgRand()	출처	[TTAK.KO-12.0331-Part2] 해시 함수 기반 결정론적 난수발생기 - 제2부: 해시 함수 SHA-2 (2018)
	예측내성	미사용
	엔트로피 입력 1 (32바이트)	7145910782ACCB48308ABB1C0A4107227B9F1 AA8F26A6CD53F3C032741913A21
	엔트로피 입력 2 (32바이트)	92BAA7658C23A7EE8E80A8EECF3E2B6891A 52DFC49686515007AC763F9244C8C
	논스 (16바이트)	BE1FC13D9266E5280C87112E955995F3
	개별화 문자열 (32바이트)	A1F6BEBDAF3ECD15519841753BF5147DE010 E9D693FD4C68EC053ACD6EB1E405
	추가 입력 1 (32바이트)	6625B06B16AF81E713A03866EC5B7B870CABB 597E25A5DC03FFF7C7DFF176951
	추가 입력 2 (32바이트)	EB57D7B9DE41125F27F686902F4B81F05C1E3 A6D34EB1171C69A185C459BD331
	결과 1 (64바이트)	FCD78232ED6A1AF0F3292FEAC467AA5DCB34FE E49A6276B0D57AA713F6EB998816F3A78EB8BA 38ACB20D2BE1BD272A6347B90B639A80932B
	결과 2 (64바이트)	CD35CC38856B8A0067B07A303B47241CA3E 44E2BADF275728C582F17021212A594E2A9CC 14217883E67565E43BDE2ED1C7A8012E983B1 A5D

라. 알고리즘 시험 절차 및 시험 결과

알고리즘별로 시험 목적, 시험 단계, 예상 실험 결과와 실제 시험 결과를 기술한다. 시험 단계를 암호모듈 시험자가 다시 수행할 수 있도록 명확하고 상세하게 기술해야 한다. 또한 시험 단계에서 테스트를 위한 소스코드를 캡처하여 증빙할 수 있다. 실제 시험 결과의 내용과 화면상에 출력된 결과의 캡처를 통해 증빙 가능하다.

다만 실제 알고리즘 시험을 수행할 때, 알고리즘별 특징을 증빙할 수 있는 형태로 수행하는 것이 바람직하다. 예를 들면 역함수가 존재하는 경우, 역함수를 수행하여 데이터가 복원되는 것을 보인다. 블록암호 암호화와 복호화를 연속적으로 수행하여 역함수의 수행도 정상적으로 수행되는 것을 보일 수 있다. 공개키 암호의 경우 동일한 키와 메시지에 대해서도 항상 다른 암호문이 생성되는 것을 증빙하는 형태로 시험을 수행하는 것이 바람직하며, 키 합의 알고리즘의 경우에는 실제로 공통의 공유키가 생성되는 것을 보이는 것이 좋다.

다음은 블록암호 SEED와 HASH_DRBG에 대한 시험 절차 및 결과이다. 나머지 알고리즘에 대해서도 동일한 형식으로 증빙하는 것이 가능하다.

1) SEED 알고리즘 시험절차 및 결과

시험 목적	SEED 블록암호알고리즘에 대한 ECB, CBC, CFP128, OFB, CTR 운영모드 정상동작 확인
시험 단계	1) 테스트 프로그램을 실행시킨다. 2) 테스트 프로그램에서 SEED 블록암호를 선택한다(테스트 코드를 캡처하여 증빙 가능). 3) 테스트를 위한 테스트벡터를 입력한다(테스트벡터를 파일로 관리하여 입력하도록 할 수도 있다). 4) 암호화 과정과 복호화 과정을 통해 암호화 API와 복호화 API가 정상동작하는 것을 보인다. 5) 데이터 출력과 상태 출력을 관찰한다.
예상 시험 결과	테스트벡터에 명세된 대로 올바른 출력을 생성한다.
실제 시험 결과	테스테벡터에 명세된 출력값과 동일한 데이터 출력을 생성한다(화면 캡처를 통해 증빙 가능).

2) HASH_DRBG (SHA-256) 시험 절차 및 결과

시험 목적	HASH_DRBG 알고리즘에 대한 정상동작 확인
시험 단계	1) 테스트 프로그램을 실행시킨다. 2) 테스트 프로그램에서 HASH_DRBG 난수발생기를 선택한다(테스트 코드를 캡처하여 증빙 가능). 3) 테스트를 위한 테스트벡터를 입력한다(테스트벡터를 파일로 관리하여 입력하도록 할 수도 있다). 4) 난수생성 과정을 최소 2회 이상 수행하여, 매번 다른 난수가 생성되는 것을 보인다. 5) 데이터 출력과 상태 출력을 관찰한다.
예상 시험 결과	테스트벡터에 명세된 대로 올바른 출력을 생성한다.
실제 시험 결과	테스트벡터에 명세된 출력값과 동일한 데이터 출력을 생성한다(화면 캡처를 통해 증빙 가능).

3 암호모듈 관리 API 시험 결과

가. 목적

암호모듈의 관리 API(상태 확인, 동작 전 자가시험, 주기적 자가시험, 제로화 등)의 기능이 정상적으로 동작하는지를 확인하기 위한 시험이다.

나. 관리 API 절차 및 시험 결과

암호모듈 관리자가 수행할 수 있는 API에 대한 정상동작 여부를 확인하기 위한 절차와 결과에 대해 명세한다. 이 관리 API 시험을 수행하기 위해 암호모듈 내부에 디버깅 메시지를 출력하도록 조정할 수 있다. 그러나 암호모듈의 최종 릴리즈 전에는 디버깅 메시지를 모두 삭제해야 한다.

1) 암호모듈 초기화(동작 전 자가시험 수행)

시험 목적	명시된 운영환경에서 암호모듈이 정상적으로 초기화되는지 확인
시험 단계	1) 응용 프로그램이 암호모듈을 링크하여 로드한다. 2) 암호모듈의 초기화 API(KISA_Crypto_initialize())를 호출한다. 3) 초기화 과정의 명확한 확인을 위하여 암호모듈은 디버깅 정보를 출력하는 형태로 동작한다(향후 최종 버전 릴리즈 시에는 디버깅 정보를 삭제된다). 4) 초기화 과정으로 수행되는 동작 전 자가시험의 수행 결과를 확인한다.
예상 시험 결과	동작 전 자가시험이 수행되어 암호모듈이 정상동작 모드로 동작한다.
실제 시험 결과	디버깅 메시지를 통해 출력되는 정보를 통해 성공 여부를 확인한다. 또한 암호모듈의 현재 상태를 출력하여 정상동작 모드 여부 확인이 가능하다.

2) 암호모듈 상태 확인(유한상태모델 확인)

시험 목적	암호모듈의 상태 확인 API가 정상적으로 동작하는지 확인 상태 확인 API를 통해 암호모듈이 유한상태모델에 의거하여 동작하는지 확인
시험 단계	1) 암호모듈의 상태가 변경하는 코드 부분에 상태 출력 API(KISA_Crypto_getState())를 통해 암호모듈의 상태 변화를 확인할 수 있도록 한다. 2) 암호모듈 초기화(동작 전 자가시험 수행) 과정에서 암호모듈의 상태 변화를 출력한다. 3) 정상동작 모드 상태에서 암호알고리즘 서비스를 수행하여 암호모듈의 상태 변화를 출력한다.
예상 시험 결과	유한상태모델에 정의된 대로 특정 이벤트에 따른 암호모듈의 상태 변화가 일치한다.
실제 시험 결과	디버깅 메시지 또는 암호모듈의 상태 출력값을 통해 암호모듈의 동작이 유한상태모델에 의거하여 동작함을 증빙한다.

3) 주기적 자가시험

시험 목적	주기적 자가시험(소프트웨어 무결성 시험, 알고리즘 시험)이 운영자의 요청에 따라 정상적으로 수행되는지 확인
시험 단계	1) 응용 프로그램이 암호모듈을 링크하여 로드한다. 2) 암호모듈의 초기화 API(KISA_Crypto_initialize())를 호출한다. 3) 동작 전 자가시험을 통해 정상동작 모드로 천이된 상태에서 동작 전 소프트웨어 무결성 시험을 재요청한다(KISA_Crypto_integritytest()). 4) 정상동작 모드에서 암호알고리즘 시험 API(KISA_Crypto_selftest())를 수행한다.
예상 시험 결과	운영자의 요청에 따라 주기적 자가시험이 정상적으로 수행된다.
실제 시험 결과	주기적 자가시험이 정상적으로 수행되는 결과를 캡처 또는 디버그 메시지를 통해 출력한다.

4 외부 API에 대한 오류 출력 시험 및 데이터 출력 금지 시험

가. 목적

설계서에 정의된 오류 상황에 대해 암호모듈이 설계서에 명세된 대로의 오류코드를 반환하는지를 확인하고, 추가로 암호모듈이 오류 상태에서 데이터 출력을 금지하는지를 확인한다. 이 시험은 설계서상에 정의된 모든 오류코드에 대해 수행하는 것이 원칙이며, 임의로 오류를 발생시켜 설계서와 일치된 오류코드가 반환되는지를 확인한다.

나. 오류코드 출력 시험 및 데이터 출력 금지 시험

암호 서비스별로 정의된 오류코드에 대해 오류를 유발할 수 있는 시험을 구성하여 수행함으로써 일치하는 오류코드가 반환되는지를 확인한다. 또한 암호모듈이 오류 상태인 경우에는 데이터 출력이 금지되는지를 확인한다. 단순한 오류의 경우 오류코드를 반환한 후 정상동작 모드로 자동천이되기 때문에, 자가시험 실패와 같은 심각한 오류를 유발한 후 암호 서비스가 수행되지 않는 것을 증빙할 수 있다. 다음과 같은 형태로 암호알고리즘별로 오류코드 출력 시험을 수행할 수 있다.

1) 암호모듈 초기화(동작 전 자가시험 수행)

시험 목적	블록암호 키 길이 오류를 발생시켜 설계서와 일치하는 오류코드가 발생하는지 확인
시험 단계	1) 응용 프로그램이 암호모듈을 링크하여 로드한다. 2) 암호모듈의 초기화 API(KISA_Crypto_initialize())를 호출한다. 3) 정상동작 모드에서 블록암호 SEED를 수행한다. 4) 블록암호 SEED에 대한 암호키를 설정 시 키 길이를 의도적으로 잘못 입력한다. 5) 반환되는 오류코드를 확인한다.
예상 시험 결과	반환되는 오류코드는 BLOCK_INVALID_KEY_LENGTH_ERROR로서 16진수값은 설계서에 명세된 대로 0x10001이다.
실제 시험 결과	예상된 오류코드 반환을 확인한다(실행 화면에 대한 캡처를 통해 증빙 가능).

2) 해시함수 오류코드 출력 시험

시험 목적	해시함수 사용 시 잘못된 해시함수 ID를 입력하여 설계서와 일치하는 오류코드가 반환되는지 확인
시험 단계	1) 응용 프로그램이 암호모듈을 링크하여 로드한다. 2) 암호모듈의 초기화 API(KISA_Crypto_initialize())를 호출한다. 3) 정상동작 모드에서 SHA-256을 실행한다. 4) 해시함수 수행 시 API에 전달되는 알고리즘 ID를 SHA-256에 대한 ID가 아닌 정의되지 않은 ID를 전달한다. 5) 반환되는 오류코드를 확인한다.
예상 시험 결과	반환되는 오류코드는 HASH_INVALID_ALGORITHM_ID_ERROR로서 16진수값은 설계서에 명세된 대로 0x10006이다.
실제 시험 결과	예상된 오류코드 반환을 확인한다(실행 화면에 대한 캡처를 통해 증빙 가능).

3) 오류 상태에서 데이터 출력 금지 확인

시험 목적	암호모듈이 오류 상태에서 데이터 출력이 금지되는 것을 확인
시험 단계	1) 암호모듈이 자가시험을 실패하도록 의도적으로 KAT 테스트벡터를 수정한다(또는 조건부 자가시험이 항상 실패하도록 소스코드를 수정한다). 2) 암호모듈의 초기화 API를 이용하여 동작 전 자가시험이 수행되도록 한다(또는 정상동작 모드에서 특정 조건부 자가시험이 수행되도록 한다). 3) 암호모듈이 오류 상태로 천이된 상황에서 암호 서비스를 호출한다.
예상 시험 결과	암호 서비스가 수행되지 않고 즉각 반환되며, 서비스 수행 실패 오류코드만이 반환된다.
실제 시험 결과	예상된 오류코드 반환을 확인하고 데이터 출력이 없음을 확인한다(실행 화면에 대한 캡처를 통해 증빙 가능).