프로젝트 #5 (팀)

소프트웨어학부 암호학

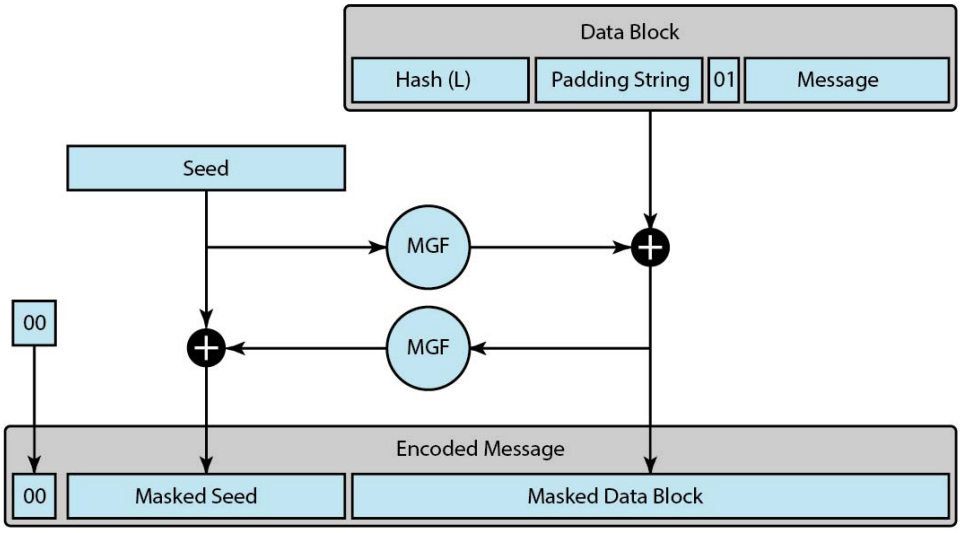
2022년 10월 27일

문제

표준문서 IETF RFC 8017에 명시된 RSA 공개키 암호체계 PKCS #1 버전 2.2를 구현한다. PKCS #1에는 RSAES-OAEP (Encryption/decryption Scheme based on the Optimal Asymmetric Encryption Padding)와 RSASSA-PSS (Signature Scheme with Appendix based on the Probabilistic Signature Scheme)가 들어있다. RSAES-OAEP는 암복호 알고리즘이고, RSASSA-PSS는 확률적 전자서명 알고리즘이다. RSA의 키의 길이는 2048비트로 한다.

RSAES-OAEP

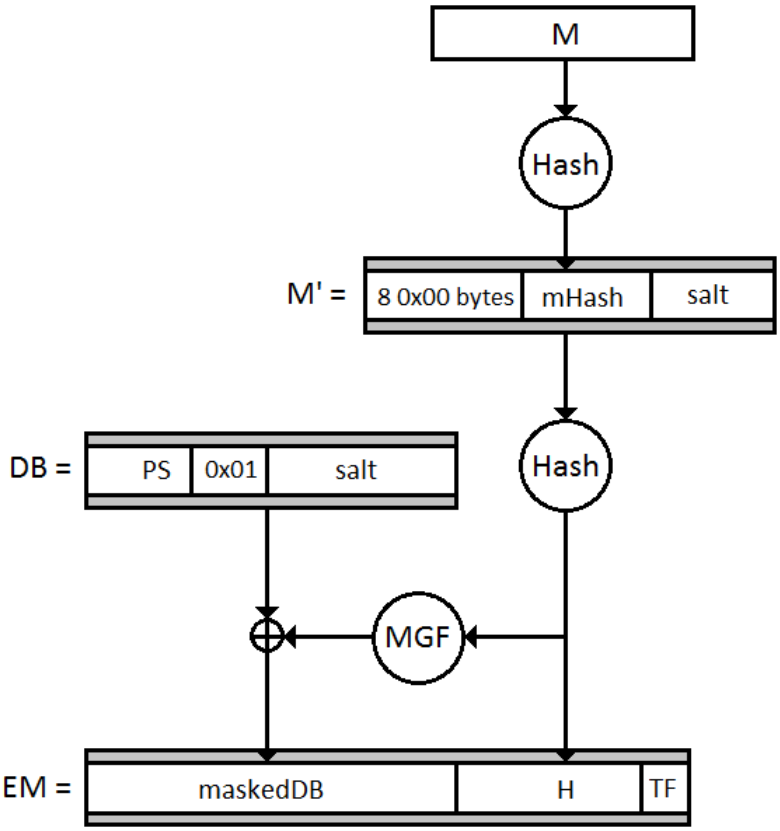
RSAES-OAEP 기법은 암호화할 메시지 M (Message)을 아래 그림과 같은 과정을 거쳐 EM (Encoded Message)으로 변환한 후, 공개키 (e, n)을 사용하여 $\mathrm{EM}^e \bmod n$을 계산한다.



- 암호화할 메시지에 라벨을 붙일 수 있는데, Hash(L)은 그 라벨을 해시한 값이다.
- Padding String은 값이 0x00인 바이트열이다.
- '01'과 '00'은 값이 각각 0x01과 0x00인 단일 바이트이다.
- 해시함수는 길이가 최소 224 비트인 SHA-2 계열의 함수를 사용한다.
- 난수 Seed의 길이는 해시함수의 길이와 같이 한다.
- Encoded Message의 길이는 RSA 키의 길이인 `RSAKEYSIZE` (2048 비트)와 일치해야 한다.

RSASSA-PSS

RSASSA-PSS 기법은 서명할 메시지 M을 아래 그림과 같은 과정을 거쳐 EM으로 변환한 후, 개인키 (d, n)을 사용하여 $\text{EM}^d \bmod n$을 계산한다.



- 해시함수는 길이가 최소 224 비트인 SHA-2 계열의 함수를 사용한다.
- 난수 salt의 길이는 해시함수의 길이와 같이 한다.
- M'의 처음 8 바이트는 0x00으로 채운다.
- PS는 길이에 맞춰 0x00으로 채운다.
- TF는 1 바이트이며 0xBC로 채운다.
- mHash = Hash(M), H = Hash(M')이다.
- EM의 길이는 RSA 키의 길이인 `RSAKEYSIZE` (2048 비트)와 일치해야 한다.
- EM의 가장 왼쪽 비트 (MSB)가 1이면 강제로 0으로 바꾼다.

GMP 라이브러리 설치

GNU GMP 라이브러리는 정수의 크기가 2^{64}보다 큰 수를 계산하기 위해 개발된 패키지이다. 이번 과제에서 기본적으로 제공하는 `rsa_generate_key()`와 `rsa_cipher()` 함수는 GMP 라이브러리를 사용하고 있다. 이들 함수를 활용하려면 각자 환경에 맞는 GMP 라이브러리를 설치해야 한다. GMP는 Linux, macOS, Windows 등 대부분의 환경을 지원한다. GMP를 먼저 설치하고 과제를 진행한다.

- **Linux** 환경에서는 터미널을 열고 다음 두 명령어를 실행한다.

```
% sudo apt update
% sudo apt install libgmp-dev
```

- **macOS** 환경에서는 먼저 Homebrew가 설치되어 있어야 한다 (프로젝트 #3 참조). 터미널을 열고 다음 명령어를 실행한다.

```
% brew install gmp
```

이번 과제에서는 GMP 라이브러리 설치가 필요하지만 당장 이를 이용해서 코딩할 부분은 없다. 그러나 다음 과제에서는 필요하다. GMP 함수를 사용하려면 `#include <gmp.h>`를 하고, `gcc` 링크시 `-lgmp` 옵션을 사용해야 한다. 예를 들어 `% gcc -o sample sample.c -lgmp`가 그 활용이다.

함수 구현

PKCS #1 버전 2.2에 필요한 함수를 아래 열거한 프로토타입을 사용하여 구현한다. 각 함수에 대한 요구사항은 다음과 같다.

- `void rsa_generate_key(void *e, void *d, void *n, int mode)` – 길이가 `RSAKEYSIZE`인 `e`, `d`, `n`을 생성한다. `mode`가 0이면 표준 모드로 `e` = 65537을 선택하고, 0이 아니면 무작위로 선택한다. 이 함수는 기본으로 제공한다.

- `static int rsa_cipher(void *m, const void *k, const void *n)` – $m \leftarrow m^k \bmod n$을 계산한다. 성공하면 0, 그렇지 않으면 오류 코드를 넘겨준다. 내부에서만 사용하는 이 함수는 기본으로 제공한다.

- `void sha224(const unsigned char *m, unsigned int len, unsigned char *digest);`
  `void sha256(const unsigned char *m, unsigned int len, unsigned char *digest);`
  `void sha384(const unsigned char *m, unsigned int len, unsigned char *digest);`
  `void sha512(const unsigned char *m, unsigned int len, unsigned char *digest);`
  `void sha512_224(const unsigned char *m, unsigned int len, unsigned char *digest);`
  `void sha512_256(const unsigned char *m, unsigned int len, unsigned char *digest);`
  – 길이가 `len` 바이트인 메시지 `m`의 SHA-2 해시 값을 `digest`에 저장한다. 이 함수군은 오픈 소스로 기본으로 제공한다.

- `int rsaes_oaep_encrypt(const void *m, size_t len, const void *label, const void *e, const void *n, void *c, int sha2_ndx)` – 길이가 `len` 바이트인 메시지 `m`을 공개키 `(e,n)`으로 암호화한 결과를 `c`에 저장한다. `label`은 데이터를 식별하기 위한 라벨 문자열로 `NULL`을 입력하여 생략할 수 있다. `sha2_ndx`는 사용할 SHA-2 해시함수 색인 값으로 `SHA224`, `SHA256`, `SHA384`, `SHA512`, `SHA512_224`, `SHA512_256` 중에서 선택한다. `c`의 크기는 `RSAKEYSIZE`와 같아야 한다. 성공하면 0, 그렇지 않으면 오류 코드를 넘겨준다.

- `int rsaes_oaep_decrypt(void *m, size_t *len, const void *label, const void *d, const void *n, const void *c, int sha2_ndx)` – 암호문 `c`를 개인키 `(d,n)`을 사용하여 원본 메시지 `m`과 길이 `len`을 회복한다. `label`과 `sha2_ndx`는 암호화할 때 사용한 것과 일치해야 한다. 성공하면 0, 그렇지 않으면 오류 코드를 넘겨준다.

- `int rsassa_pss_sign(const void *m, size_t len, const void *d, const void *n, void *s)` – 길이가 `len` 바이트인 메시지 `m`을 개인키 `(d,n)`으로 서명한 결과를 `s`에 저장한다. `s`의 크기는 `RSAKEYSIZE`와 같아야 한다. 성공하면 0, 그렇지 않으면 오류 코드를 넘겨준다.

- `int rsassa_pss_verify(const void *m, size_t len, const void *e, const void *n, const void *s)` – 길이가 `len` 바이트인 메시지 `m`에 대한 서명이 `s`가 맞는지 공개키 `(e,n)`으로 검증한다. 성공하면 0, 그렇지 않으면 오류 코드를 넘겨준다.

오류 코드

PKCS #1 실행 과정에서 발생하는 오류를 아래에 열거한 코드를 사용하여 식별한다.

- PKCS_MSG_OUT_OF_RANGE – RSA 데이터 값이 모듈러스 n 보다 크거나 같음 [내부 함수]
- PKCS_MSG_TOO_LONG – 입력 메시지가 너무 길어 한도를 초과함 [공통]
- PKCS_LABEL_TOO_LONG – 라벨의 길이가 너무 길어 한도를 초과함 [RSAES-OAEP]
- PKCS_INITIAL_NONZERO – Encoded Message의 첫번째 바이트가 0이 아님 [RSAES-OAEP]
- PKCS_HASH_MISMATCH – 해시 값이 일치하지 않음 [공통]
- PKCS_INVALID_PS – Padding String 뒤에 오는 값이 0x01이 아님 [RSAES-OAEP]
- PKCS_HASH_TOO_LONG – 해시 값의 길이가 너무 커서 수용할 수 없음 [RSASSA-PSS]
- PKCS_INVALID_LAST – EM의 마지막 (LSB) 바이트가 0xBC가 아님 [RSASSA-PSS]
- PKCS_INVALID_INIT – EM의 처음 (MSB) 비트가 0이 아님 [RSASSA-PSS]
- PKCS_INVALID_PD2 – DB의 앞 부분이 0x0000..00||0x01과 일치하지 않음 [RSASSA-PSS]

RSAES-OAEP 테스트 벡터

다음은 RSA 키의 길이가 1024 비트이고 SHA-224를 사용해서 생성한 검증 벡터이다. 아래 벡터를 사용하여 프로그램이 올바르게 돌아가는지 확인하고 난 후, 키의 길이를 2048로 확장하여 구현한다.

```
RSA key pairs with size = 1024

e = 0298c1838a337b4adb8ffce087ee748dd284e41fbcf655698bc803e63d86c9fa
    2aa0863c090a408ea7d612851946acc99f359f3b959e34fa2a6e37d89cc869e3
    2547b5d9a4f2e0d4a6a5be8de889aad703374aa0a454fe2bad307c2f3fab17c7
    ed961e3e30a852a59c3396fe66ab54135b555c787fe5ec35210157ac17d0a385

d = 01e356b87636214fc4c3f30ad7aea29d305db8143882ceb9e8521e43513cab6c
    8d2065d9b38612962ed1c129a90952893a9a07c9afd232975283e9808adc7a4a
    a39e9f66747f614e84355b229c35ad9163f2003f3b211ac46986f52cb268e6bd
    e3a48f938b0220b0424c42df42a6335c94ad39341d4336f6232cb7d40e3aa2cd

n = 8d22c4b32917767565ead7c088133ba6b876a945fcd2b0db6b74ca53885ea919
    ce4f7b41e40a43d743fa033617bd42be67a7b9ee955a64496b7cd2e648d51d6a
    7f8cb204178f6f1e9c6e65a14b4302c0099347d6ef5ad414b77c9389d4118796
    b690784663b37089fd596f3fcf8ac43309bd3801ec1b593a574771dddc5d2fc1

With SHA-224, label = "", message = "sample"

lHash = d14a028c2a3a2bc9476102bb288234c415a2b01f828ea62ac5b3e42f
seed  = 2010ff31602cf33341383ee7b263583623a0ce5671df22b25013c642

MGF1 = 8679eccabe80cff717375e8f87235bbe2caca3114f75ce6745c978f33a06cafe
       f0081e6bcf3f28d06bc8232d12851ead1100423bd71995f06a24df0b238e9b50
       00dc623e672afbcb1f2a4a731c776270c4ec230a7ec460d7a0fdd8a6f1ac9eea
       5a75fa

MGF2 = fe2aff6276137e2aea444234c22f657daba709814e7c9b3048d2ea6f

EM   = 00de3a0053163f8d19ab7c7cd3704c3d4b8807c7d73fa3b98218c12c2d5733ee
```

```
       4694bae43e50565c34afa16f7a390e130ecdfb684d807a9cdc3a06cafef0081e
       6bcf3f28d06bc8232d12851ead1100423bd71995f06a24df0b238e9b5000dc62
       3e672afbcb1f2a4a731c776270c4ec230a7ec460d7a0fdd8a6f0dfff872a199f

c    = 704ff5b744789a3e308294e8db60eac4c37910a3e6cb15b70f33fe72fdf3b58a
       63048640133575564d4d4ee834c7e0bd45559d515af48590b99c41ee54996b19
       e4b0f73a76a282b000da1cd49a4df7146b57156c26301692091b02d05f4c6e3a
       10dd2e508ef418760615dfe96e5474f4eae69fb960bdd22b43c6e5b796bd75b2
```

RSASSA-PSS 테스트 벡터

다음은 RSA 키의 길이가 1024 비트이고 SHA-224를 사용해서 생성한 검증 벡터이다. 아래 벡터를 사용하여 프로그램이 올바르게 돌아가는지 확인하고 난 후, 키의 길이를 2048로 확장하여 구현한다.

```
RSA key pairs with size = 1024

e = 00000000000000000000000000000000000000000000000000000000000000000
    00000000000000000000000000000000000000000000000000000000000000000
    00000000000000000000000000000000000000000000000000000000000000000
    00000000000000000000000000000000000000000000000000000000000010001

d = 06c3eeac459356f743956e0810a0d3f9142943bcce19d111de0e5297aed166b4
    369b3b3e15c9479bf48392b34f71923b3569777e4f05295137518f13e6f48fde
    bd6ebf84a5d8ced584848b8a764ae480506f7d4a5ef7fb7cd5fdf8bf89e5aa45
    e39b99af2beda7fdf43261634ddae38f5beffd0bebf9bd51bfe401d5f6d68671

n = c7b8ab9cdb83fbb008d80e78b2265aa088cdb5f9c11c0a92948c4c56e138730a
    4c815dc9b096fe4c1f4fb5259c0209c6c330ff8349bd9e0687ee49824f63f551
    414795733bdee587b4d3efecda10b2baaf0666458b5d21fd2b975a1babe9305d
    3ac28bed9037a4dab14ce9c414a96ebb412e8d26d6e69610191b3bed82e42dc1

With SHA-224, message = "sample"

mHash = 9003e374bc726550c2c289447fd0533160f875709386dfa377bfd41c
salt  = 6e41978602acca182e8bf511b9acdb04ab324572358c153de6cd3e0a

M'  = 0000000000000000009003e374bc726550c2c289447fd0533160f875709386dfa3
      77bfd41c6e41978602acca182e8bf511b9acdb04ab324572358c153de6cd3e0a

H   = de07484989543520563 9e196634c62305d6433373f688dd840b07f6b

DB  = 0000000000000000000000000000000000000000000000000000000000000000
      0000000000000000000000000000000000000000000000000000000000000000
      000000000000016e41978602acca182e8bf511b9acdb04ab324572358c153de6
      cd3e0a

MGF = c93c72a5e3780e32843ab05bfeef5bc18a78688c859081c546bd86ed2895ce14
      028c98ae4df01c28a6d00b29d7bd9d919b2e28fe95ab09bb9db7c688813627ad
      26d0ee008824235a1ab836231651d93b4e3261c925b5a61e2957c54ae44d00d8
      b84f1f

EM  = 493c72a5e3780e32843ab05bfeef5bc18a78688c859081c546bd86ed2895ce14
      028c98ae4df01c28a6d00b29d7bd9d919b2e28fe95ab09bb9db7c688813627ad
      26d0ee00882422345b2fb021ba9bc115c5c77070896ea2b51b12b77f68583d3e
      757115de07484989543520563 9e196634c62305d6433373f688dd840b07f6bbc
```

```
sig = 80dbbc4987617db7bf5f07f85078bdea501588eca3525dd69023926340e57125
      ec4443d54632acd4c97895394bcf7242fc38a57ce40243783e9a97b3d1eac45f
      8256aa520e44d77120a0585db14c3f4d5195058014e6d924092e57de6237c405
      88b08a6cdc1b73b5fbed022ea12470993dfd3c01480f841863eb23d0383f1b8f
```

골격 파일

구현이 필요한 골격파일 `pkcs.skeleton.c`와 함께 헤더파일 `pkcs.h`, 프로그램을 검증할 수 있는 `test.c`, SHA-2 오픈소스 `sha2.c`, `sha2.h` 그리고 `Makefile`을 제공한다. 이 가운데 `test.c`, `sha2.c`, `sha2.h`를 제외한 나머지 파일은 용도에 맞게 자유롭게 수정할 수 있다.

제출물

과제에서 요구하는 함수가 잘 설계되고 구현되었다는 것을 보여주는 자료를 보고서 형식으로 작성한 후 PDF로 변환하여 `PROJ5(팀원이름).pdf`로 제출한다. 다음과 같은 것이 반드시 포함되어야 한다.

- 작성한 함수에 대한 설명
- 컴파일 과정을 보여주는 화면 캡처
- 실행 결과물의 주요 장면과 그에 대한 설명, 소감, 문제점
- 프로그램 소스파일 (`pkcs.c`, `pkcs.h`) 별도 제출
- 프로그램 실행 결과 (`pkcs.txt`) 별도 제출
- 팀원 평가표 (LMS를 통해 개별적으로 제출한다. 미제출자는 서류제출 미비로 취득한 점수에서 50% 감점한다.)

평가

- **Correctness 50%**: 프로그램이 올바르게 동작하는 지를 보는 것입니다. 여기에는 컴파일 과정은 물론, 과제가 요구하는 기능이 문제없이 잘 작동한다는 것을 보여주어야 합니다. 학생들이 제출한 `pkcs.h`와 `pkcs.c`는 Ubuntu 20.04 LTS 환경에서 컴파일하고 검증합니다. `Makefile`, `test.c`, `sha2.h`, `sha2.c`는 수정할 수 없습니다. 경고를 포함한 모든 오류는 큰 감점을 받습니다. macOS 사용자는 제출하시기 전에 우분투 환경에서 오류가 없는지 확인하시기 바랍니다.
- **Presentation 50%**: 자신의 생각과 작성한 프로그램을 다른 사람이 쉽게 이해할 수 있도록 프로그램 내에 적절한 주석을 다는 행위와 같이 자신의 결과를 잘 표현하는 것입니다. 뿐만 아니라, 프로그램의 가독성, 효율성, 확장성, 일관성, 모듈화 등도 여기에 해당합니다. 이 부분은 상당히 주관적이지만 그러면서도 중요한 부분입니다. 컴퓨터과학에서 중요하게 생각하는 best coding practices를 참조하기 바랍니다.
- **점수 비공개**: 과제 점수는 팀원 평가에 따라 각자 다릅니다. 평가자를 보호하기 위해 과제 점수와 팀원 평가 점수는 공개하지 않습니다.

HK