PROJ 5

2018037356 안동현

2017012051 백채현

2021029798 바트온드라흐

1. 함수 설명

(1) int rsaes_oaep_encrypt(const void *m, size_t mLen, const void *label, const void *e, const void* n, void *c, int sha2_ndx);

EM을 구성해 공개키로 암호화해줍니다.

먼저 label을 해시해 lHash를 구합니다. 이후

lHash || 0x00.... || 0x01 || m으로 DB를 구성해주고 arc4random_buf를 통해서 seed값을 얻어냅니다.

해당 seed값을 MGF 해서 dbMask를 얻어냅니다. 이후 dbMask와 DB를 XOR 연산해서 maskedDB를 알아냅니다. 이제 이 maskedDB를 한번 더 MGF해서 seedMask를 얻어내고 seedMask와 seed를 xor연산해줘서 maskedSeed를 얻어냅니다.

이제 EM을 구해줍니다. 가장 첫번째 바이트를 0x00으로 해주고, 이후 maskedSeed, maskedDB 순으로 ||를 해줍니다. EM이 완성되었으므로 공개키 (e, n) 으로 암호화 해줘서 c에 저장해줍니다.

(2) int rsaes_oaep_decrypt(void *m, size_t *mLen, const void *label, const void *d, const void * n, const void *c, int sha2_ndx);

c를 복호화 해줘서 EM을 구해내고, 이로부터 DB를 구해서 평문 m과 길이 mLen을 복원해냅니다.

먼저 c를 복호화 해줘서 EM을 구해줍니다. 여기서부터 maskedSeed와 maskedDB를 구해주고, maskedDB를 MGF해서 seedMask를 얻어냅니다. 이제 seedMask와 maskedSeed를 XOR연산해주면 seed를 구해줄 수 있습니다. 이제 DB를 구하기 위해 seed를 다시 MGF해주면 dbMask를 구할 수 있고, dbMask와 maskedDB를 XOR연산 해줘서 DB를 알아냅니다.

이제 제대로 DB를 구해냈다면 패딩을 지나 0x01이 나온 이후부터는 평문 m이므로 m을 복원해주고 mLen을 복원해줍니다.

만약 주어진 label값을 해시한 lHash가 DB의 앞부분 즉 해시 부분과 다르다면 데이터가 변조된 것이므로 복호화에 실패합니다.

(3) int rsassa_pss_sign(const void *m, size_t mLen, const void *d, const void *n, void *s, int sha2_ndx);

mLen 길이 m을 EM으로 만들어서 개인키(d, n) 으로 서명해서 s에 저장해줍니다.

먼저 m을 해시해서 mHash를 만들어줍니다. 이후 salt값을 알기 위해 arc4random_buf 를 이용해서 hLen * sizeof()unsigned char) 길이의 랜덤값을 뽑습니다. 이것이 salt값입니다. 이제 0x00들과 결합해서 m 프라임 즉 M_P를 만들어냅니다.

이제 M_P를 해시해서 M_P_Hash값을 알아내고, 해당 해시값을 MGF 하여서 dbMask를 구해냅니다. 이제 0x00들과 0x01 그리고 salt값을 합쳐서 DB를 만들어내고,

만들어진 DB와 dbMask를 XOR 연산해서 maskedDB를 구해냅니다.

이제 마지막으로 maskedDB와 M_P_Hash, 그리고 0xBC를 || 해서 EM을 구성하고 만약 맨 왼쪽 비트가 1이라면 0으로 만들어줍니다.

이제 이렇게 완성된 EM을 개인키 (d, n) 으로 서명해서 s에 저장해줍니다.

(4) int rsassa_pss_verify(const void *m, size_t mLen, const void *e, const void *n, const void * s, int sha2_ndx);

s값을 공개키 (e, n) 으로 검증해줍니다. 검증하는 방법은 s를 공개키를 사용해서 EM으로 만들어준 후에, EM의 해시값을 얻어냅니다. 그리고 m 프라임 해시값을 알아낸 뒤 두 해시값을 비교해줍니다.

먼저 메시지 m을 해시해 mHash를 얻습니다. 이후 s를 공개키로 풀어 EM을 얻어냅니다.

해당 EM에서 maskedDB와 EM의 해시(EM_H)를 알아내고 이를 MGF1을 이용해서 dbMask로 바꿔줍니다. 이제 두개를 XOR 연산을 통해 DB값을 알아냅니다. 그런데 해당 DB값은 완벽하지 않습니다. 이전에 sign에서 MSB 비트를 1일 경우 0으로 만들었기 때문입니다. 따라서 DB의 맨 왼쪽 비트는 0이여야 하지만 0이 아닐 수도 있습니다. 이 부분을 참고해야 합니다.

이후 DB 에서 salt값을 뽑아내고 이를 이용해 0x00과 mHash, salt값으로 m 프라임 즉 M_P를 만들어냅니다. 이제 이것을 해시해주면 M_P_Hash값이 나오고, 이것을 EM_H와 비교해줍니다.

만약 두 해시값이 다르다면 인증 실패입니다.

(5) void MGF1(const unsigned char *mgfSeed, size_t seed_len, size_t mask_len, int sha2_ndx, unsigned char *T);

seed_len 길이의 mgfSeed를 ceil((double)mask_len/hLen) * hLen 길이만큼으로 바꾸어 T에 저장해주는 함수입니다. 먼저 해시 길이인 hLen을 구해주고, ceil((double)mask_len/hLen) 만큼 반복문을 돌아서 I2OSP와 매 반복의 i를 이용해 C를 구하고, mgfSeed || C 의 값을 tem 에 넣어줘 해시시켜 digest에 저장해주고 이 T = T || digest를 이용해 T를 완성시켜 나갑니다. 이때 저희는 64비트 정수에서만 한번에 ||가 가능하므로 여기서의 || 는 실제로는 메모리를 확인해가며 배열로 합쳐 저장해줍니다. 이때 주의할 점은 T의 길이는 mask_len보다 길수도 있고 같을수도 있습니다.

ceil((double)mask_len/hLen) * hLen >= mask_len 이기 때문입니다.

(6) void I2OSP(size_t x, unsigned char *t);

MGF1을 위해서 x라는 수를 32비트 정수로 바꾸고 4 단위로 나누어서 각각의 8비트 원소로 나눕니다. 이후 t[0], t[1], t[2], t[3]에 각각의 값을 넣어주고 리턴해줍니다.

(7) int countBits(size_t num);

주어진 메시지의 길이가 해시를 할 수 있는 크기인지 확인하기 위해 비트의 개수를 새어주는 함수입니다.

(8) static void sha_gen(int sha2_ndx, const unsigned char *message, unsigned int len, unsigned char *digest);

sha2_ndx 의 값에 따라서 함수포인터 배열에 각각의 해시 함수들을 담아놓고, 적절한 해시함수를 뽑아서 길이 len, 메시지 message인 데이터를 해시해, digest에 넣습니다.

(9) static int sha_len(int sha2_ndx);

sha2_ndx 의 값에 따라서 미리 정의 되어있는 해시 길이를 리턴합니다.

2. 실행 결과

(1) OAEP

```
wollong@wollong-virtual-machine:~/proj#5-1$ ./test
e = 02d335c2518f27f8eb54937c61dbbd2ec81500f509f0cf2a7dc7ea730f75d8d246e8f76cd1d8fa15261b408f5b07
be898649b3048d499378d4ba3a4c384b87c6b6158db018402ab0a4f09ca817d808760ccf371463a1a67d79df5829013b
682d214ad8fd17802c44ed9d80946fcc20656e2b5f082cd7310a7f5b014a297a840c7770dc9ad1c0f880f54d4b82448f
4e316c8895ae4558f9481936a75ae8b51fc8bcaf41d75f38939a5b5e7ebdf6ebb34568e87d2a16ee67a5ec246cf00357
86d669d69a43ec38defccbeadc03cc7ac4e930b14130f788dea90e60583d20e7c6373cd224227baea39d7f2a17c732d6
d21a4a7d24b51cf8a19e699f63da2d70655b
d = 01be33aa87f177797e4448f3955ae8cf3e559508af1c68547c631a26567580e3501cec833e3bbd45d5e5ff00147d
eca4ff01b9123d29dca8dc556c04bb50b1d0f0a261c2033a998194bff49c2e05d0a20c5bee178c9768a59a5bb5221807
4968cd4ba1c3ba858bee23949d084eb833d0974ac3af67344633bfca3d1ba51601c99cd1af5d90d80b8ba58c1c43f395
9136aaf500c074d9673e85e28db9405faf041f7367999256d4cf3f37bea850e8e7313d95364a8ac65b20c35e775a3537
2bd4ff18acfa70e8e1ed8bb8266fff09e271fabfa60dc98cf4da050861c93454ece2a27f58d34d6fd98f0308a683af6d
f3456220199a2be1936e4544900f2bf014ff
n = d5a980cab3c12e8dbd6ddfd011d9befe7276dfae0883481ae37d7a73a8b3b6aa90ca5e124cf1317a2b2603b07335
e4ee326f890143e17941106d235a68ed56478e863e0c19cb56d38c61613c1106d19ded2b86391b9c766ed4ff40a1d9bd
a87487e0bfca1b98d688db699665d47917ad98b6123101efb3bc1a87ca6711c1ca211c00a9cf1d80880f033f239bfc2e
5e5c28e6fe659e9145528cbef8a459ce1d9d3e389dc77e9377f8b0c2a989a85e2f7f364f3db53b107fca858423b7ce57
f048dad3d1d5236736a0209b1eb8a4c64ae9c90c9ee0362a071b59aef4567ac060be09fa18951cd93cf93e501b6a3928
a5624e4276e70a265532f98ba62256d40559
```

먼저 rsa키가 생성되는 모습입니다.

```
---
c = a391c1a8efd495da04b69c1b8ff874920282fb2ecef055e4a70cd9da16a26b997984626395eb56c59a824546c526
ca56e409c7cbce460ab664dcf1f9d07562ad6f9bd382b93fd66d4412bf9f946a4e098daa1dd168ea69c939efc6b90a39
2df5efea68aed60e1130487b9f0dc4f09f2d894e5c1f9e77d7a6db7ed5cf347ef56cf42de3f5f659af3ed768f0e1017f
94e9c01666469d4d6a0770643cb2444cc64f8ed9fa2879c772f0b169541c848b33bbc5217eeadcbdb96085bc22d1632f
092d5b53de6b46f4b781a9fc2661fd4fbc881aeb907031acb98b8f5b490a70b249ca146dbcafb6ac29fd20e5601709d2
9bc45e0867abd7be39dcbe3ab9b50306a390
m = 73616d706c65206461746100
msg = sample data, len = 12 -- PASSED
---
```

첫번째 OAEP 예제에 대해서 문제 없이 통과합니다.

```
---
c = 9c1da75cc8b71147ecb114c7ff4cf3ea8bc8cf3455a536043861a0a070cc3cadc67d94a8aee1042298c8c1d33bd0
8382d9fc2a4918c22fa2cd353926d3c20ce4e00eeda5524c0b566cc2731bb96b7ceddc1668b1c2cf552843f65f55d550
e74d8af15f7ae091d1ce2a34f1129e0323bfac64639d1ae0ce780fe8442b6a59b5de403e2dddfc9ae39d4187d64e4405
5482fa86ba698e620afb1112fc1f9d4a5365cabbc83061077a73c925b09f6a786f117b6322c8dcc5560d1ea1e2ab2e63
e259fec4c411c59f23cb1509992838a16c9dc726eba1876c6c0a896508e978ad8ef4444fa306385216d0458f47d9d76b
0e01007315f20b889c0c097a370734c65566
m = 6d617820646174610 06d61782b206461746100456d707479206d657373616765202d2d205041535345440a2d2d2d
006b6f7265616e20706f657400ec9ca4eb8f99eca3bc0045696e737465696e006d7367203d202573202d2d2050415353
45440a2d2d2d0a00525341455532d4f4145502052616e646f6d2054657374696e67004572726f723a206c656e203d2025
7a75202d2d204641494c45440a004e6f206572726f7220666f756e6421202d2d205041535345440a2d2d2d0073203d20
msg = max data, len = 190 -- PASSED
---
Encryption Error: 2, message is too long -- PASSED
---
c = 190c3750ee1e69afb58eba6df2c91c02affd51ea4f7b104c4a52d65ac279add2fe0274fba01d1996aa126d0fabb7
ff2c4569aa1995d6a22ddd76cdcc663b380b62ac7134ba189a9dab441e7d188b7040091feac4075d254434a96484789c
c6eeb54c45fbc498f1d311f18f15ef0b47531180f7cb7e1098eec07f81f53062b3c8265af56189902ab8dcfb7d38e143
168add406d105745f9f99acfaa6808219d508dc2a37b8cea2381fcf4804633b10a528c755b1ad1b74095693274088661
b7323ff1247f097932928f29824e8450b076faabf45579e0d6019a2c50d9fb298180b3d76ab895ea79301f0c9fa7a74b
f2fe3091340c0b344510db31ed787c3d526b
Empty message -- PASSED
---
m = ec9ca4eb8f99eca3bc
msg = 윤동주, len = 9 -- PASSED
msg = 배움은 경험에서 얻는다. 경험하지 않은 것은 정보에 불과하다. - 아인슈타인 -- PASSED
---
RSAES-OAEP Random Testing.....................................................................
.......................No error found! -- PASSED
---
```

나머지 OAEP 문제들 역시 문제 없이 통과가 되고 있고 윤동주 예제가 맞는 것을 보아서 표준 방식을 준수했음을 알 수 있습니다.

(2) PSS

```
.....................No error found! -- PASSED
---
s = 946f259de2dacb5956b19870092039cc7bb27b98974194a850056efa27936bf2878175eab8846cad8a58d12f082a
3eda4803b3efdc5e3bee080718df5d0fdf58302dd5ab8c1ab6a940bca4ca2819263986ca487ff4929ebbc48bf835d415
50861421c79ea693459528bc0546ba0ffe2ced99fcf03563323f3550a534ecaff97efc095dc60597dc85e09a0a99c867
0d6038a3a83033e955f51468f2c5d757db8aa279c8faa7be6035a2f5cdee82c547f92901b8ee7c7d31a9065d72aad82c
15b52726a8db7cdd66268a371fc06f725d76c01451ba99dcadb054157a700070d09cfe9698bba2d44d51fa6bdaac7825
ff0c5280dc6d93317ede51292e1024e77cff
Valid Signature! -- PASSED
---
Hash Input Error: 2 -- PASSED
---
s = a792d8e98509ee5988fec5da9f4e6e448c143ac895ed7da1fb083ebec6c9c91e3c76367cd993fbb5c4d191471d43
6d242350491af1acc04ee6b9a92876b8c6414323231b60c04373c2932d41607037a397afe07a9d03b3ca07f0a06ee2d8
d9f79c8ee6c9b8896003b49c37d4ccc332d58a8816c700b610ac043d1b24f27203d4ec948fd91c5e2b8230b72820c566
1cbe83fcc85254ded2526a5c22aa434442f0b221b6031cf23ff3b81aefd05c6db4e21a1ffda6f1508924032e5aba2a57
3d6b8fca887ef456d721c422fe34f7daf32dddd5c861fafe4f7ba9260fcf5535f2503541d623cdab2fb2a548457ca094
aa37d04eecdced0288254f4d94c56b96abf0
Verification Error: 5, invalid signature -- PASSED
---
s = d34d5efd564991be589861589b9d1f7b1a942eec3791e87b24fffb780ff7c9152492ecfd6279a0d82463776f6f19
b8da97039ebdaaad2f5e00991adadf0615a2118089b297d345c9a7370d3a47c9b25cb8b0affea155476ee8abfe71de02
60b98ee3c2a2db175f5a7185142fd23e4d0f9d28a181689460bc6be153f1a42a17de32506fbc844d12a430e48d762181
a34ec671fb96e774cd80db5aba70c48a67d540f07387e3b8c2ee22f2bc6795184c11a702435a9c5237474603ca18b1e4
508a281368b3aa8268e06030045effe2f78c535f6b89b34df74b03774d539b200848603800f50f1ead186e37c3b2fe4d
00f15420dbb0a7a87f7e5f4cc23313b8426b
Verification Error: 8 -- PASSED
---
s = 0000915f00e490f24aaf52505543d9dea2165efebc2a14929cfb40bb7769a4001aa545a67897d551077dfeefa7fc
491be61b20ca146ea8d7cd7e23bd83d76cc6d8813fe65a7c96f17dbb83ddba1520f31f36920f9c69d5e0f176eba36609
ab02bc2a5b255fe09642091861c74fdc4f420d701467f038d99e6844f554185fa198c54ab6258b6edd885a4973b51e89
07bad50deb02d4cdb60575eff5acd2bae82fb9a7e4b74368ca3c1bf97a11d78cd2c2f5c43682826aee485e5a07521f73
5dfad34198b0a199ff05ee921935e8e9543e69a8f593b2fa221b57ea8a5a9cadeb69adbeac85f48722f45c4c52006d3e
54ca713aaa030b0bbfbd1256a410ede5719f
Verification Error: 8 -- PASSED
---
Compatible Signature Verification! -- PASSED
---
RSASSA-PSS Random Testing.....................................................................
.....................No error found! -- PASSED
CPU 사용시간 = 497.3743초
wollong@wollong-virtual-machine:~/proj#5-1$ vi pkcs.c
wollong@wollong-virtual-machine:~/proj#5-1$
```

PSS예제들에 대해서도 문제 없이 통과가 되고 있습니다. 에러코드들을 보아하니, 예상 출력들과도 일치합니다.

3. 소감

2018037356 안동현

먼저 가장 어려웠던 부분은 MGF에서 배열의 길이를 신경써주는 부분과, ceil(a/b) 를 해줄 때 a/b 에서 (double) 로 캐스팅해주지 않아 오랜 시간이 걸렸던 점이 있습니다. 또한 PSS에서는 OAEP 와는 다르게 맨 왼쪽이 0x00인 바이트가 아니라, 0인 비트인 점을 놓치고 있었어서 에러 코드를 잡는데 시간이 조금 걸렸습니다. 또한 적재적소에 에러 메시지를 리턴하는 부분이 보고서를 해석하는 데 시간이 뙈 걸렸습니다. PSS 검증 함수와 MGF, I2OSP, countBits 함수들을 직접 구현하고 팀원들이 어려워 하는 부분들을 도와주고, 전체적인 디버깅을 하면서 모든 함수에 대해 깊이 이해할 수 있어서 좋은 경험이었습니다.

2017012051 백채현

RSASSA-PSS 기법 자체가 RSAES-OAEP 기법과 크게 다르지 않아서 쉽게 구현할 수 있었다. 각 블록들의 길이를 알고 원하는 해쉬 함수로 해쉬를 만들어 주었더니 한번에 테스트를 통과했다. 블록들을 이어붙이는 과정 또한 memset, memcpy 같은 메모리 계열 함수들을 이용해 깔끔하게 만들 수 있었다. 오류도 아예 블록들의 길이를 정해놓는 배열을 사용하여 피해갈 수 있었다. 중간에 만든 MGF 함수에 결함이 있어 다른 팀원이 만든 함수를 사용하였지만, 덕분에 MGF함수의 개념과 원리에 대해 잘 이해할 수 있었다.

2021029798 바트온드라흐

이 프로젝트를 통해 메모리 기능과 함수 호출 방법에 대한 이해를 높일 수 있었습니다. 처음에는 RSAES-oaep 함수를 작성하는 데 어려움이 있었지만, 인터넷에서 조사하고 도움을 구하여 코드를 작성하는 방법을 익힐 수 있었습니다. memcpy 함수는 처음 사용해 보았지만, 이 프로젝트를 통해 그 사용법을 익힐 수 있었습니다. 제 코드에서 일부 문제를 해결했지만 해시 함수와 MGF를 올바르게 처리하는 데 어려움을 겪었습니다. 다행히도 팀원들의 도움으로 이 문제를 극복하고 코드를 수정할 수 있었습니다. 이 프로젝트를 통해 코딩 지식이 확장되었고, 강의에서 배운 OAES에 대한 개념을 실제로 응용하여 암호화와 복호화에 대한 이해가 높아졌습니다.