시계이(가) 표시된 사진

자동 생성된 설명

암호학 프로젝트4 mRSA

암호학

2023/10/19

2021004866

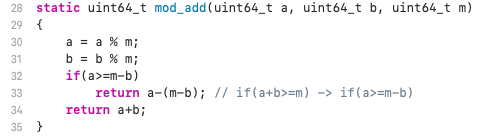
컴퓨터학부

지현도

오희국 교수님

\*본인이작성한함수에대한설명:

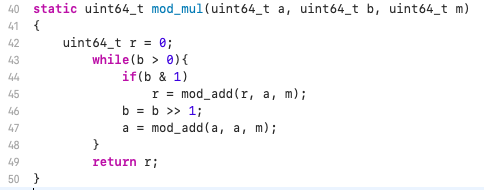
1. static uint64\_t mod\_add(uint64\_t a, uint64\_t b, uint64\_t m)



a+b mod m을 계산하는 함수. 이전에 구현했던 함수를 그대로 사용하였다.

오버플로우가 발생할 위험을 방지하기 위해 a+b>=m이라면 뺄셈연산으로 바꿔 사용한다.

2. static uint64\_t mod\_mul(uint64\_t a, uint64\_t b, uint64\_t m)

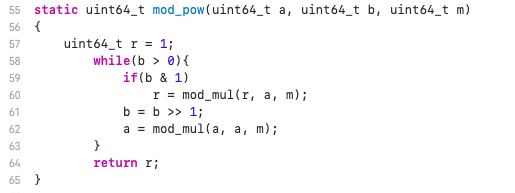


ab mod m 을 계산하는 함수.

이전에 구현했던 함수를 그대로 사용하였다.

매번 귀찮게 a\*2^i를 할 필요없이 a=2a로 계산해준다.

3. static uint64\_t mod\_pow(uint64\_t a, uint64\_t b, uint64\_t m)

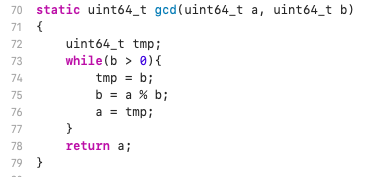


a^b mod m 을 계산하는 함수.

이전에 구현했던 함수를 그대로 사용하였다.

mod\_mul과 연산과정은 동일한데, a^b 는 a의 b번 곱셈과 같으므로 덧셈을 곱셈연산으로 사용해주었다.

4. static uint64\_t gcd(uint64\_t a, uint64\_t b)

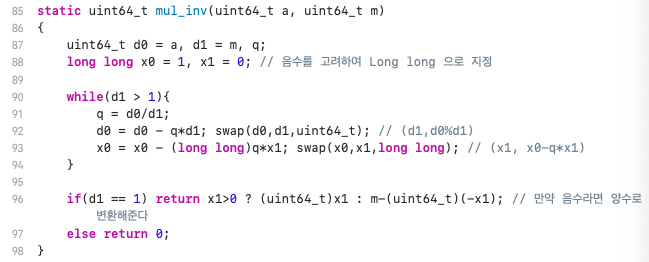


Gcd(a,b), 즉 a와 b의 최대공약수를 구하는 함수이다.

이전에 구현했던 함수를 그대로 사용하였다.

음수인 경우는 배제하였다.(unsigned 이므로)

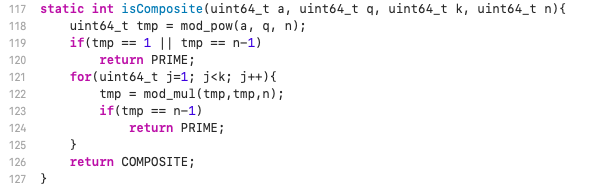
5. static uint64\_t mul\_inv(uint64\_t a, uint64\_t m)



위 함수 또한 이전에 구현했던 함수를 그대로 사용하였다.

역원을 반환할때 음수이면 양수로 바꿔주고 m에서 빼주었다.

6. static int isComposite(uint64\_t a, uint64\_t q, uint64\_t k, uint64\_t n)



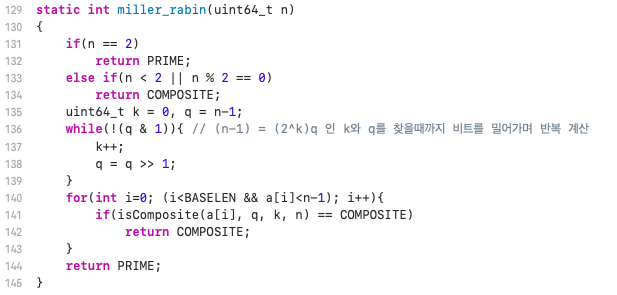
주어진 a로 n이 합성수인지 아닌지 확인하는 함수이다.

만약 합성수라면 COMPOSITE(==0)를 아니라면 PRIME(==1)을 반환한다.

q,k는 n이 홀수일때 (n-1) = 2^k \* q 인 정수이다.

위 함수 또한 이전에 구현했던 함수에서 반환값만 과제에 맞게 변경해주었다.

7. static int miller\_rabin(uint64\_t n)

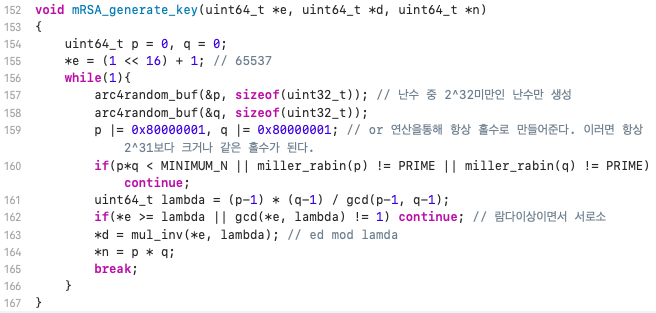


밀러라빈을 구현한 함수이다. 이전에 구현했던 함수를 사용하였다.

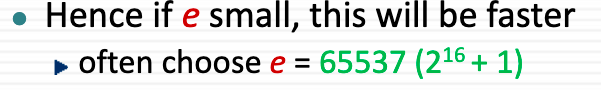
n 이 소수일 확률이 높다면(소수가아닐수도있다) PRIME을, 합성수라면 COMPOSITE를 반환한다.

상세과정은 밀러라빈 알고리즘의 과정과 동일하다.

8. void mRSA\_generate\_key(uint64\_t \*e, uint64\_t \*d, uint64\_t \*n)



RSA의 키 e,d,n을 생성해주는 함수이다.



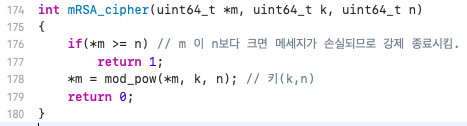
교수님의 강의노트에 나온것처럼 자주 선택되는 효율적인 e인 65537를 e로 사용하였다.

우선 p\*q >= MINIMUM\_N인 소수 p,q를 계산한다. 그 후 2^32 미만의 난수로 p,q를 초기화하고 소수라면 무조건 홀수이므로 or연산으로 홀수로 만들어준다.

계산이 끝난 p,q를 밀러라빈 함수를 사용하여 소수인지 확인해주고 동시에 e가 pq의 카마이클 람다보다 작고 서로소인지 확인해준다.

모두 만족하고 서로소라면 ed mod lamda =1 를 만족하는 d를 구해준다.

9. int mRSA\_cipher(uint64\_t \*m, uint64\_t k, uint64\_t n)



메세지 m을 키 k,n으로 암호화해주는 함수이다.

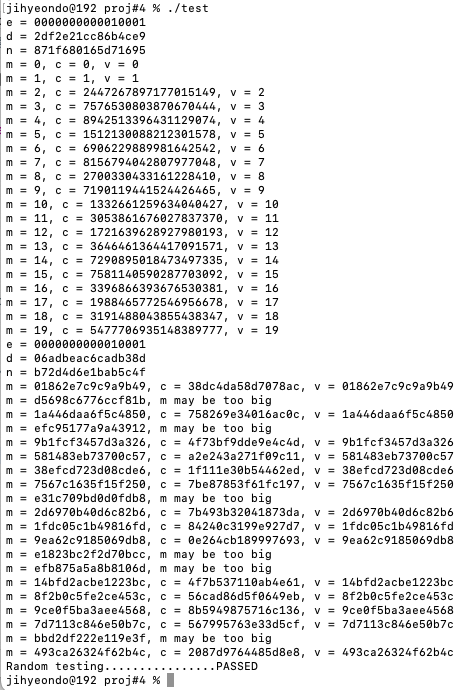
m^k mod n 을 계산하는데, m이 n보다 크다면 메세지의 내용이 손실될 수 있으므로 계산을 하지 않는다.

\*컴파일 과정



오류없이 정상적으로 컴파일되는 모습이다.

\*실행 결과물



실행 결과가 기대한 대로 잘 나오는 것 같습니다. 랜덤 난수로 계산하기때문에 교수님이 주신 예상 출력과 완전히 같지는 않지만 처음 키 e,d,n이 정상적으로 출력되는것으로 보아 올바른 64비트의 키를 잘 생성하는 것 같습니다.

의도한대로 만약 m이 n보다 클때는 경고 메세지인

‘m may be too big’ 이 출력됩니다. 이는 메세지의 손실을 막기위한 안전장치가 제대로 실행되고있다는 것을 알 수 있습니다.

랜덤 테스트도 오류없이 완료되었습니다.

\*프로젝트 소감 및 부족한 점

지금까지 구현해온 함수들을 총집합하여 RSA를 구현해본 것이 인상깊었습니다. arc4random 함수를 배우게되어 흥미로웠습니다.

지금까지 수행했던 AES같은 과제에 비해 비교적 최적화를 잘 이뤄내 테스트 시간이 오래 걸리지않아 다행이였던 프로젝트였습니다. 더 빠르게 최적화하는 방법에 대해 생각해보고 싶습니다.