Homework1

葉富銘　工海四 B08505045

1 CIA

(a) Confidentiality: Protection from unauthorized disclosure.
ex : Symmetric-key encryption provides confidentiality by protecting data against an eavesdropper.

(b) Integrity: Protection from unauthorized changes.
ex : Cryptographic hash function such as sha256 provides integrity check.

(c) Availability: Ensures intended users can access service.
ex : Denial of Service violates availability.

2 Hash Function

(a) One-wayness: Given y, it's computationally infeasible to find x such that y = H(x)
ex : Commitment scheme needs one-wayness to ensure binding.

(b) Weak collision resistance: Given x, it's computationally infeasible to find x$'$ $\neq$ x such that H(x$'$) = H(x).
ex : Integrity check using cryptographic hash function such as sha256 needs to ensure weak collision resistance.

(c) Strong collision resistance: It's computationally infeasible to find x and x$'$ such that x$'$ $\neq$ x and H(x$'$) = H(x).
ex : Birthday paradox is an application of attack against lack of strong collision resistance.

3 Multi-prime RSA

(a) $c^d = m^{ed}$ (mod n) = $m^{\phi(N)+1}$ (mod n) = m (mod n)

(b) Since the difficulty of factoring large composite numbers into prime factors provides security of RSA. For example, if 2-prime RSA uses two identical prime numbers, then the factorization of the public key is simple, and we can easily retrieve private key by the equation : d $\leftarrow e^{-1}$ (mod $\phi(N)$).

(c) Given C, N $\leftarrow n_1 n_2 ... n_i$:
$c_1 \leftarrow$ C (mod n_1)
$c_2 \leftarrow$ C (mod n_2)
$\cdots$
$c_i \leftarrow$ C (mod n_i)
then according to CRT, there exist an unique x :
$x \equiv c_1 \pmod{n_1}$
$x \equiv c_2 \pmod{n_2}$
$\cdots$
$x \equiv c_i \pmod{n_i}$
so we can solve x using CRT and x is the plaintext.

(d) Advantage :

i. Multi-prime RSA key-generation is more efficient than the regular RSA key-generation.

ii. Multi-prime RSA can have smaller key size by using multiple small prime numbers instead of two large prime numbers.

Disadvantage :

i. Multi-prime RSA may be more complex and difficult to implement and maintain than traditional RSA.

ii. Multi-prime RSA may be more vulnerable when the key's length isn't long enough, since Multi-prime RSA key is usually shorter than regular RSA key.

(e) Miller-Rabin is a method to test whether a number is a prime or not. For regular RSA, we use Miller-Rabin to generate each prime which has an expected runtime of O(n^4/log(n)). For multi-prime RSA, we generate k primes with each bit length (n/k), hence we get expected runtime O($(n/k)^4$/log(n/k)) $\cdot$ k = O(n^4/(log(n)$\cdot k^3$)), which is more efficient than regular RSA.

4 Fun With Semantic Security

(a) In security game, since regardless of the plaintext challenger encrypts, the ciphertext will have same suffix r, so we can simply remove it, and hence it is the same with ϵ = (Enc,Dec), which is semantically secure.

(b) In this question I prove by contrapositive that if Enc$'$ is not secure then Enc is not secure. I construct a security game, which Adv^{Enc} use $Adv^{Enc'}$ as subroutine:

i. Adv^{Enc} receives (m_0,m_1) from $Adv^{Enc'}$.

ii. Adv^{Enc} samples $b_{Enc} \xleftarrow{R}$ (0,1), samples r $\xleftarrow{R} M$ and sends $c_{Enc'}$ = Enc(k,m+r)||r to $Adv^{Enc'}$.

iii. Adv^{Enc} receives $b'_{Enc'}$ from $Adv^{Enc'}$.

iv. Adv^{Enc} sends (m_0+r,m_1+r) to challenger.

v. challenger receives (m_0+r,m_1+r), samples b and sends c_{Enc} = Enc(k,m+r) to Adv^{Enc}.

vi. Adv^{Enc} outputs $b_{Enc} \leftarrow \neg$ ((c_{Enc} & $c_{Enc'}$[:-|r|]) $\oplus$ $b'_{Enc'}$), where & is bitwise and.

If ϵ' = (Enc',Dec') isn't secure, then we can use it to derive b_{Enc}, and ϵ = (Enc,Dec) isn't secure. Hence it's proved that if ϵ = (Enc,Dec) is secure, then ϵ' = (Enc',Dec') is secure.

(c) Can't derive proof yet.

5 Simple Crypto

(a) Caesar cipher : Simply use brute-force to find all possible plaintexts with shift from 0 to 25 and see which one is most possible.

(b) Fence cipher : Derive a solution according to this article.

(c) OTP : I first xor c1 with m1 to get key, and then xor key with c2 to get m2. But since the length of m1 and m2 are not the same, when m1.length < m2.length, the length of the key is not long enough to derive complete m2, so I need to try to guess the remain plaintext or exit and retry again until m1.length $\geq$ m2.length.

(d) Two-way fence cipher :

6 ElGamal Cryptosystem

(a) With reused ephemeral key, we can implement chosen plaintext attack :
Assume we have prime P, generator g and public key y = g^x, where x is the private key. Given (c_1, c_2) which is the ciphertext of m, and we want to know about m. With same ephemeral key k, we can construct another ciphertext with m' chosen by us, and we have :
$(c_1, c_2) = (\ g^k, \mathrm{m} \cdot y^k)$
$(c_1', c_2') = (\ g^k, \mathrm{m}' \cdot y^k)$
We can compute $y^{-k} = (c_2^{-1} \cdot \mathrm{m}')$ (mod P) and obtain $\mathrm{m} = (c_2 \cdot y^{-k})$ (mod P).
flag: CNS{n0_r3us3d_3ph3m3ra1_K3Y!}

(b) It seems that it should exploit the information leak by the server and the property of homomorphic encryption, but I can't derive a feasible solution yet.

(c) Threshold-ElGamal cryptosystem with Shamir's secret sharing, can't derive a feasible solution yet.

7 Bank

According to line 126, 127 of the source code, It needs to find sha1 collision in order to create two accounts whose username have same sha1 hash, and I get it from two pdf provided by google, hence get the flag 1. And according to the paper, if once we find the collision, we can add modify these two texts and still get collision by adding any same suffix, so I append "I love CNS" to create two new accounts and get flag 2.

8 Clandestine Operation

(a) I use padding oracle attack to get flag1. By guessing the byte and xor it with the previous block's byte at same position. This procedure may take a while.
flag1 : CNS{Aka_BIT_f1ipp1N9_atTaCk!}

(b) Due to the mechanism of decryption, I modify block2 to change block3's plaintext from "icer||name:Cyno|" to "icer||name:Azar|". However, the change of block2's ciphertext will result in decode_error. In my opinion, since blocks other than block2 has either restriction or nothing to do with the decryption of block2, I can only try to modify block2 itself to get valid id, and since block3's plaintext can't be changed except first four bytes, namely "icer" from "icer||name:Azar|", so I use brute-force to find all possible combinations. This procedure may take a bit longer.
flag2 : CNS{W15h_y0U_hav3_a_n1c3_d@y!}