## Part A

1.(a)

p = 35219018721046519018661

q = 12532072192921

Thus, we have n = p*q = 441367285175991202374244491191098781

and $\Phi(n)$ =441367285175955983355510912599887200.

We use the magma to calc:

> phin := 441367285175955983355510912599887200;

> Factorization(phin);

[ <2, 5>, <3, 3>, <5, 2>, <7, 2>, <11, 1>, <13, 1>, <17, 1>, <19, 1>, <23, 1>,

<29, 1>, <31, 1>, <883, 1>, <2633, 1>, <187807381691899, 1>]

We find the smallest ‘e’ to satisfy the gcd($\Phi(n)$, e) =1 is 37.

Thus, RSA public key is 37.

$\Phi(n)$= q1*e + r1 = 11928845545296107658257051691888843*37 + 9

e = q2*r1+r2= 4*9 + 1

r1= q3*r2+r3 = 4*1 + 0

By substitution, we have,

1 = [e-q2r1] mod Φ(n) = (37-4*9) mod Φ(n)

= [37-4*(Φ(n)- q1e)] mod Φ(n)

= [37+ (-4) * (Φ(n) – q1*37)] mod Φ(n)

= [37+ ((-4) * Φ(n) + 4* q1 *37)] mod Φ(n)

= [(4*q1+1) *37 – 4 * Φ(n)] mod Φ(n)

Thus, d = 4*q1+1 = 47715382181184430633028206767555373. The corresponding private key for Alice is 47715382181184430633028206767555373.

(b)

I use the magma to help me calculate. I also refer some functions and operations from the magma documentation (Magma, 2017). The code is shown below:

```
// Two random primes

> p := RandomPrime(600);

> q := RandomPrime(600);
```

```
// Find n

> n := p * q;


// Find Φ(n)

> PHin := (p-1) * (q-1);


// Find the public key e (increase from 1)

> e := 1;

    repeat

    e := e + 1;

    until (GCD(Phin, e) eq 1);


// Find the private key d

> d := InverseMod(e, PHin);


// A random message m

> m := RandomBits(600);
```

```
// A blinded message mb

> r := RandomBits(600);

> Xe := Modexp(r, e, n);

> Mb := (Xe * m) mod n;


// Signature of m through blinded process

> ri := InverseMod(r, n);

> Mbd := Modexp(Mb, d, n);

> Sb := (Mbd * ri)    mod n;


// Direct signature of m using the private key

> s := Modexp(m, d, n);


> print "p:",p;

> print "q:",q;

> print "Phin:",PHin;

> print "e:",e;

> print "d:",d;
```

> print "m:",m;

> print "Mb:",Mb;

> print "Sb:",Sb;

> print "s:",s;

And the result is shown below, and the Sb and s of it is identical:

p:

272671652571510360879997969119710767002750917213653072415789

0884342553947371\

100581300416726700566262995489189769142117290371676553309614

8366431655434289821\

706311710142359884625384705510888323790138983759597049209589

0320129459788964928\

591114015710206512684831943296464403546982212672396463448042

1933766741851243623\

56753958561800876450948275391795157058122126773\

3

q:

327237856268060408422792286744646629447048425747566020726835

373502227957862 6\

5300238163747219639492410107599939459807648929965378508152031857829139626794083\

0537061340283067958331842571354391058898932938198022211765304392877691712410000\

7299352684109525435556518555187625655143698772916404762485502203455457792401554\

207211322665948008878528749671365839997243051263

Phin:

8922848705257041172979359673395852435776794303120452638804565070088272193\

3869252366526749853322119627745715580376055469219565997969124465985173532917282\

0907501541919338977354444808094802471451106939435309521014391064804785099104498\

8457314100882027697246602146681299605882474833964338923840160230439010211598154\

3688218035586234372228910892636095235023569810837762880421856901942598144764829\

4768279688923361093497191066695839026584845943951328377267947855698453844597157\

74276890550452951833602160614186148940287598420791431230438

85465764932956106336\

80527036625299020682842398013599612276689612751933871632080

71190658773686694285\

87019744302847138953524745564503875342324847088848673938917

09068392207076670005\

8979427502477784

e: 5

d:

17845697410514082345958719346791704871553588606240905277609

13014017654438677\

38504733053499706644239255491431160752110938439131995938248

93197034706583456418\

15003083838677954708889616189604942902213878870619042028782

12960957019820899769\

14628201764055394493204293362599211764949667928677847680320

46087802042319630873\

76436071172468744457821785272190470047139621675525760843713

80388519628952965895\

36559377846722186994382133391678053169691887902656754535895

71139690768919431548\

55378110090590366720432122837229788057519684158286246087770

93152986591221267361\

05407325059804136568479602719922455337922550386774326416142

38131754737338857174\

03948860569427790704949112900775068464969417769734787783418

13678441415334001179\

5885500495557

m:

52842015308239466552856772121720317841584416252185009230139

93881257530986146\

64275852082018553151279408135225215703090912501720158317499

02083990162136584690\

54795199647618545307373385183483880805834616815212897325615

51296715409726293745\

03663274345645514422281936234900422035292885069044785508258

72507112948957334096\

04194813363295851258613582205254303612800707537844433708683

95910369323303508969\

06495

Mb:

662434468797973523836734759544338159901748434363222990286139588916665222704\

810083925566791179112278735981737452739021792773928526147186679583924471492935 2\

8824366409009963594483328354732888839130799170572380873757102407228564940410497\

3498928560363641729305727337555021943776278617517940833805860252498390025773384\

5959256762017188573092328822946038985815851170111542342929075555708865769947874\

0746537678579059879414126169442327980045513196198901704971266650886536395632465\

1678799440575674057685900923369426114013101369041473499599021285040720644186628\

4198929752209417475811400031629747794573145709115991760838437169654946664200869\

9841794541285912163494563350726039747253962772595648528182608356612864844473946\

26905581193762

Sb:

3975467745199825420550530998864938684481283645068171168480002480620114700386\

7174329526803175186782896726838798283329983144607832521728799870784992155320599\

6856984371322042495470787151466835238847586927747581848018328290323225226690684\

7802752069480903987632299235833972270058687728171415040935315884262381026589515\

0901418911236297629346100891376523761032164824122738971672765762285920468525711\

8294804653070294411188506887949525834786832933064329162513729305411842563117102\

2805107642927280624456601508640687301721309715674746795517316846921083534959143\

2147686900016496384704460647436412696194192205859930745156354854921739561434648\

7600895149788456154644027102365989838003605198884235803377575229883657728872195\

6219913329076

s:

39754677451998254205505309988649386844812836450681711684800

02480620114700386\

71743295268031751867828967268387982833299831446078325217287

99870784992155320599\

68569843713220424954707871514668352388475869277475818480183

28290323225226690684\

78027520694809039876322992358339722700586877281714150409353

15884262381026589515\

09014189112362976293461008913765237610321648241227389716727

65762285920468525711\

82948046530702944111885068879495258347868329330643291625137

29305411842563117102\

28051076429272806244566015086406873017213097156747467955173

16846921083534959143\

21476869000164963847044606474364126961941922058599307451563

54854921739561434648\

76008951497884561546440271023659898380036051988842358033775

75229883657728872195\

6219913329076

(c)

This is not a secure method.

The known plaintext attack will be against this encryption method. The attacker could try to calculate all the represents of 26 alphabets by the RSA encryption algorithm $C_m = (m)^e \bmod n$ ($0<=m<26$). Then decrypt the ciphertext $C = \{c_1,c_2,c_3\ldots c_n\}$ by calculating the decryption algorithm $D(c_k)=i$ for all $0<=i<26$, $1<=k<=n$ and $c_k$ could be obtained from the last step.

And the countermeasure is, modifying the encryption algorithm to $C_m = (m)^{me} \bmod n$ ($0<=m<26$) for each alphabet.

2.(a)

One-way property: The hash function does not satisfy the requirement. Supposed we have the $h(M)=x$ where $0<=x<n$. We could easily find the message $M = \{x,0,0,0\ldots 0\}$ where $0<=x<n$ is a valid preimage.

Second image resistance: The hash function does not satisfy the requirement. Supposed we have the message $M1 = \{x,0,0,0\ldots 0\}$ where $0<=x<n$. Thus, we get the $h(M1) =x$. However, if we have the message

$M2 = \{0,0,x,0\ldots0\}$ or $\{0,0,0\ldots0,x\}$, we also get the $h(M2) = x$, but $M1 \neq M2$.

Collision resistance: Since the second image resistance is a kind of collision resistance and the second image resistance does not satisfy the requirement, this hash function does not satisfy this requirement.

(b)

One-way property: The hash function satisfies the requirement as it is difficult to calculate the m when we have $h(m)=x$ because the hash function consists of the square root and modulo.

Second image resistance: The hash function does not satisfy the requirement. Supposed we have the message $M1 = \{x_1, x_2\ldots x_n\}$ and we get the $h(M1) = y$. However, if we have the message $M2 = \{n-x_1, n-x_2\ldots n-x_n\}$, we also get the $h(M2) = y$, but $M1 \neq M2$.

Collision resistance: Since the second image resistance is a kind of collision resistance and the second image resistance does not satisfy the requirement, this hash function does not satisfy this requirement.

(c)

i. Message authentication codes

We can prevent the man-in-the-middle attack as MAC is a hash function. Both of Alice and Bob could authentic the opposite identity by using the promissory secret key to encrypt the message. Meanwhile, because of the confidentiality of the secret key, it is impossible for the attacker in the middle to intercept the message.

ii. Public key digital signature

Digital signature could also prevent the man-in-the-middle attack. Both of Alice and Bob could sign the public keys through their private keys. Then sending the message mutually. However, the attacker in the middle does not have the access to private keys. Therefore, he cannot intercept the message.

iii. Hash functions

Hash function does not provide the authentication function. Thus, it cannot secure DH protocol from man-in-the-middle attack.

3.(a)

In the First step, A sends message, identity $ID_A$, master key $K_a$ and encrypted identifier $N_a$.

In the second step, B sends message, identify $ID_A$ and $ID_B$, master key $K_a$ and $K_b$ and identifier $N_a$ and $N_b$ to KDC to request a session key for protecting the connection with A.

After receiving the request from B, in the third step, KDC replies to B with two messages encrypted by $K_a$ and $K_b$ respectively. This two messages contains one-time session key $K_s$, identify $ID_A$ and $ID_B$ and identifier $N_a$ and $N_b$.

And in the last step, A receives the message encrypted by the master key which KDC shares with A and Ka, including with the identity of B $ID_B$, identifier $N_a$, and one-time session key $K_s$.

After these four steps, A and B begin to know each other through the identifiers and the message is originated from KDC, and are able to start their protected connection.

(b)

①The security level of these two scheme are high.

②The steps of the scheme given above in the figure 1 is 4, which means it has better efficiency.

③Their orders of the detection of replay attack are different. The attack in the scheme in figure 1 will be detected in the end. Correspondingly, the attack will be detected at the beginning.

(c)

The scheme is secured. ①Identifier $N_a$ and $N_b$ are encrypted by the master key $K_a$ and $K_b$, which ensures that the request is not modified by others before receiving by KDC. ②Through the scheme, A knows that it connects with B, and message is originated from KDC.

(d)

pros:

①High security level and efficiency.

cons:

①Once the KDC is threatened, the connection will be risky.

②It is possible that the KDC simulates as the sender or receiver.

(e)

Supposed there are n senders and responders who want to communicate with each other (n users and n*(n-1)/2 communications). Since one pair of users need a session key, the number of session keys stored in the KDC is n*(n-1)/2 and the number of master key is n. Thus, the memory requirement of KDC is n + n*(n-1)/2.

And for each user, it is required to store one master key and (n-1) session keys. Hence, the memory requirement for each user is n.

## Reference:


1. Magma, (2017). [online] Available at: https://magma.maths.usyd.edu.au/magma/handbook/text/167 [Accessed 19 Sep. 2017].
2. Magma, (2017). [online] Available at: https://magma.maths.usyd.edu.au/magma/handbook/text/32 [Accessed 19 Sep. 2017].