CS4355/6355: Cryptanalysis and DB Security
Instructor: Kalikinkar Mandal
Faculty of Computer Science
University of New Brunswick

Student Name: _____ Matriculation Number: _____

The marking for each task is shown in [ ], and [100] constitutes the full mark. You must implement all tasks on your own. You are NOT allowed to use any code or part of code from Internet and use any library APIs that directly implement these tasks as a whole. Submit all source codes along with a README file containing instructions to compile and run your code via the D2L dropbox.

A1. [**45**] Please implement the key generation, encryption and decryption algorithms of the RSA public-key scheme either in C using GMP, Java using BigInteger or Python using gmpy library for a 128-bit security level. The large primes p and q are provided in the parameters section, where the sizes of p and q are about 1536 bits. For convenience, these three algorithms are described below. You are **prohibited** to use any RSA code available on Internet or other sources, and RSA APIs available in your programming language libraries.

Key generation	Encryption $c \leftarrow E(pk, m)$	Decryption $m' \leftarrow D(sk, c)$
1. Compute $n = pq$	1. Randomly message $m \in \mathbb{Z}_n$	1. $q' = q^{-1} \mod p$, $p' = p^{-1} \mod q$
2. Compute $\phi(n) = (p-1)(q-1)$	2. Compute $c = m^e \mod n$	2. $d_p = d \mod (p-1)$, $d_q = d \mod (q-1)$
3. Randomly generate e s.t. $e \cdot d \mod \phi(n) = 1$	3. Return c	3. $c_p = c \mod p$ $c_q = c \mod q$
4. Public-key $pk = (e, n)$		4. $m_p = c_p^{d_p} \mod p$, $m_q = c_q^{d_q} \mod q$
5. Private-key $sk = (d, p, q)$		5. $m = m_p \cdot q \cdot q' + m_q \cdot p \cdot p' \mod n$.
		6. Return m

Sample I/O:

```
The first prime is p = _____
The second prime is q = _____
The composite modulus n = _____
The encryption exponent e = _____
The decryption exponent d = _____
----------------------------
Encryption:
Plaintext (randomly generate) to be encrypted is m = _____
Ciphertext is c = _____
----------------------------
Decryption:
Ciphertext to be decrypted is c = _____
Decrypted plaintext is m = _____
----------------------------
```

A2. [**20**] Suppose two e-commerce websites named `https://cryptoa.io` and `https://cryptob.io` use the same RSA modulus N with two different public keys $e_a = 65537$ and $e_b = 65539$. They use the RSA cryptosystem for secure communication. A user wants to make some purchases from both websites using a credit card. To do so, the user encrypts the credit card number using the public keys e_a and e_b and obtains ciphertexts c_a and c_b, respectively and sends the ciphertexts to the respective websites. Assume that an attacker has some side information that the ciphertexts c_a and c_b are for the same credit card number and knows N, e_a, e_b, c_a, and c_b. Describe and implement the technique to recover the credit card number from N, e_a, e_b, c_a, and c_b.

RSA modulus and ciphertexts:

$N =$
4607153990015597302006011664133436377689420512204803704626324935751916693847322504468
9319636070419753456879267193913660367372551411418886789747571468428246330832154414868
2297079283463625678958237605023210457026820438645129274979026460998182524351770627262
1052631009742536020294201365750680523601148276913174723601697973618662120386649602464
3910544291400306716266958173953717383902879005480827334120889731684716665190145730373
3150546502525250824699006784152358574104315242517584825147004821128289626577001576661
9044165684584617105734271704390276479793297291193984189427029719069373303105942552991
1176672851052067876291088783092590822398123625637700840047981400870883519651277154402
7282713903013676662513648305366192220799601824182387485040055196216936365391890056761
7647022672883342219368416376391293556825350378781290879516010006766505326120693474210
6760183300252237504739255687616123000470173039409061474221359814589123 9

$c_a =$
8974527441135403774157444867365848706520818002224328852598116080044909523590167709751
3383012540032346283953189863061675936739427912215438860532640318594720396516548337307
3972262983745472875966366158613689843314450453287074901283710634125681887356566606992
4476761559788526860032759887350958012632051994545862812450869881985898611469809353189
0194029335302867638174697097148507460901602925425387863897923383252117030747454358800
3354268831406991784847480529662526815481808508807972614005098749201229799648271356379
1535863653208822205760282666981724975669627088199251333272075913984460246331522472443
4502326691456295873082683551428116010953103271686356160686653120943811896279012915210
8745084156954940384565372335106655005966694959491239283144689373545179858261295294427
0009803128749880246596989928669561595590051620585814707436572410330735482463303592908
8425897478035611149971777058966650698784623116834891416817231967554332

$c_b =$
6467243448388246119773044997263138026831115805522870239550768429193723828955992246567
9151846830440225291345678962039222670484403723017402390330172026773124600110978255007
2398062838204168244947619428669323606263431486140516423758544239950513077298052917843
8327965715647053947360753243173634819057727814161129354890536355510121638745360189518
8517091462028615026933764170226880044925280759809483903795425318957029462785485146342
8176409196242398024964547345363643971891104732611606843843715610588899385143971465629
2462113969356370791279687857062564155169839219139501787828159813038023370550811333930
7042583381329736043772827508993161755492930296363424990139576514800412088984148946474
4798202811127813227289097623100613031269856705163594361126397933003236666696815003440
5672970042174741196830153336503715993596354169614743497335750066963430911813868529102
8153847392853890726563339187964595380466023713492302411167729571747842

Sample I/O:

```
Print credit card number m = _____
----------------------------
```

A3. [**35**] This question is on comparing the encryption time taken by RSA encryption and AES encryption. Consider 10 Megabyte (MB) data which you can generate randomly. Encrypt this data separately using both RSA and AES encryption algorithms. You can use the same RSA parameters, encryption and decryption functions from **Assignment 1**. You can use the AES encryption and decryption API from a cryptography library in your programming language (See below).

- **RSA:** For the RSA encryption, use the public key $e = 65537$. As the plaintext message for RSA is from $\mathbb{Z}_n^*$, you need to divide the 10 MB plaintext data into about 3072 bits (or numbers of about 925 digit integers) blocks and apply the RSA encryption on each block.
- **AES:** Randomly sample an AES key of size 128 bits (or 16 bytes), $K \leftarrow \{0,1\}^{128}$. As the AES accepts a plaintext message of 128 bits, you need to divide the 10 MB data into 128 bits plaintext blocks and apply the AES encryption $\mathsf{AES.Encrypt}(K, M)$ on each block.

Please implement the above tasks and record the timing for both RSA and AES encryption and computed the overhead of the RSA encryption compared to AES.

Sample I/O:

```
----------------------------
The AES encryption time in milliseconds:  _____
The RSA encryption time in milliseconds:  _____
Overhead of RSA compared to AES: _____
----------------------------
```

Resources for implementations. Below are some libraries in C, Python, Java that you can use for large number operations.

- The GMP library. `https://gmplib.org/` (for C)
- The gmpy2 library. `https://pypi.org/project/gmpy2/` (for Python)
- The BigInteger class in Java

RSA Parameters for Assignment 1

$p =$
155549030353038563440076710635682130716698221846161019925955348608638035062627600676157270000882953304937057969022961024817982409882271950603161990809306160355329806173096440987193417530377824356457814364206972619848709697420964657658557824915380435549172852854714078669764653594464006956924599559295815611074962500577613244724385143511597466067372606767658726361401196699711053145393932706123980555389283618452372378553361497926189080509318701779259108193186 23

$q =$
152399300484575259702958032032073795143430317141511545179984152484707118114429564933421752862164704978551325104890152535135190738898259274367925807075120512998172909250387390237223664992921964000022047646657621144457646431793583487057504277534169773996941848047695964695615940137169527946313838727453390204035488818632154824807194458141652426270566377863026124826979239733032505886848220219880081751067357364116898003801793023473548827154966322910695258856532 97

Where can you find an AES API?

Programming language	AES API
C/C++	`OpenSSL`
Python	`cryptography`
Java	`javax.crypto`