CSE 565:Project 2 Report

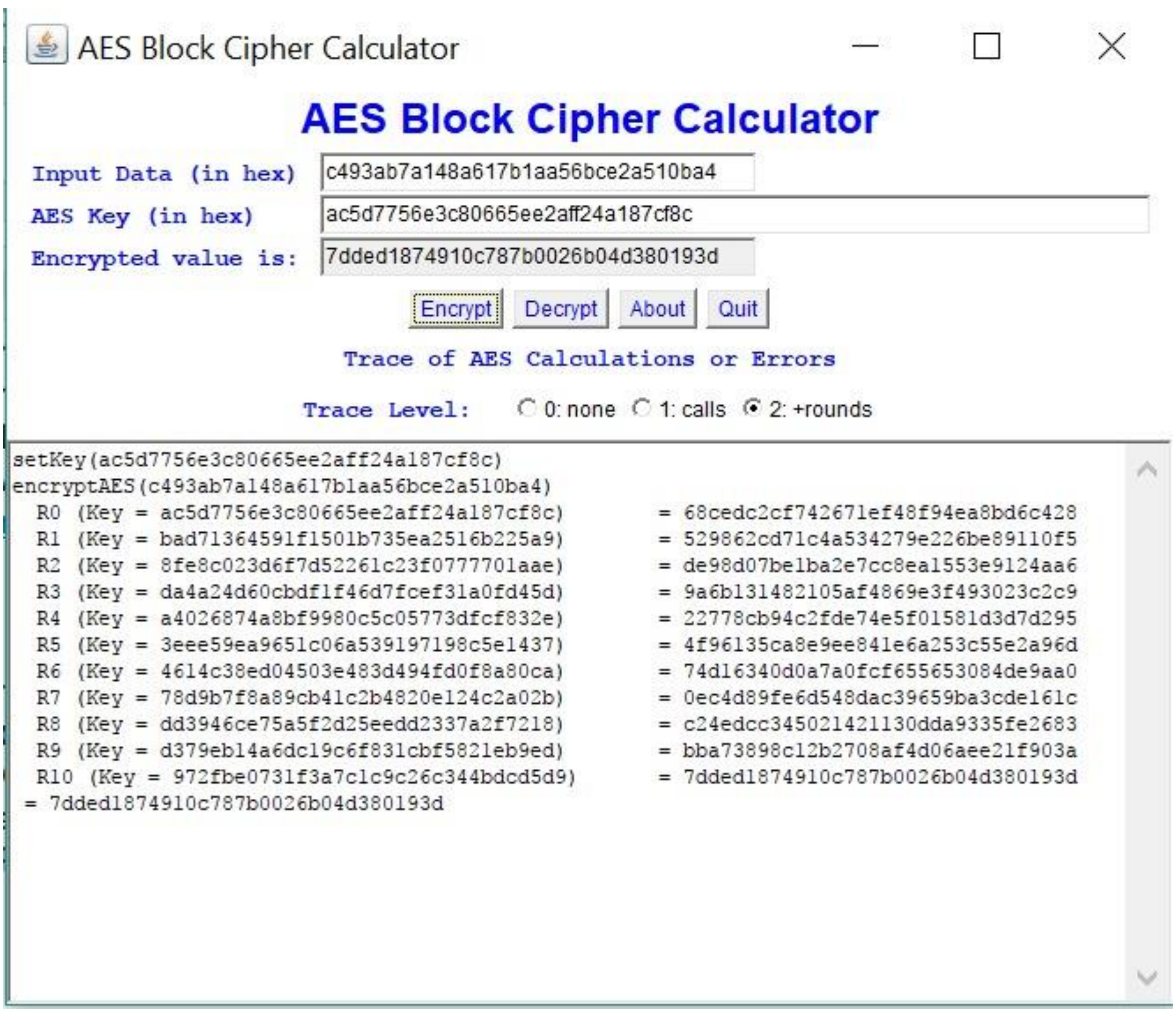
Submission by : Charanya Sudharsan
Sri Sai Sadhana Natrajan
Sneha Parshwanath

Part I – AES Encryption Algorithm

Part a - Block Cipher Internals

i. Encrypt the plaintext using the key given in your triple, with tracing of the round values. Note how the value of the **state** (result of each round) changes from round to round. What is the value of your **state** after round 4?

Plaintext that is given: c493ab7a148a617b1aa56bce2a510ba4
Key that is given: ac5d7756e3c80665ee2aff24a187cf8c



Change in values of the round key after each round :
R0 (Key = ac5d7756e3c80665ee2aff24a187cf8c) = 68cedc2cf742671ef48f94ea8bd6c428
R1 (Key = bad71364591f1501b735ea2516b225a9) = 529862cd71c4a534279e226be89110f5
R2 (Key = 8fe8c023d6f7d52261c23f0777701aae) = de98d07be1ba2e7cc8ea1553e9124aa6
R3 (Key = da4a24d60cbdf1f46d7fcef31a0fd45d) = 9a6b131482105af4869e3f493023c2c9
R4 (Key = a4026874a8bf9980c5c05773dfcf832e) = 22778cb94c2fde74e5f01581d3d7d295
R5 (Key = 3eee59ea9651c06a539197198c5e1437) = 4f96135ca8e9ee841e6a253c55e2a96d
R6 (Key = 4614c38ed04503e483d494fd0f8a80ca) = 74d16340d0a7a0fcf655653084de9aa0
R7 (Key = 78d9b7f8a89cb41c2b4820e124c2a02b) = 0ec4d89fe6d548dac39659ba3cde161c
R8 (Key = dd3946ce75a5f2d25eedd2337a2f7218) = c24edcc345021421130dda9335fe2683
R9 (Key = d379eb14a6dc19c6f831cbf5821eb9ed) = bba73898c12b2708af4d06aee21f903a
R10 (Key = 972fbe0731f3a7c1c9c26c344bdcd5d9) = 7dded1874910c787b0026b04d380193d
Cipher text = 7dded1874910c787b0026b04d380193d

The value of state after round four: 22778cb94c2fde74e5f01581d3d7d295

ii) Change AES bit 12 of the PLAINTEXT in your triple (ie change the 0 to 1, or 1 to 0 as appropriate), assuming AES bit numbering from left (MSB) bit 0 to right (LSB) bit 127. Encrypt this new plaintext value using the AES Calculator. Using the trace output, after each of the first four rounds list in a table how many **bits** of **state** differ from the corresponding values in part i (nb. you will have to convert between hexadecimal & binary and compare the relevant bits to do this).

When the bit 12 of the plaintext is flipped (0 changed to 1, or 1 changed to 0), the plaintext obtained is
==> Plaintext: c49bab7a148a617b1aa56bce2a510ba4
Key used : ac5d7756e3c80665ee2aff24a187cf8c

Input Data (in hex) c49bab7a148a617b1aa56bce2a510ba4

AES Key (in hex) ac5d7756e3c80665ee2aff24a187cf8c

Encrypted value is: 648a18e668891737a3e7246423bec824

Encrypt Decrypt About Quit

Trace of AES Calculations or Errors

Trace Level: 0: none 1: calls 2: +rounds

```
setKey(ac5d7756e3c80665ee2aff24a187cf8c)
encryptAES(c49bab7a148a617b1aa56bce2a510ba4)
  R0 (Key = ac5d7756e3c80665ee2aff24a187cf8c)    = 68c6dc2cf742671ef48f94ea8bd6c428
  R1 (Key = bad71364591f1501b735ea2516b225a9)    = 529862cd71c4a534279e226ba9ef2fca
  R2 (Key = 8fe8c023d6f7d52261c23f0777701aae)    = 4c0a7d443ec08ba32a564b0d795a027e
  R3 (Key = da4a24d60cbdf1f46d7fcef31a0fd45d)    = 1d31c459d7f431c990ed6e922ee71d73
  R4 (Key = a4026874a8bf9980c5c05773dfcf832e)    = 3df6dca62427d7ee636ae7fa677631e3
  R5 (Key = 3eee59ea9651c06a539197198c5e1437)    = bafcb2f91f150dbd58a33be17860b8b5
  R6 (Key = 4614c38ed04503e483d494fd0f8a80ca)    = 69bad583a0bca1f07126ce5088d72a99
  R7 (Key = 78d9b7f8a89cb41c2b4820e124c2a02b)    = 5b820fc178515df6eb7e0411d1f0011c
  R8 (Key = dd3946ce75a5f2d25eedd2337a2f7218)    = a928ee851c18cd722cdf9d6dfd615a1a
  R9 (Key = d379eb14a6dc19c6f831cbf5821eb9ed)    = 7eabd4d61529de6c58bdc521f7c2fce0
  R10 (Key = 972fbe0731f3a7c1c9c26c344bdcd5d9)   = 648a18e668891737a3e7246423bec824
 = 648a18e668891737a3e7246423bec824
```

Rounds	Plaintext (Original): c493ab7a148a617b1aa56bce2a510ba4	Plaintext (after bit 12 change): c49bab7a148a617b1aa56bce2a510ba4	Difference in bits of state after each round
R0	68cedc2cf742671ef48f94ea8bd6c428	68c6dc2cf742671ef48f94ea8bd6c428	1 bit
R1	529862cd71c4a534279e226be89110f5	529862cd71c4a534279e226ba9ef2fca	20 bits
R2	de98d07be1ba2e7cc8ea1553e9124aa6	4c0a7d443ec08ba32a564b0d795a027e	30 bits
35	9a6b131482105af4869e3f493023c2c9	1d31c459d7f431c990ed6e922ee71d73	35 bits
R4	22778cb94c2fde74e5f01581d3d7d295	3df6dca62427d7ee636ae7fa677631e3	43 bits

iii. Describe which characteristic(s) of a good block cipher design have been illustrated by this exercise, and how they are demonstrated.

Avalanche Effect has been illustrated in this block cipher design. Even a change in 1 bit in plaintext can cause the Cipher text to change drastically as we can see in the above example. This is a desirable property as it makes it cryptanalysis hard and difficult for an intruder to find the plain text.

Part b - Block Cipher Round

Plaintext that is given: c493ab7a148a617b1aa56bce2a510ba4
Key that is given: ac5d7756e3c80665ee2aff24a187cf8c

Finding sub keys :

k0= ac5d7756

k1= e3c80665

k2= ee2aff24

k3= a187cf8c

k4=k0⊕g(k3)

k5=k1⊕k4

k6=k2⊕k5

k7=k3⊕k6

g(k3) :

1) RotWord - 1-byte circular shift, k3 becomes (87cf8ca1)
2) SubWord - byte substitution on each byte of the input using the 16x16(Hex) S-box.

   S-box :

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	Fe	d7	ab	76
1	Ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	Af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	Cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	Eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	Fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	Aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	Cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	De	5e	0b	db
A	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
B	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	Ea	65	7a	ae	08
C	Ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
D	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
E	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	Ce	55	28	df
F	8c	a1	89	0d	Bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

SubWord(k3) = (17 8a 64 32)

3) XOR SubWord(k3) with a round constant,

Rcon[j] = (RC[j], 0, 0, 0) = (RC[1], 0, 0, 0) with RC[1] = 1 = 0X01

g(k3) = SubWord(k3) ⊕ Rcon[1] (j = 1)

g(k3) = (17 8a 64 32) ⊕ (01 00 00 00) = (16 8a 64 32)

The rest of the values can be calculated as follows:

k4=k0 ⊕ g(k3) = (ac 5d 77 56) ⊕ (16 8a 64 32) = (ba d7 13 64)

k5=k1⊕ k4 = (e3 c8 06 65) ⊕ (ba d7 13 64) = (59 1f 15 01)

k6=k2 ⊕ k5 = (ee 2a ff 24) ⊕ (59 1f 15 01) = (b7 35 ea 25)

k7=k3 ⊕ k6 = (a1 87 cf 8c) ⊕ (b7 35 ea 25) = (16 b2 25 a9)

On performing the initial AddRound Key stage we need to XOR the key and plaintext provided.

Plaintext that is given: c493ab7a148a617b1aa56bce2a510ba4
Key that is given: ac5d7756e3c80665ee2aff24a187cf8c

Key	**Plaintext**	**New state**
ac5d7756	c493ab7a	68cedc2c
e3c80665	148a617b	f742671e
ee2aff24	1aa56bce	f48f94ea
a187cf8c	2a510ba4	8bd6c428

Next stage , by looking into the S-Box :

45 8b 86 71

68 2c 85 72

Bf 73 22 87

3d f6 1c 34

Shift row transformation

45 8b 86 71

2c 85 72 68

22 87 Bf 73

34 3d f6 1c

Mix Column Transformation

02 03 01 01 45 8b 86 71

01 02 03 01 * 2c 85 72 68

01 01 02 03 22 87 Bf 73

03 01 01 02 34 3d f6 1c

$S_{0,0}$={02}.{45}⊕{03}.{2C}⊕{01}.{22}⊕{01}.{34}= e8

$S_{0,1}$={02}.{8b}⊕{03}.{85}⊕{01}.{87}⊕{01}.{3d}= 4f

$S_{0,2}$={02}.{86}⊕{03}.{72}⊕{01}.{Bf}⊕{01}.{f6}= 71

$S_{0,3}$={02}.{71}⊕{03}.{68}⊕{01}.{73}⊕ {01}.{1c}= a9

$S_{1,0}$={01}.{45}⊕{02}.{2c}⊕{03}.{22}⊕ {01}.{34} = 28

$S_{1,1}$={01}.{8b}⊕{02}.{85}⊕{03}.{87}⊕ {01}.{3d} = db

$S_{1,2}$={01}.{86}⊕{02}.{72}⊕{03}.{Bf}⊕ {01}.{f6} = b0

$S_{1,3}$={01}.{71}⊕{02}.{68}⊕{03}.{73}⊕ {01}.{1c} = 35

$S_{2,0}$={01}.{45}⊕{01}.{2c}⊕{02}.{22}⊕ {03}.{34} = 90

$S_{2,1}$={01}.{8b}⊕{01}.{85}⊕{02}.{87}⊕ {03}.{3d} = ab

$S_{2,2}$={01}.{86}⊕{01}.{72}⊕{02}.{Bf}⊕ {03}.{f6} = c8

$S_{2,3}$={01}.{71}⊕{01}.{68}⊕{02}.{73}⊕ {03}.{1c} = 4e

$S_{3,0}$={03}.{45}⊕{01}.{2c}⊕{01}.{22}⊕{02}.{34} = Fe

$S_{3,1}$={03}.{8b}⊕{01}.{85}⊕{01}.{87}⊕{02}.{3d} = 23

$S_{3,2}$={03}.{86}⊕{01}.{72}⊕{01}.{Bf}⊕{02}.{f6} = 35

$S_{3,3}$={03}.{71}⊕{01}.{68}⊕{01}.{73}⊕{02}.{1c} = 5c

State after mix column:

e8 4f 71 a9

28 db b0 35

90 ab c8 4e

Fe 23 35 5c

State	**Sub-key for round 1**	**State after round 1**
e8 4f 71 a9	**ba d7 13 64**	**52 98 62 cd**
28 db b0 35	**59 1f 15 01**	**71 c4 a5 34**
90 ab c8 4e	**b7 35 ea 25**	**27 9e 22 6b**
Fe 23 35 5c	**16 b2 25 a9**	**e8 91 10 f5**

We can verify this value using the AES calculator. The new state can be verified after step R1.

Block Cipher Modes of Use

Task C:
Cipher Block Chaining(CBC)
Key: CharanyaSudharsa
Hexadecimal format: 43 48 41 52 41 4e 59 41 53 55 44 48 41 52 53 41
Plaintext: This is a computer security project on block chain
Plaintext (divided into three 16 bytes blocks): Thisisacomputers ecurityprojecton blockchain
PT1: 546869736973616 36f6d707574657273
PT2: 6563757269747970726f6a6563746f6e
PT3: 626c6f636b636861696e

Steps for CBC :

Each block of plaintext is given as an input to the encryptions.

The cipher text of the previous encryption is XORed with the current plaintext block, making the cipher text block dependent on all plaintext blocks.

$C_i = AES_{K1} (P_i \text{ XOR } C_{i-1})$
$C_{-1} = IV$

Let IV be 00000000000000000000000000000000

First cycle of encryption:

According to the given formula, the first cipher text is taken to be the AES of key and the input which is taken to be an XOR of the first plaintext block (PT1) and IV.

PT1 XOR IV = 546869736973616 36f6d707574657273
Using the AES calculator, we encrypt the XORed value of PT1 and IV to get the first cipher text.
C_1: d7af52e2262925eb6a68d067eab11e09

Second cycle:
We XOR this cipher text C_1 with the PT2 to get the input for the second round.
C_1: d7af52e2262925eb6a68d067eab11e09
PT2: 6563757269747970726f6a6563746f6e
XOR output : b2cc27904f5d5c9b1807ba0289c57167
AES encryption with key
C_2: 285ab833614b7680cad57b54a5134011

Third cycle:
PT3 is not 16 bytes, but the key and input C_2 are both 16 bytes. We have to increase PT3 to 16 bytes. We can append it using the PKCS5 padding method, which says append the required value with the number of bytes that are short of the plaintext to make it 16 bytes long, in this case, 8 bytes have to be added, so append it with 0x08 eight times.
PT3 (appended with 08): 626c6f636b636861696e080808080808

C_2: 285ab833614b7680cad57b54a5134011
XOR output (taken as input for AES round three): 4a36d7500a281ee1a3bb735cad1b4819
AES encryption with key –
C_3: 24527d47a9a8c537da47abd5fda6d9c8
Encryption is complete.

Decryption:
In decryption, we use the cipher text that is obtained in each cycle, decrypt it using the key and XOR the value.
Third cycle:
C_3: 24527d47a9a8c537da47abd5fda6d9c8
Key: 43 48 41 52 41 4e 59 41 53 55 44 48 41 52 53 41
After performing decryption, the output is XORed with C_2.
Decrypted value: b63514bea24e68ed5bf23c977a24fdbe
XOR output: 4a36d7500a281ee1a3bb735cad1b4819
Since we had padded PT3 using PKCS5, we remove the extra bits to get the original undersized PT3.
PT3: 6e636f6d70736369

Second cycle:
C_2: 285ab833614b7680cad57b54a5134011
Decrypt this cipher text with the key and XOR it with C_1.
Decrypted value: d36b95ff5056f114bceae32cad657fc8
XOR output: 6f6a656374666f726d61737465727369
This is the value of PT2.
PT2: 656375726974797072
6f6a6563746f6e

First cycle:
C_1: d7af52e2262925eb6a68d067eab11e09
Decrypt with the key.
Decrypted value: 546869736973616 36f6d707365637072
After performing decryption, the output is then XORed with IV, just for the first block.
IV: 00000000000000000000000000000000
XOR: The XOR of any number with zero is the number itself.
PT1: 546869736973616 36f6d707574657273

By combining the three plaintext blocks, we get the original plaintext value .Since we are using PKCS5 padding, after the receiver decrypts the cipher text block 3, they will know that padding has been performed and will also understand which bits are relevant to the plaintext that the sender generated. Therefore, this process of padding helps to encrypt/decrypt any undersized block along with the usual 16 bytes blocks.

Cipher Text Feedback (CFB)
In this mode, the cipher text is taken after the XOR operation between cipher text and plaintext. The process before the XOR involves an AES step with the cipher text of the previous block. An initial vector (IV) is taken again, with all 0's.

PT1: 54686973697361636f6d707574657273
PT2: 6563757269747970726f6a6563746f6e
PT3: 626c6f636b636861696e

First cycle:

IV = C_{-1} = 00000000000000000000000000000000
Key: 43484152414e594153554448415253 41
Taking the IV and the key, we calculate AES(C_{-1})
AES(C_{-1}) = 6e673f98072f6590aef8ccba6072576c
Using the AES value computed and the PT1, we can calculate C_0 using the formula given below.
C_0 = PT1 XOR AES(C_{-1}) = 3a0f56eb6e5c04f3c195bcc90511271e

Second cycle:
Taking C_0 and key, we calculate the AES(C_0)
AES(C_0) = 7d3ebf829251db514bc8306385aeb146
C_1 is computed as below:
C_1 = PT2 XOR AES(C_0) = 185dcaf0fb25a22139a75a06e6dade28

Final Cipher text = PT3 XOR C_1
Cipher Text, C_2 = 7a31a5939046ca4050c9520eeed2d620

Decryption:
We take 's' bits of the cipher text which are then XORed with the current cipher text block to get the plaintext back.
P_i = C_{i-1} XOR s bits

PT3 = C_2 XOR s bits (which is taken to be 16, same as cipher text)
= 626c6f636b636861696e080808080808

We know that the final plain text block had been padded using PKCS5 padding method so that it becomes 16 bytes. So after removing the padding, the receiver will get the original plaintext block.
PT3 = 6e636f6d70736369

PT2= C_1 XOR s bits
= 65637572697479707 26f6a6563746f6e

PT1 = C_0 XOR s bits
= 546869732069732061636f6d707574657273

By combining the three plaintext blocks, we can get the original plaintext that the sender had computed. In this method, the 's' bits which we take are essentially the AES values of the cipher text that we had computed in the encryption step.

Original Plaintext: This is a computer security project on block chain

We can conclude that CBC and CFB use similar process encryption and decryption, using iterative exclusive-or operation.

CBC yields the cipher text after the AES operation is performed, whereas for CFB, the cipher text will be taken after the exclusive-OR is computed.

CFB is advantageous over CBC as it uses error recovery when the cipher text is garbled.
Both can be used for decryption on multi core processors.

Part 2

Section 1

Describe the reasons for having the decryption process as explained in Sec. 3, Phase 1 for embedded devices.

Any cryptographic system used in embedded system is bound to various restrictions.

The secure RSA-CRT decryption has to be computed within a reasonable time. This is because the micro-controllers in running under a clock frequency of only a few megahertz. Using CRT with RSA allows the recipient to compute the exponentiation more efficiently. This is more efficient than computing exponentiation by squaring even though two modular exponentiations have to be computed. The reason is that these two modular exponentiations both use a smaller exponent and a smaller modulus. As this makes the decryption process faster, it is commonly used in embedded systems

The code size of decryption is small enough and can be run on ROM of microcontrollers that have small ROM size.

The public key is published on the server. The overhead of security of protecting the public key lies upon the server and not the embedded system.

REFERENCE:

https://en.wikipedia.org/wiki/RSA_(cryptosystem)#Using_the_Chinese_remainder_algorithm

https://iacr.org/archive/ches2008/51540128/51540128.pdf

Section 2

Given that $M_2 = C_q^{\beta} mod$ q and $M_2{'} \neq C_q^{\beta} mod$ q where M_2 is the correct message and $M_2{'}$ is the faulty message.

We also know that M is computed from M_1andM_2.

Let us assume that $M'$is computed from $M_1$and M_2'.

Based on Lenstra's improvement one faulty message and a cipher text is enough to find the factors of N by simply taking the gcd(C – M'^e, N). Here M'^e is the cipher text obtained by encrypting the faulty message, C is the cipher text of the original message and N is the modulus value.

By finding the gcd(C – M'^e, N) we get the value of p. Since N = pq, we can easily find q.

We also know that private key d = e^{-1} mod $\varphi(N)$ where$\varphi(N) = (p-1) * (q-1)$. Therefore the private key d can be exploited and this algorithm is vulnerable to fault attacks.

Section 3

The reason for having such an encryption protocol is that it improves the performance of the encryption process. The basic advantage is that the large modulo exponentiation is split up into two much smaller exponentiations, one over p and one over q. Hence it is much faster than other encryption methods.

But this protocol also has the same weakness. Here the cipher text C_1 depends only on the prime number p and the cipher text C_2 depends only on the prime number q.

By following the same Lenstra's improvement we can compute gcd($C_1 - C_2$, N) to find the factors of N. Using the values of p and q, the private key d can be computed from the equation d = e^{-1} mod $\varphi(N)$ where $\varphi(N) = (p-1)*(q-1)$. Thus the private key d is exploited.

REFERENCE: Dan Boneh, Richard A. DeMillo, Richard J. Lipton: On the Importance of Eliminating Errors in Cryptographic Computations. J. Cryptology 14(2): 101-119 (2001)

Section 4

1) We know that in RSA,the modulus N = p * q.The modulus is a part of the public key hence we first derive the value of the modulus from the public.key file given by using the following SSL command:

```
gurukrupa@gurukrupa-VirtualBox:~/Documents/openssl-1.0.1g$ openssl rsa -pubin -inform PEM -text -noout < ../public.key
Public-Key: (2048 bit)
Modulus:
    00:e8:52:42:77:0a:66:69:8c:64:49:61:5a:d4:8f:
    70:e5:ff:7f:49:ca:45:33:43:7d:85:36:7e:1a:f3:
    8f:31:aa:35:94:8e:b3:3f:97:88:f7:16:4a:1d:d5:
    c3:87:5b:f8:6b:69:3b:d8:cc:82:e2:cb:cb:d0:1c:
    f7:d1:b4:51:ef:67:cb:72:90:fa:79:0e:e1:02:24:
    e3:72:5b:37:b6:bc:3d:53:56:da:9d:0f:ba:c1:e0:
    6b:b2:6f:f2:43:03:d9:06:d3:c9:66:c8:1b:19:9a:
    78:b9:ef:02:2b:0f:b9:28:e5:82:fc:0c:e0:29:57:
    f6:b1:64:21:01:f9:2e:83:4a:ab:47:24:9e:e4:08:
    c2:91:d3:fc:e8:72:c1:44:69:12:31:37:f4:da:49:
    28:00:75:03:36:47:20:69:f4:e2:4b:4a:0e:3e:e5:
    15:85:ae:78:68:43:a3:c0:39:61:c2:12:a1:e3:94:
    d2:71:e8:26:14:c4:e7:aa:1d:5d:a4:16:01:1f:9b:
    40:81:a8:e4:70:65:75:1a:de:de:51:d0:90:97:fb:
    8a:41:ac:be:2e:54:5c:b6:d4:04:40:1d:59:16:c3:
    f6:86:16:e9:66:79:b3:5f:77:74:a9:e4:42:b1:98:
    74:14:b0:22:ee:06:f0:0f:ac:3d:dd:b6:14:19:43:
    e5:53
Exponent: 65537 (0x10001)
```

The value of the modulus given in decimal is 65537.

2) We then convert the value of modulus N into decimal which gives 10001.

3) Since C2 is faulty and C1 is correct, we use the section 2.2 in the paper in the reference mentioned (Dan Boneh, Richard A. DeMillo, Richard J. Lipton: On the Importance of Eliminating Errors in Cryptographic Computations. J. Cryptology 14(2): 101-119 (2001)) ,to conclude that

GCD(C1- C2,N) = p

C1=
3AE81964C8ECF1524B47C42CB0ECD2A3B6768DCCD55960D7FF0A998F839B8C312A2CD821C270AE961777DD4D
D50AA631FE823A8AFD914911ADF69C1C6CFDA3B3AED01DAD372CFDD6E9F63A4CC39E1A455CBFD04DEA72BF07
C4790D5FEC469198CE28113D6D38A7BACED9D3C3695AB27CBC5AB434AA8D2B5F53F66A383E079DDAED485D4A
2B446E410EAFCADBBA9F159494C28C4A19FD416DFF90F8C141E96D8260F8E6E0901832E31899C48CE0CBDAE6A2
4595A19A01E490C87E7B48860E09006920D8EF7384217358C6DB90638D6E8CBC795A091240F24105D8F3B27FE4
B98FE9A507E00590B4CDED41777B1B8967B0F752231E0E856B8F0132BDE30A6E082E

C2=
391E0340E5931A202012572DDACAD877E5AF3A1D846B70C1E64E3041F9AC0A3C7E8F82621DF908EADCA44FE777
A6B1C799610BE829E13CA233982FD268034ADDB5A79FA19F984631FDF3A61D32FC75ED77176C7A0B719504E804
076DCA66F10111AA124A7EFE743ADA75DDA2EC53F3C28882A7724928685918261739F960A3648AA3EADC426181
AA146A8BA0FF20F1C53DE2113E0196AF09595DC2AD1A0FE12096FF681F61363044615A7F72EDF1F8C6531055E66
C1E5F4498434C731D2308FECAE46C779379EA3D7D7A5F1C2A0EFEB5BC1B8A4AF4FB21FCE1DAE943C27043E8664
2B3B1E6B889A31E7C4BC01BC2EBAE4DC8432344532567D1D3DF8B9BCAFCBF

C1-C2=
1CA1623E359D7322B356CFED621FA2BD0C753AF50EDF01618BC694D89EF81F4AB9D55BFA477A5AB3AD38D665D
63F46A65212EA2D3B00C6F7A5E6C4A04FA58D5F9287E0B9794B7A4EC02942F90A1A457E5A863D3DF012A02DC75
05F221DFA097BC7DFEF2EE3A337FF463F6207D06BEBA33D80CC26164C3063BD052FE44A6FA7662A4726DE8E2EC
96FA453F3ABB7E23CF56E07B0C18669264A633361427D98C61C9F97EC12EE2029EB73F4519F2D9E2204F353FBB2
DE3854C303B2ED568EB00019E3C6C77E00A3735DB4C7C74397E6FD7005DCFBE1D45D04423FE0A6EBD7475A783
62542E1ED82B2ACEC52BBAFFC67C02A989DFFACA3246272F5EDE576EA30B6F

We get the value of p is hexadecimal as follows:

p=
F37E1660AD957C49E0CB831D993569B9288EA5B1F41F8AC7A4E45B696F3DEB8C2F0EE6EF2B1FE66CCDEBB508CD
A3F611AA627352030D89FBDD8107790CDEC7178069673B2D8C7EC7F24D0556FDF9277B775ECF1EF9CAEA034401
AAEE139B68E9D1DB61AE887D9AF4CF78326E96C3C20BC1199F5BD59CB7415921137BAB738357

4) We next derive the value of the prime number q by dividing the value on modulus N by p. In order to do this,we first convert the values of N and p to decimal.
N/p = q

q=
17152167084450574852007809804869942743144720244739002843918288568780389599412011567684456945
03103418357636880604513480185516834820455123418559724688297134415746229205922307617180898927
07849593188829588731712908801079272698459745939569379886544109127546607681279888629526192822
444046787008855336604753568140389(in decimal)

q =
F4414584B643F161141FD6FCB293E61D98E51702AB1088E057CA60A86984405294AE99713C04315F384C0B5A533
530483071C491BE29105C53B392A7F671535C9420A6BC618BBC15A832DDDC6B4E156E8CE358DC648C02D1A807
544F0B0FED2A9B1E585C4B27FF4008F44EDA7B59A0910B2F0DD37915D45A617FB54B151F0C65(in hexadecimal)

5) We can now calculate the value of phi(n) = (p-1)*(q-1)

phi(n) =
E85242770A66698C6449615AD48F70E5FF7F49CA4533437D85367E1AF38F31AA35948EB33F9788F7164A1DD5C38
75BF86B693BD8CC82E2CBCBD01CF7D1B451EF67CB7290FA790EE10224E3725B37B6BC3D5356DA9D0FBAC1E06B
B26FF24303D906D3C966C81B199A78B9EF022B0FB928E582FC0CE02957F6B1642101F92E8348C387C8B9802F54E
6DF118E58757B193B6FC43825A9F7ECCD06878B0E9132B66C8650BE84AE6196AC620BE35D18889BB8C70F5CEEB
0B18BBC93B30FFC5A53FB8D0B11328A19CC3587DA9AFBAAE88953ADF74819B15FD76787CACB0502FEADC0AF89
8C5CDE92D4192A9F08289B309435D7486775BEB83D8410833CED4D58B15598

6) We can now calculate the RSA private key ,d as $d = e^{-1} \bmod phi(n)$.We get d as follows :

D=
BD69FC08C470F553629CB58471D3B58FC03D4EB35CB24C2F75C21514F249688BC6FDA54F1DE4F5E5C6C58D07B
150963AB790681BFDBA252155BE0B303E48CF98EABE645D31EA2BA866B01EC528FC06BFF91C0EAF54E8F332894
0258EC22DC3605A01A8AE652E3B34C783890BF281F24C16BC31B669B92B9FD5C7B665BE7D1ECADFA8C08E05A6
1189DE8B98CC2494C5E0C0E27188BBC888C82CAFAC3FA7407A9D86F12434575CF794FAA788AB698A4FB45A8FFB
0C1D01D4D55E72DAF0BBD0C7429AA85DD55CED3323DDA708D3B29DEC9CE50562F54C17FAB02B365592B8847F
9A5F0212E1E1428E76428D4B562448629170D5F1991AEC04D5D5BA821BD2D95AD9

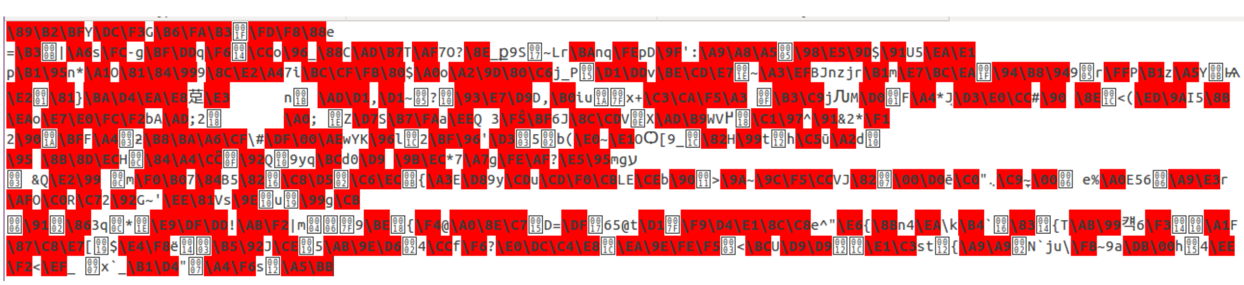
7)Now the encrypted AES key given is the file misc .text is

CipherText=
19303A82CBC50A56DC22F9AAFC554DA2EA632E33BEE1D33C35EDB13269BA0FD2FA791744E86EDA7FC1CB15433
F1232F86A42AFCD5470215CCF0D05096CE1B8F075E6DC45DF74896345297FCC145A4765AEAEA78BABAA6441EA
D3A2E73B37931DC7C07D4E04B7115284C7C04C85A61C7E555194D0F4EE762D47A8AEEC2EFCD75EE45E3E6A65F
876F9A67AA01016F3CE9552D8F0B50CC150C70333AA6AC3A4AC8860D2879CAD8439566F0FFDA32612CB75DD9C
1B456A4E1828582F05932F495452F19D71F300F2F48B8C1F8CDE1B1B8D8ADA3F6FF506E10D5D18D91D61402BC3
6756A88196381FF795980EEE932A179525264E3A968F0ABE9EDBE672560C41833A

We apply the private key d and compute $M = C^d \bmod n$.The AES key(plain text) decrypted form is
PlainText = FABADABABECAFEDEADBEEF0102030405

8) Now that we have 128 bit AES key, we use this to decrypt the test given is encrypt.aes using OpenSSL:

The resulting decrypted file contents



Reference: Online tools used in conversion

http://www.binaryhexconverter.com/decimal-to-hex-converter

http://www.dcode.fr/big-numbers-division

https://www.mobilefish.com/services/rsa_key_generation/rsa_key_generation.php(to calculate phi(n))

https://www.mobilefish.com/services/big_number/big_number.php

---

Contributions: Charanya Sudharsan : Part 1 and report of part 1.

Sri Sai Sadhana Natrajan : Part 2 analysis and report phase 2 and phase 3 and screenshots.

Sneha Parshwanath : Part 2 analysis and final project report.
