CRYPTOGRAPHY & COMPUTER SECURITY (CS 265)

SPRING 2014

SAN JOSÉ STATE
UNIVERSITY

IMPLEMENTATION PROJECT

ATTACKS ON RSA PUBLIC KEY CRYPTOSYSTEM

(Guided by Dr.Thomas Austin)

By

ANUMEHA SHAH

VEENA REGURI REDDY

NEHA RAJKUMAR

TABLE OF CONTENTS

1. INTRODUCTION

The RSA cryptosystem was invented by Ron Rivest, Adi Shamir and Len Adleman. The RSA got its name from its inventors. This cryptosystems is most commonly used in authenticity of digital data, Email. It is also used to provide secure web traffic. Electronic card payment system also used RSA extensively. RSA cryptosystem security rests on factoring problem. In order to break RSA, an attacker need to factor 2 very large prime numbers, and till date no one got succeeded in finding a proper solution of factoring RSA. However in last 20 years many vulnerability of RSA was analyzed and many attacks were devised. However these attacks are mainly due to improper use of RSA. In our implementation project, we are also aiming to attack RSA cryptosystem by exploiting its improper uses.

2. GENERATING RSA PUBLIC AND PRIVATE KEY PAIR

To generate RSA public and private key pair, we calculate a product N = pq, where p and q are very large prime numbers.

Next, choose an exponent e such that e is relatively prime to (p-1)(q-1). Finally choose d such that ed = 1 mod (p-1)(q-1). Now the public key pair is (N,e) and private key pair is (N,d). Message in RSA are treated as number. To encrypt a message M, one can compute $C = M^e \bmod N$ and an encrypted message C can be decrypted by computing $C = M^d \bmod N$. RSA can be broken if d is known or if pq can be factored to find p and q. Factoring RSA is very hard, and d is secret, so attacker can try to exploit the some other vulnerability of RSA which does not require knowing d and factoring RSA.

3. PROBLEM SPECIFICATION 1

Alice wants to invite her friends for her birthday party. She wants to send encrypted message to her friends, so that an attacker Trudy cannot know about the location of party. Alice used RSA public key cryptosystem and her friend's public key to encrypt the messages. However instead of using different modulus N for all the messages, she decides to keep the N value same to reduce complexity and time for calculating cipher texts. However Trudy is able to decrypt the message.

3.1 COMMON MODULUS ATTACK ON RSA

Problem specification 1 is an example of common modulus attack.
A common modulus attack on RSA can be made if a message M is encrypted using different exponent value but same modulus N.

Message to be encrypted is M
Modulus is N = pq, where p and q are very large prime numbers
Exponent values are e1, e2
Ciphers C1 and C2 can be calculated as
$C1 = M^{e1} \bmod N$
$C2 = M^{e2} \bmod N$

This looks secure as Trudy does not know d and factoring RSA is very tough. However it is not secure. Trudy can use extended Euclidean theorem to decrypt the cipher text c1 and c2 to M.

3.2 EXTENDED EUCLIDEAN ALGORITHM

Extended Euclidean theorem states that if for two numbers m and n, gcd(m, n) = r , then there exists integers and b such that

a*m + b*n = r

Trudy can use this theorem and calculate two integers a and b such that a*e1 + b*e2 = 1, provided that e1 and e2 are relatively prime to each other and gcd (e1, e2) = r. Now Trudy can calculate message M by calculating $M = C1^{a} \bmod N * C2^{b} \bmod N$.

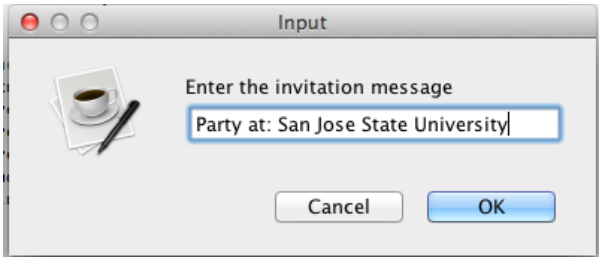
Where $C1 \bmod N = M^{e1} \bmod N$ and

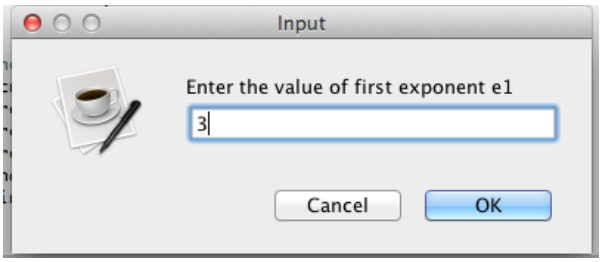
$C2 = M^{e2} \bmod N$.

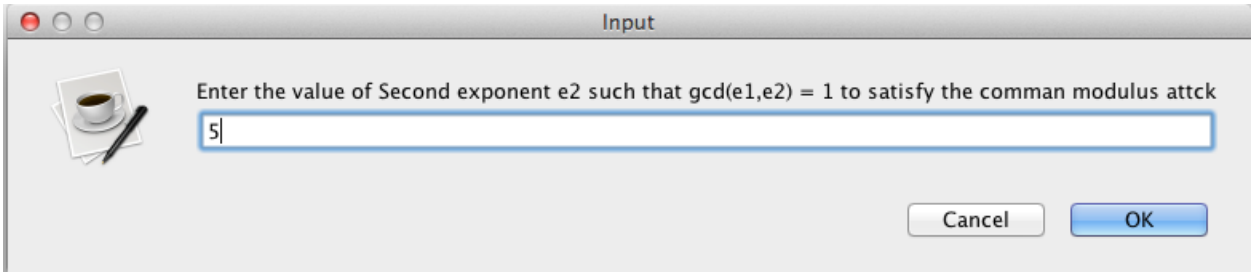
3.3 IMPLEMENTATION

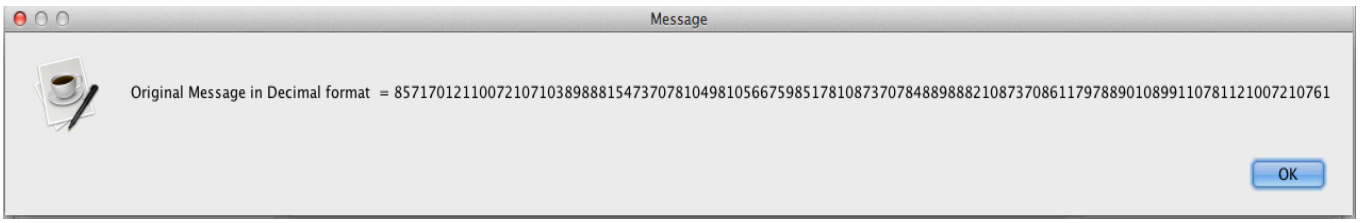
- Programing language Java has been used to implement common modulus attack
- Inputting message M as plain text
- Converting message M to base64 and from base64 to decimal.
- N = 402394248802762560784459411647796431108620322919897426002417858465984510150839043308712123310510922610690378085519407742502585978563438101321191019034005392771936629869360205383247721026151449660543966528254014636648532640397857580791648563954248342700568953634713286153354659774351731627683020456167612375777
- Implemented RSA public key encryption and decryption
- Implemented extended Euclidean theorem.
- Used Big Integer to handle large prime numbers in java.
- Exponents e1 and e2 have been chosen such that e1 and e2 are relatively prime to each other so that gcd(e1,e2) =1
- Program has been tested for many values of M and e1, e2.

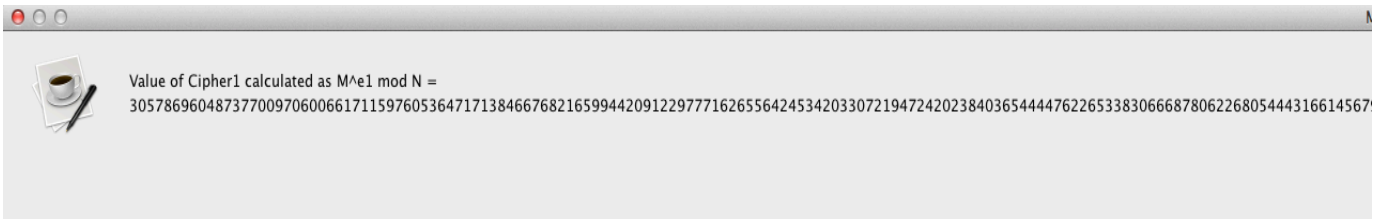
3.4 SCREENSHOTS OF OUTPUTS

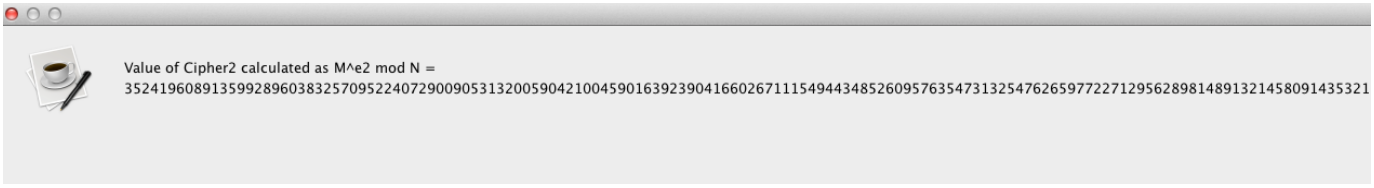




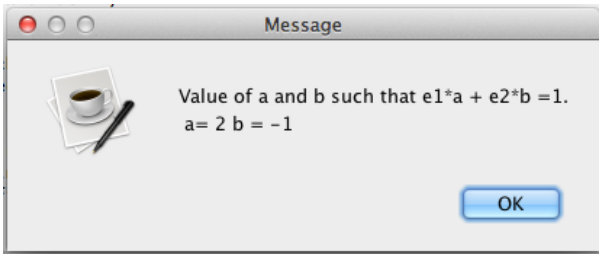


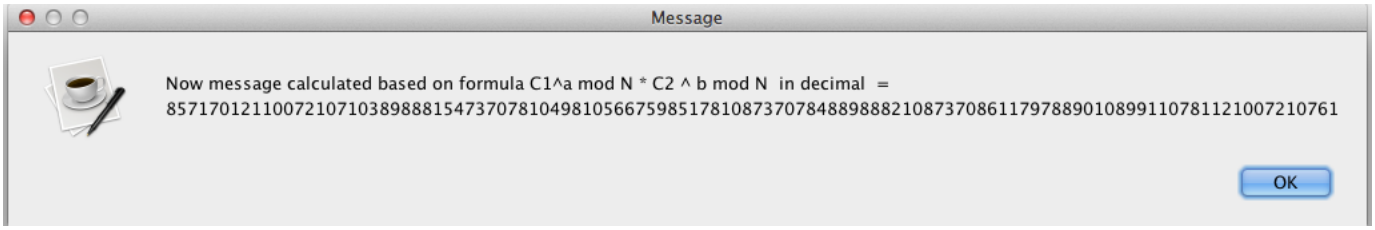



Value of Cipher1 calculated as M^e1 mod N =
3057869604873770097060066171159760536471713846676821659944209122977716265564245342033072194724202384036544447622653383066687806226805444316614567

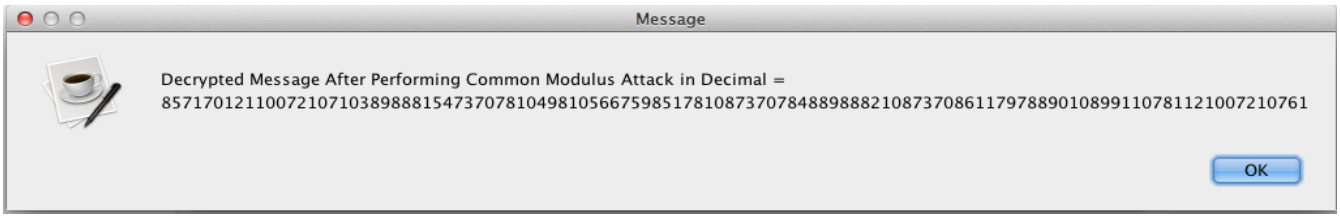

Value of Cipher2 calculated as M^e2 mod N =
352419608913599289603832570952240729009053132005904210045901639239041660267111549443485260957635473132547626597722712956289814891321458091435321


Message
Performing Common Modulus Attack
OK


Message
Value of a and b such that e1*a + e2*b =1.
a= 2 b = -1
OK


Message
Now message calculated based on formula C1^a mod N * C2 ^ b mod N in decimal =
857170121100721071038988815473707810498105667598517810873707848898882108737086117978890108991107811210072107 61
OK


Message
Decrypted Message After Performing Common Modulus Attack in base 64 encoded Format =
UGFydHkgYXQ6IFNhbiBKb3NlIFN0YXRlIFVuaXZlcnNpdHk=
OK




```
<terminated> CommonModulusAttack (1) [Java Application] /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java (May 13, 2014, 12:42:26 AM)
Exponent e1 = 3

Exponent e2 = 5

Original Message in Base 64 encoded format =
UGFydHkgYXQ6IFNhbiBKb3NlIFN0YXRlIFVuaXZlcnNpdHk=

Original Message in Decimal format  = 857170121100721071038988815473707810498105667598517810873707848898882108737086117978890108991107811210072107 61

Value of Cipher1 calculated as M^e1 mod N =
305786960487377009706006617115976053647171384667682165994420912297771626556424534203307219472420238403654444762265338306668780622680544 4316

Value of Cipher2 calculated as M^e2 mod N =
352419608913599289603832570952240729009053132005904210045901639239041660267111549443485260957635473132547626597722712956289814891321458091 4

Performing Common Modulus Attack

Value of a and b such that e1*a + e2*b =1.
 a= 2 b = -1

Now message calculated based on formula C1^a mod N * C2 ^ b mod N  in decimal  =
857170121100721071038988815473707810498105667598517810873707848898882108737086117978890108991107811210072107 61

Decrypted Message After Performing Common Modulus Attack in base 64 encoded Format =
UGFydHkgYXQ6IFNhbiBKb3NlIFN0YXRlIFVuaXZlcnNpdHk=

Decrypted Message After Performing Common Modulus Attack in Decimal =
857170121100721071038988815473707810498105667598517810873707848898882108737086117978890108991107811210072107 61

Orignal Message matches with the decrypted message
Original Message = Party at: San Jose State University
Decrypted Message = Party at: San Jose State University
```

4. ALICE BIRTHDAY PARTY -2

4.1 SOURCE:

- https://www.**mysterytwisterc3**.org/
- Level II Challenge

4.2 PROBLEM SPECIFICATION:

- Alice has learned from last year's mistake and no longer sends encrypted emails to recipients who use the same RSA modulus N.
- This year, she invites her friends Bob, Bertha, and Birte to her birthday party.
- Thus, Alice sends the same message to all three of her friends, encrypted with their respective public RSA keys.
- Is Eve again able to decrypt the cipher texts and thereby find out when and where the party will take place?

4.3 ENCRYPTION METHOD:

- The public keys of Bob, Bertha, and Birte are (N1,e1), (N2,e2) and (N3,e3) and are available for download on the Mystery Twister website.
- The plaintext is encoded with the same method as in the previous year, i.e. applying a base64 encoding and then using the ASCII codes of the characters as a hexadecimal representation of an integer.

4.3.1 GIVEN RSA PARAMETERS:

```
N1=
5147451670252223874341323771370567159547507298071514479298942896955872857938
8909997853690449445586247304569439235361226052858207452171173586408238050587
4261026769465596315849668245703081452047808798727647904141791488099702631575
6921706831026224717983763974406002922250384121766813441662040278427248771626
81931
```

N2=

3324595527999155443560226416054481376170799213918322225578929498080609530284
4942232828141362991233505144074495545501085101230891829454976500548012106169
7711447087615327860789708246235156912421474047484838827777697938563515420810
6503935535280588313174093405771492335542353464458902386429553901374655112864
14033

N3=

6657019121622430690596537816692308054734574277675143232627628917711223523287
0669540910371386438483343743864812021761599076522036574501373924602220359323
4785338178963805463643869398986119431772931646042972240277833431035018628949
9248134635534192431088373090783164555047497550628652580639262436062068065499
69161

e=3

4.4 SOLUTION:

In this scenario, Alice uses the same encryption coefficient, i.e. e=3 and the same message is sent to three recipients using {(e,N1),(e,N2),(e,N3)} where gcd(Ni,Nj) = 1 for i,j = 1; 2; 3 with i != j, and M < Ni. In such cases, M can be recovered very efficiently using the "**Chinese Remainder Theorem**".

4.4.1 CHINESE REMAINDER THEOREM:

This theorem gives solutions to systems of congruencies with relatively prime moduli. The solution to a system of congruencies with relatively prime moduli may be produced using a formula by computing modular inverses, or using an iterative procedure involving successive substitution.

Theorem:

Let m1 ....., mk be integers with gcd(mi; mj)=1 whenever i != j. Let M be the product m = m1m2...mk. Let a1;:::;ak be integers. To compute (A mod M) initially (ai mod mi) are computed individually and then these individual computations are aggregated to produce final result which can be depicted by using the following formula

$$A \equiv \left(\sum_{i=1}^{k} a_i c_i \right) \bmod M$$

$$c_i = M_i \times \left(M_i^{-1} \bmod m_i \right) \quad \text{for } 1 \leq i \leq k$$

4.5 IMPLEMENTATION:

4.5.1 Platform :

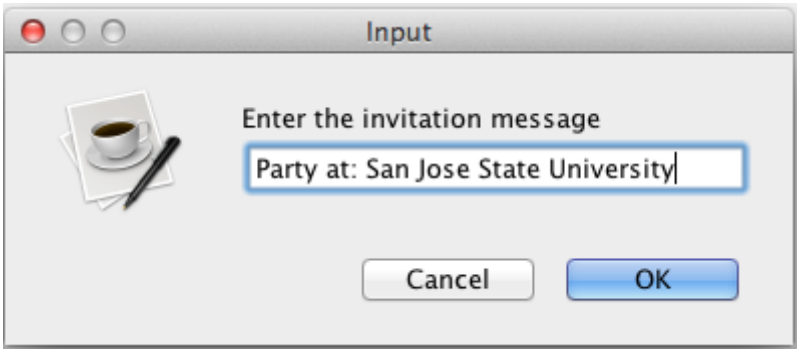
Java is used to implement the project as it is flexible and facilitation of BigIntegers in java enabled proper implementation of the project.

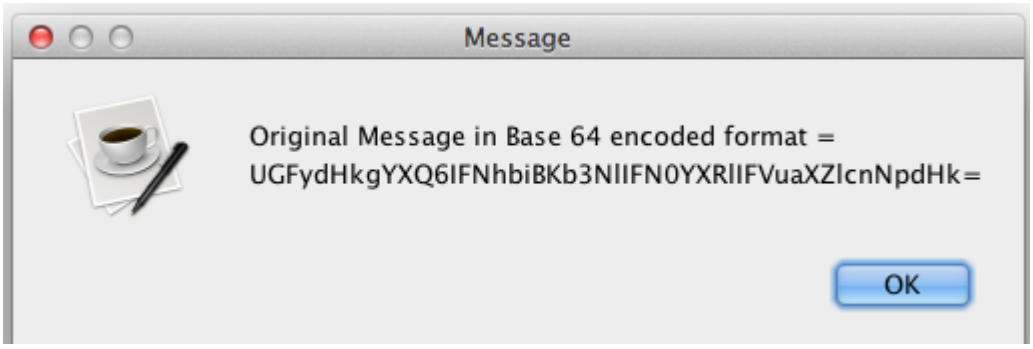
4.5.2 Steps:

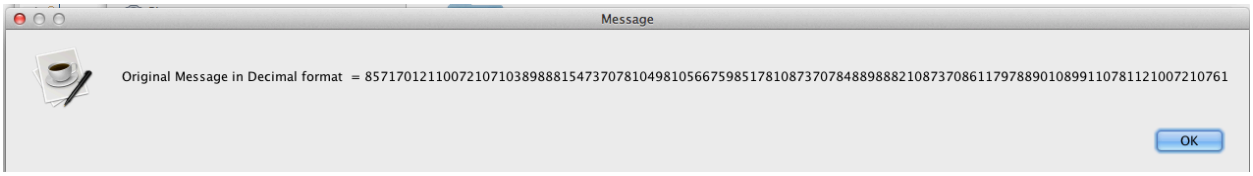
The following steps denote the basic project flow.

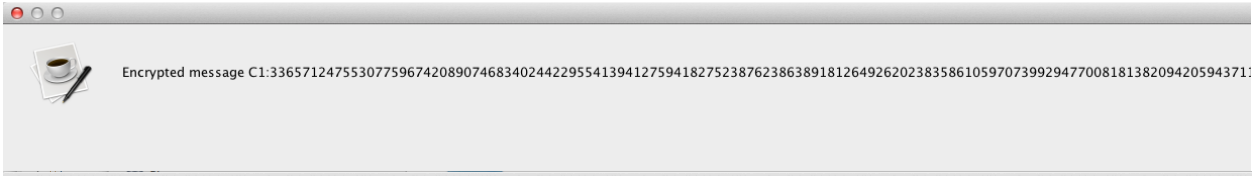
- Inputting message M and is considered as Plaintext.
- Converting M to base64 and from base64 to decimal.
- Encrypting message M with N1, N2, N3 which are provided in input parameters generating C1, C2, C3.
- Chinese Remainder is implemented that can be depicted as follows
  - (M^3)= C1.(N2.N3).((N2N3)-1Mod N1) + C2.(N1.N3).((N1N3)-1Mod N2) + C3.(N1.N2).((N1N2)-1Mod N3) (Mod N1.N2.N3)
- To generate M from (M^3) cube root is evaluated.
- Obtained M which is in decimal format is passed through base64 decoding to generate plaintext message M .
- Program has been tested for different values of M.

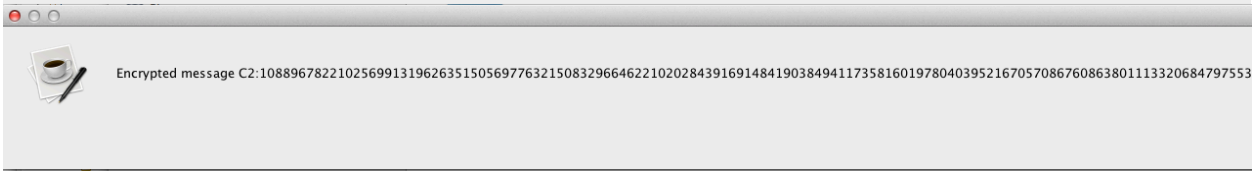
4.6 SCREENSOTS:

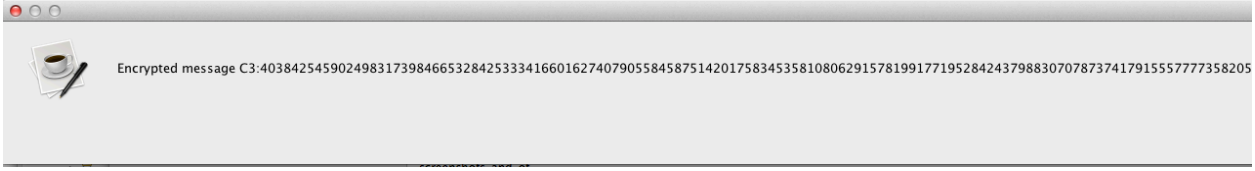


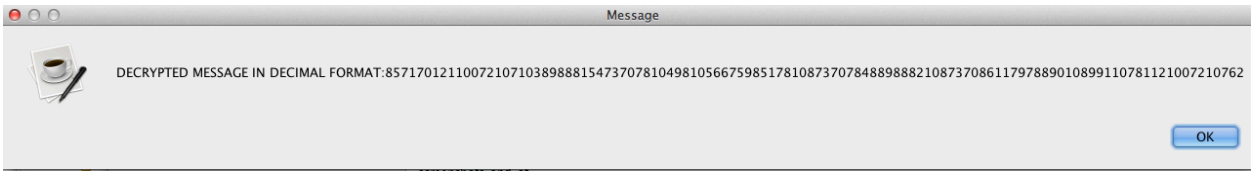


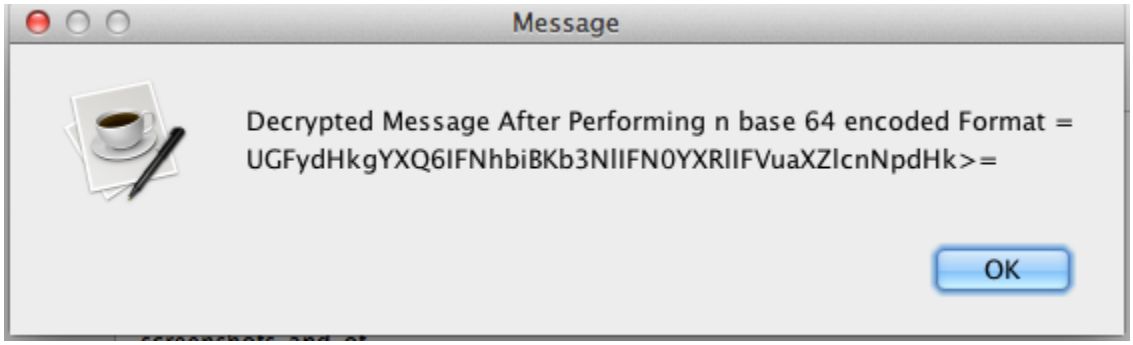


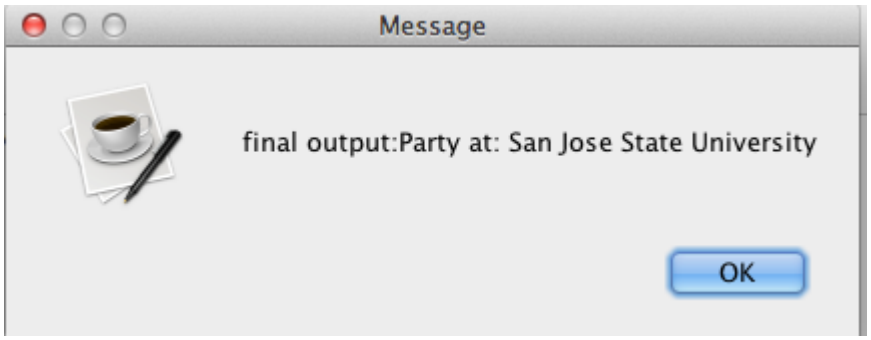












Problems Javadoc Declaration Console

```
<terminated> ChineseRemainderTheorem (1) [Java Application] /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java (May 13, 2014, 9:39:33 AM)
Enter the invitation message M

Original Message in Base 64 encoded format =
UGFydHkgYXQ6IFNhbiBKb3NlIFN0YXRlIFVuaXZlcnNpdHk=

Original Message in Decimal format  = 857170121100721071038988815473707810498105667598517810873707848898882108737086117978890108991107811 21

Encrypted message C1:336571247553077596742089074683402442295541394127594182752387623863891812649262023835861059707399294770081813820942059 4
Encrypted message C2:108896782210256991319626351505697763215083296646221020284391691484190384941173581601978040395216705708676086380111332 0
Encrypted message C3:403842545902498317398466532842533341660162740790558458751420175834535810806291578199177195284243798830707873741791555 7

DECRYPTED MESSAGE IN DECIMAL FORMAT:857170121100721071038988815473707810498105667598517810873707848898882108737086117978890108991107811210 0
Decrypted Message After Performing n base 64 encoded Format =
UGFydHkgYXQ6IFNhbiBKb3NlIFN0YXRlIFVuaXZlcnNpdHk>=

final output:Party at: San Jose State University
```

5. TIMING ATTACK

RSA is a public key cryptosystem which is used for secure data transmission. It uses the public key pair (N,e) and the private key d for decryption. The modulus "N" is the product of prime numbers p and q. The values e and d are chosen so that ed = 1 mod (p-1)(q-1). In RSA the message M is encrypted by the public key pair, C=M^e mod N, while the decryption is by the private key pair, M=C^d mod N. Timing attacks are based on the fact that a few computations in RSA take longer than others. By carefully measuring the amount of time required and analyzing the timing variations it is possible to recover the private key "d" or a few bits of "d".More advanced versions of timing attacks were found to attack OpenSSL in network connection.

5.1 KOCHER'S TIMING ATTACK

Kocher views timing attack as a signal detection problem. It is a form of side channel attack since it doesn't rely on the mathematical part of the cryptosystem. The main objective it to recover the private key "d" one bit at a time. Repeated squaring is the method used for computing modular exponentiation.

The algorithm is given in fig 5.1

Repeated Squaring

```
// Compute y = x^d (mod N),
// where d = d0 d1 d2 ... dn in binary, with d0 = 1
s = x
for i = 1 to n
    s = s^2 (mod N)
    if di == 1 then
        s = s · x (mod N)
    end if
next i
return(s)
```

Fig 5.1

In the algorithm, "x" denotes the cipher text and "s" is a string. Here the private key d consists of bits d0,d1,d2…dn in binary, with d0 as 1.In the algorithm it was found that with the value of di==1, there was an extra computation of s= s *x (mod N). So if an attacker can carefully measure the time taken to complete the operation, it is possible to recover the private key "d".

An attacker can choose random cipher texts Cj, where j=0,1,2..n and measure the corresponding timing T(Cj) with possible values of bits "dk=0" and "dk=1". Let "t~" denotes the time at each iteration. So by comparing the variances of (T(Cj) - t~),for values of dk as 0 or 1, it was found that the correct bit of dk had a lesser variance.

5.2 SAMPLE VALUES

In the program the value of N was chosen as 33, with p as 3 and q as 11.The value "e" was chosen as 3. The private key "d" to be recovered is 7. Throughout the experiment three cipher text values were chosen, 15, 12 and 10. Here it was assumed that the first bit of private key "d" is 1. The next aim was to recover the second bit. We chose the values of d as 4 and 6 to recover the second bit of private key d. The values 4 and 6 were chosen since the second bit of

these values in binary is a one and a zero. So it was possible to find the private key “d” by measuring the time differences for random messages.

5.3 IMPLEMENTATION

The timing attack was implemented in Java. The user gives the message and is encrypted with RSA public key pair (N, e). The cipher text was then decrypted by using the private key “d” of the user. The time for decryption by the user was calculated and stored. In the Java program “System.nanoTime()” was used to compute the time taken for each operation.

In the program three random messages were chosen “15, 12 and 10” and the private key value “d” was converted to binary format. The time of operation taken by the user to decrypt the message was measured. Here the repeated squaring method was used for modular exponentiation. The difference between the calculated timing and the time at each iteration were tabulated for values of “dk” as 0 or 1. The variance was computed and compared with for the values of d as 4 and 6.The below denotes the timing measurements to find out the second bit of private key “d”. The unit of time is nano seconds.

Timings:

		d as 4 , 100			d as 6, 110		
j	**Message**	**T(Cj)**	**t~0,1**	**T(Cj)-t~0,1**	**T(Cj)**	**t~0,1**	**T(Cj)-t~0,1**
0	15	373586	26030	347556	379085	23097	355988
1	12	298429	23464	274965	298062	23464	274598
2	10	247102	23464	223638	276798	28963	247835

Fig 5.2

5.4 CALCULATION

For d=4, 100

Mean timing E(T(Cj)-t~0,1) = (347556+274956+223638)/3 = 282053.0

Variance is var(T(Cj)-t~0,1) = 2.584398326E9

For d=6, 110

Mean timing E(T(Cj)-t~0,1) = (355988+274598+247835)/3 = 292807.0

Variance is var(T(Cj)-t~0,1) = 2.115295742E9

var(d as 6) < var(d as 4)

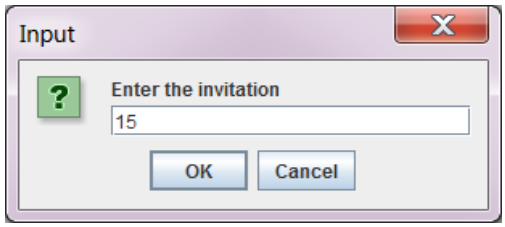
So the second bit is 1.

5.5 RESULTS

Case 1: Selecting value of "d" as 4, 100

a)

N=33
e=3
Trying to recover the value of d which is 7, 111(in binary)
Attacker choosing value of d as 4, 100

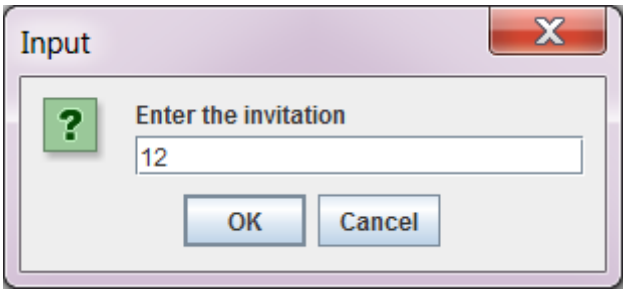


```
The user
Enter the value of N
33
Enter the value of e
3
the cipher text is9
Enter the value of d
7
The value of d in binary is:
1
1
1
The time required for decrypting the ciphertext: 9for the the user is: 373586


Attacker measuring the time
enter the cipher text
9
Enter the value of d
4
The value of d in binary is:
1
0
0
The time of iteration for the 0:th bit is: 26030
The time of iteration for the 1:th bit is: 13565
The time of iteration for the 2:th bit is: 21998
the time array[26030, 13565, 21998, 0, 0, 0, 0, 0, 0, 0, 0, 0]Which bit to check give 1 for bit **1** and 2 for bit **2**

1
[347556, 0, 0]
do u wish to continue??press 1 to continue1
```

b)

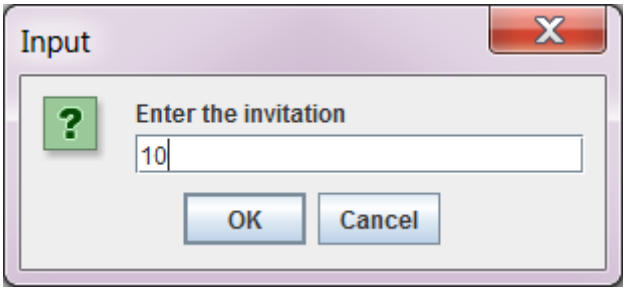


```
do u wish to continue??press 1 to continue1
The user
Enter the value of N
33
Enter the value of e
3
the cipher text is12
Enter the value of d
7
The value of d in binary is:
1
1
1
The time required for decrypting the ciphertext: 12for the the user is: 298429


Attacker measuring the time
enter the cipher text
12
Enter the value of d
4
The value of d in binary is:
1
0
0
The time of iteration for the 0:th bit is: 23464
The time of iteration for the 1:th bit is: 12832
The time of iteration for the 2:th bit is: 26030
the time array[23464, 12832, 26030, 0, 0, 0, 0, 0, 0, 0, 0, 0]Which bit to check give 1 for bit **1** and 2 for bit **2**

1
347556, 274965, 0]
do u wish to continue??press 1 to continue1
The user
```

c)



```
The user
Enter the value of N
33
Enter the value of e
3
the cipher text is10
Enter the value of d
7
The value of d in binary is:
1
1
1
The time required for decrypting the ciphertext: 10for the the user is: 247102


Attacker measuring the time
enter the cipher text
10
Enter the value of d
4
The value of d in binary is:
1
0
0
The time of iteration for the 0:th bit is: 23464
The time of iteration for the 1:th bit is: 13931
The time of iteration for the 2:th bit is: 20164
the time array[23464, 13931, 20164, 0, 0, 0, 0, 0, 0, 0, 0, 0]Which bit to check give 1 for bit **1** and 2 for bit **2**

1
[347556, 274965, 223638]
do u wish to continue??press 1 to continue3
Finished...checking for mean for the variables...
```

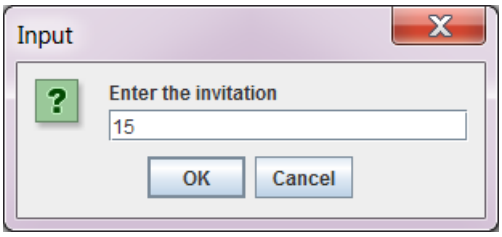
d) Displaying Mean and Variance

```
1
[347556, 274965, 223638]
do u wish to continue??press 1 to continue3
Finished...checking for mean for the variables...
[347556, 274965, 223638]
the mean is 282053.0
the value of sum2 is 7.753194978E9
checking for variance for the variables...
[347556, 274965, 223638]
the variance is: 2.584398326E9
```

Case 2: Selecting value of "d" as 6, 110

a)

N=33
e=3
Trying to recover the value of d which is 7,111(in binary)
Attacker choosing value of d as 6, 110

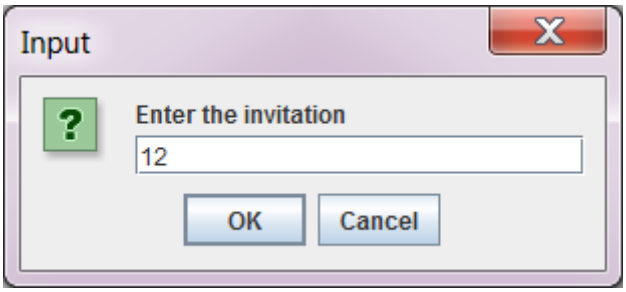



```
The user
Enter the value of N
33
Enter the value of e
3
the cipher text is9
Enter the value of d
7
The value of d in binary is:
1
1
1
The time required for decrypting the ciphertext: 9for the the user is: 379085


Attacker measuring the time
enter the cipher text
9
Enter the value of d
6
The value of d in binary is:
1
1
0
The time of iteration for the 0:th bit is: 23097
The time of iteration for the 1:th bit is: 23097
The time of iteration for the 2:th bit is: 22364
the time array[23097, 23097, 22364, 0, 0, 0, 0, 0, 0, 0, 0]Which bit to check give 1 for bit **1** and 2 for bit **2**

1
[355988, 0, 0]
do u wish to continue??press 1 to continue1
```

b)

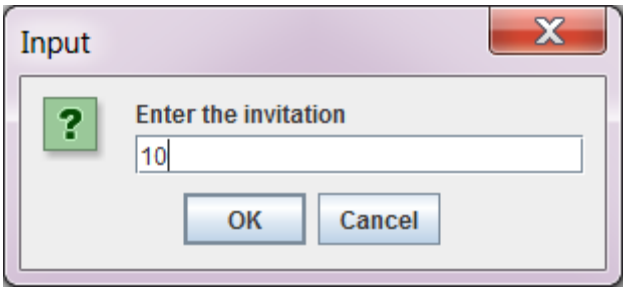



```
The user
Enter the value of N
33
Enter the value of e
3
the cipher text is12
Enter the value of d
7
The value of d in binary is:
1
1
1
The time required for decrypting the ciphertext: 12for the the user is: 298062


Attacker measuring the time
enter the cipher text
12
Enter the value of d
6
The value of d in binary is:
1
1
0
The time of iteration for the 0:th bit is: 23464
The time of iteration for the 1:th bit is: 22730
The time of iteration for the 2:th bit is: 22731
the time array[23464, 22730, 22731, 0, 0, 0, 0, 0, 0, 0, 0, 0]Which bit to check give 1 for bit **1** and 2 for bit **2**

1
[355988, 274598, 0]
do u wish to continue??press 1 to continue1
```

c)



```
The user
Enter the value of N
33
Enter the value of e
3
the cipher text is10
Enter the value of d
7
The value of d in binary is:
1
1
1
The time required for decrypting the ciphertext: 10for the the user is: 276798


Attacker measuring the time
enter the cipher text
10
Enter the value of d
6
The value of d in binary is:
1
1
0
The time of iteration for the 0:th bit is: 28963
The time of iteration for the 1:th bit is: 21264
The time of iteration for the 2:th bit is: 22364
the time array[28963, 21264, 22364, 0, 0, 0, 0, 0, 0, 0, 0, 0]Which bit to check give 1 for bit **1** and 2 for bit **2**

1
[355988, 274598, 247835]
do u wish to continue??press 1 to continue3
```

d) Displaying Mean and Variance

```
do u wish to continue??press 1 to continue3
Finished...checking for mean for the variables...
[355988, 274598, 247835]
the mean is 292807.0
the value of sum2 is 6.345887226E9
checking for variance for the variables...
[355988, 274598, 247835]
the variance is: 2.115295742E9
```

5.6 CONCLUSION AND FUTURE WORK

Timing attacks are applicable for public key systems. Crypto algorithms like RSA are subjected to timing attacks. From the experiments and results provided above it was seen that it was possible to recover the private key “d” bit by bit. The small differences in timings can help to reveal the private key. One possible defense against the RSA timing attack is RSA blinding. Here randomness is incorporated to make timing information impractical. The present system helps in the recovery of private key “d” of three bits. Our future work aims to extend the implementation to recover private key “d” of 8 bits.

6. CONCLUSION

RSA cryptosystem though seems to be secure can be attacked which is exemplified with three scenarios of RSA attacks namely Alice Birthday Problems 1 and 2 from mystery twister and timing attack. Cryptosystem is broken in both the cases of Birthday party problems where in message is encrypted with same modulus in first case and in latter with different modulo values using Extended Euclidean theorem and Chinese Remainder theorem respectively. Timing attack though time consuming can be achieved by measuring the time for an operation to determine private key by implementation of repeated squaring for computing modular exponentiation thus showing that cryptanalysis is as equally achievable as cryptography.

7. REFERENCES

- Timing Attacks on RSA: Revealing Your Secrets through the Fourth Dimension.
- http://cs.sjsu.edu/~austin/cs265-spring14/PublicKeyCrypto.pdf
- http://crypto.stanford.edu/~dabo/papers/RSA-survey.pdf
- http://crypto.stackexchange.com/questions/3549/common-modulus-attack-on-rsa-when-the-2-public-exponents-differ-by-a-single-bit
- Chinese Remainder Theorem.
- http://en.wikipedia.org/wiki/Chinese_remainder_theorem
- http://www.math.tamu.edu/~jon.pitts/courses/2005c/470/supplements/chinese.pdf
- Cube Root Attack.
- www.cs.bilkent.edu.tr/~mustafa.battal/cs470/slides/cs470.RSA.**ppt**