Cryptanalysis of

Broadcasting And Low Exponent - RSA-Attack

By Madhura Kaple and Vigneshwari Chandrasekaran

Madhura Kaple
madhuramukesh.kaple@sjsu.edu

Vigneshwari Chandrasekaran
vigneshwari.chandrasekaran@sjsu.edu

TABLE OF CONTENTS

Madhura Kaple
madhuramukesh.kaple@sjsu.edu

Vigneshwari Chandrasekaran
vigneshwari.chandrasekaran@sjsu.edu

1. INTRODUCTION

RSA is one of the oldest asymmetric public cryptosystem and is named after Rivest, Shamir and Adleman. In RSA cryptosystem a public and private key pair is generated using two large prime numbers p and q. Some of the terms in RSA are,

N= modulus

Also, N=p*q

e= encryption exponent

d= decryption exponent

Thus, public key is given as (N, e) and private key is d.

If M is the message and C is the cipher text then we can give below formulas,

$C = M^{e} \bmod N$

$M = C^{d} \bmod N$

In order to achieve, a high level of security p and q must be relatively large numbers. If these configuration properties are not set correctly security is compromised. In our problem challenge such an attack occurs due to low exponent value and we are able to guess the values for plain text.

2. PROBLEM CHALLENGE:

A circular was encrypted with RSA for three different recipients. You were able to catch the following three cipher texts.

Cipher texts:
C1 :
34d2fc2fa4785e1cdb1c09c9a5db98317d702aaedd2759d96e8938f740bf982e
2a42b904e54dce016575142f1b0ed112cc214fa8378b0d5eebc036dc7df3eeea

C2 :
3ddd68eeff8be9fee7d667c3c0ef21ec0d56cefab0fa10199c933cffbf0924d4
86296c604a447f48b9f30905ee49dd7ceef8fc689a1c4c263c1b3a9505091b00

C3 :
956f7cbf2c9da7563365827aba8c66dc83c9fb77cf7ed0ca225e7d155d2f573d
6bd18e1c18044cb14c59b52d3d1f6c38d8941a1d58942ed7f13a52caccc48154

Madhura Kaple
madhuramukesh.kaple@sjsu.edu

Vigneshwari Chandrasekaran
vigneshwari.chandrasekaran@sjsu.edu

The same message was sent encrypted to three business partners. In addition to the cipher texts above, certificates (see additional file mtc3-brunner-01-rsacrt.zip) are available. These certificates contain the public exponents and the moduli of the three business partner. There is no further information necessary to decrypt the original message because the three business partners are all using the same public key 3.

Objective: Decryption of plain text using Chinese Remainder Theorem.

3. CHINESE REMAINDER THEOREM

The Chinese Remainder Theorem is based on congruence in number theory. It gives a unique solution such that when it is divided by the given divisors leaves given remainders.

Theorem:

Let n_1, n_2 , ... n_m be positive integers and co-prime numbers then we can give a unique solution x such that it satisfies,

$x = a_1 \bmod n_1$
$x = a_2 \bmod n_2$
....
$x = a_k \bmod n_k$

We can use Chinese Remainder Theorem in RSA decryption. Decryption process is relatively faster if we use Chinese Remainder Theorem. This is because it simplifies and speeds up the modular reductions.

Consider that n1, n2 and n3 are three moduli used to encrypt message m and c1, c2 and c3 are the corresponding cipher texts then using Chinese Remainder Theorem,

x = c1 mod n1
x= c2 mod n2
x= c3 mod n3

Now $x= m^3$. Hence, message can be decrypted by taking cube root value of x.
Thus, we decrypted plain text using Chinese Remainder Theorem.

4. ANALYSIS:

We are given following things as input.
a. Public Keys:
(N1, e)
(00:96:23:51:1e:67:69:64:4d:69:3e:89:f6:92:ff:c2:55:8e:ef:12:1d:42:ca:98:69:97:81:e1:39:e2:9c:
2e:1a:a5:8d:88:83:bb:db:a4:11:65:fd:eb:85:a9:a5:64:8f:c2:9a:65:d5:9e:94:01:69:4d:d1:1a:
e2:05:f0:ce:3b,3)
(N2 , e)
(00:ad:4b:c0:f9:80:f4:52:3f:49:0f:c4:0c:12:ef:ce:cc:1e:8a:f6:78:90:b6:56:24:49:87:6e:8e:09:
1e:86:1c:da:69:9e:5a:8e:b3:09:b0:a9:d6:b2:93:10:0c:12:29:fb:d1:8a:59:51:f3:3b:6f:ba:b1:fd:
8d:90:f7:c8:29,3)
(N3 , e)
(00:b7:22:33:64:d8:83:53:ec:02:b0:85:0e:8a:01:d2:ba:9c:a2:66:3c:32:c1:5d:f7:b5:96:40:6c:6f:
c1:c1:71:ac:96:5a:55:4b:8b:33:8f:4b:b0:46:c5:43:93:7b:4b:19:c6:99:86:4f:1d:0d:d4:be:01:77:
ec:cc:e0:bb:57,3)

b. Cipher Texts:
C1 :
34d2fc2fa4785e1cdb1c09c9a5db98317d702aaedd2759d96e8938f740bf982e
2a42b904e54dce016575142f1b0ed112cc214fa8378b0d5eebc036dc7df3eeea

C2 :
3ddd68eeff8be9fee7d667c3c0ef21ec0d56cefab0fa10199c933cffbf0924d4
86296c604a447f48b9f30905ee49dd7ceef8fc689a1c4c263c1b3a9505091b00

C3 :
956f7cbf2c9da7563365827aba8c66dc83c9fb77cf7ed0ca225e7d155d2f573d
6bd18e1c18044cb14c59b52d3d1f6c38d8941a1d58942ed7f13a52caccc48154

From the given information we have to decrypt plain text. We divided this task into two subtasks:
a. Implementing Chinese Remainder Theorem to get plain text in decimal form.
b. Converting decimal to text to obtain the final solution.

a. Implementing Chinese Remainder Theorem:
The given input is in the Hexadecimal string form. To perform mathematical computations, we first converted the hexadecimal numbers to decimal form. Let n1,n2 and n3 be the moduli. Following are the steps to find m using Chinese Remainder Theorem.

1. Convert input moduli and cipher texts from hexadecimal to decimal form.
2. Compute N as N = n1*n2*n3 where n1, n2 and n3 are co-prime.
3. Calculate intermmediate values,
   N1= N/(n1) , N2 = N/(n2) and N3 = N/ n3
4. Calculate the mod Inverse values d1, d2 and d3.
   d1= N1 modInverse n1, d2 = modInverse n2 , d3 = modInverse n3.

Madhura Kaple
madhuramukesh.kaple@sjsu.edu
Vigneshwari Chandrasekaran
vigneshwari.chandrasekaran@sjsu.edu

5. Compute x as x = c1*N1*d1 + c2*N2*d2 + c3*N3*d3
6. The message m is cube root of x.

b. Number to text Conversion:
We obtained the decimal value of message from the Chinese Remainder Theorem. Now we need to convert it into text form.

To convert to text values below steps are performed.

1. Convert the decimal value of plain text to Hexadecimal format.
2. Loop through this hexadecimal string,
    a. group two elements pair wise.
    b. Convert it to ascii format
    c. Print the character value.

Using this approach we obtained the final value which is a German string.

" Das fuer morgen festgelegte Meeting muss unbedingt stattfinden!"

5. CHALLENGES

The main challenge which we faced was implementing the arithmetic operations of the numbers having greater than 150 digits. We represented the values of these numbers using a structure containing array of characters and length of the arithmetic number that is being represented. After implementing the logic for arithmetic operations, the challenge which we faced was that the execution time taken for the arithmetic operations especially multiplication and division was high. In order to solve this we used recursion in addition and subtraction operations. We changed the logic of multiplication and division from repeated add and subtract to a logic similar to how humans compute division and multiplication. Also, to reduce the time of execution in Hexadecimal conversion we maintained a Hex table containing values of powers of sixteen.

6. RESULTS

The implementation of the solution is done in C. Given below are the stepwise results in implementation.

```
n1 in Hex:
009623511e6769644d693e89f692ffc2558eef121d42ca98699781e139e29c2e1aa58d8883bbdba41165f
deb85a9a5648fc29a65d59e9401694dd11ae205f0ce3b

n1 in decimal
7863362828396945422671641651092900868787418304416582734806554598466028883107969839732
075710100920911073968048265197152545404817498266214859796653183913531

n2 in Hex:
00ad4bc0f980f4523f490fc40c12efcecc1e8af67890b6562449876e8e091e861cda699e5a8eb309b0a9d
6b293100c1229fbd18a5951f33b6fbab1fd8d90f7c829

n2 in decimal
9076243440203680321542238609937774679337631521681187228501903278671607488481537902656
962068758672732513386128438533711417528571581253925350724791101278249

n3 in Hex:
00b7223364d88353ec02b0850e8a01d2ba9ca2663c32c15df7b596406c6fc1c171ac965a554b8b338f4bb
046c543937b4b19c699864f1d0dd4be0177eccce0bb57

n3 in decimal
9591484727325841676251343113176121253643898199338756148385302105171888523657925108317
240391030375402041472558339579972248025672763160987601054264115379031

c1 in Hex:
34d2fc2fa4785e1cdb1c09c9a5db98317d702aaedd2759d96e8938f740bf982e2a42b904e54dce0165751
42f1b0ed112cc214fa8378b0d5eebc036dc7df3eeea

c1 in decimal
2766625776468490517020442165200622585353972364734257493634656285428849818088436679743
017768288951285542149396220355412088478729843101404556595060419063530

c2 in Hex:
3ddd68eeff8be9fee7d667c3c0ef21ec0d56cefab0fa10199c933cffbf0924d486296c604a447f48b9f30
905ee49dd7ceef8fc689a1c4c263c1b3a9505091b00

c2 in decimal
3240126800603977412954894850356586499778223497401543135771704391187352751230185581072
740440093233339171219744489365222351865844491714312596605562736155392
```

c3 in Hex:
956f7cbf2c9da7563365827aba8c66dc83c9fb77cf7ed0ca225e7d155d2f573d6bd18e1c18044cb14c59b
52d3d1f6c38d8941a1d58942ed7f13a52caccc48154

c3 in decimal
7826572050150696266209091757688995432053463211631140206664377915185639550299837873119
35279071951212072600213849293562476809725166534434695067385887696520

Value of N
6845423015085344238357264468284182895882058952672668469793978695011625168983736958426
3384319249843207443634213613056435253723420813844857248501483169399686120075877524157
1174860038764756069176223770854665370539563718125052671549866258029282222912406297939
4302022369660196589322980763900172639130265037756334341063294319795253591466025299461
7839225802609062560876393137668352228433890054638861812272996491007159899900245192492
841386859749471889663229136224070478

Value of N1
8705465033820495594988405497215255992982739674834397529272656011435696520384658669092
9925326857704175854908861216884437887397606297170715756686223539029568253925651847194
1635597006128237780308915152687886375840801749323831685399344685433273004713474698752
074007023027966158120787003223172719568466432309967

Value of N2
7542132447399103523998497361560115603578256209193123382603551034883574032276155630844
5346919312192995474080448384435792810560362461644500707887294796791513885554146812175
5127029244246844678065173031391909450994232133303376807749189529482842328230904676256
791722293659714179736027154602966397162491919945684

Value of N3
7136979528917923387785269149837827893074746295801717329022416923729081744116971321586
4684199327224651252235417067626231056097127118068194455885964129593966783607552487519
5834203699120827251328360810624583334831522805363711845794960105348820048552413274584
563141921662127411324599260216447182525740583870872

Value of d1
7090969898858236864724548738600976854488081964380229248864088238758224285207325028718
5153519486818370313181027115380663947397757837366299343025811765671

Value of d2
1994819837169253542592874533791981884828915847538265234452537502512860881268851075780
1906086583232573359944754641942788133151543078358445199586613985619

Value of d3
8425563521240802743481332293039310140218255117090296103626165098238394695561047124555
6027051406228696597422297516907170840020008788433356905541304392978

Value of prod1

```
1707843362523390818889848472454530364346761649079249459699078131537453703850162111829
0553456141936752460312300477232392683946749173389906793264338654926796039046532458831
5881729874660595078533761422812277085248608668633115242543792267406056452354719866866
2431941738837910608248513122461086023025739213523352357293927352237176057614473895127
2038787234542050738736802405888704435157442392277045255816775667695925347734088848200
9225980988996292185037201400889077812518215604333200155534890104840451275995276009272
9251899618329060260174327036010141955017633155402295159554017891215840690983966130421
118786381080582890680
```

```
Value of prod2
4874834090270627386341507799512708097826862976309342277928786007806459365211898955390
2538485749189386031315332547343755591230758454233898974404745819066544857557559813471
4726253606445168493812977894510879256934828490623650329558212686903121977492457158439
2619695928453911359198066212522547519398304130827964542262640098451646074506654615326
6077074135259252284101064084915529528730770664233359755749512925709811558909155165538
5637942832863824852831986973848669952022961715697455979969418409145971275626516446958
5135642135025800910900580194819631751991718814820320195652300156354571208802178064883
68867857447960141824
```

```
Value of prod3
4706358391593007570395630193775423324844341411428934213623429274952647772604273740793
7180266072077521133678703884450587062974697531607126437128106647041296338005274721567
8547834243520500271014797803633240428474453496495778519137186478250888678936140174433
8184690196473586790834057980092551872127564275642043410561337964947457822813156177304
4172519472835652699609683437637974293207549323812003124205163940118002827470458133559
9688117074697963202194380419942166956103628380590080182907946610111654607105957976200
6656204706894381875834401084118142198561322436313206754899989935108189820354502606577
535820813543893615404
```

```
Value of m cube
2738000157236934814338093005442215614318048293376028520113136927658904457235256235999
1444352617412433062971760662580475962048334685399998562459425255534710795445833709496
9630430789440703332497430915386597524152534278560650614356202924519887457431540112407
7278635141765879918716191762745311605103221009069103055815915673831288029705922863660
8197688341715607133715294036968153846444520789086607600295865689792450010525842590071
602371577777625104859083136 97
```

```
Plain Text Message in Decimal
1398978873974242727227719301545390025011151258379320100966823441557620404867276058115
699969070162492669492455054732425439211713417883622854512543304451 3
```

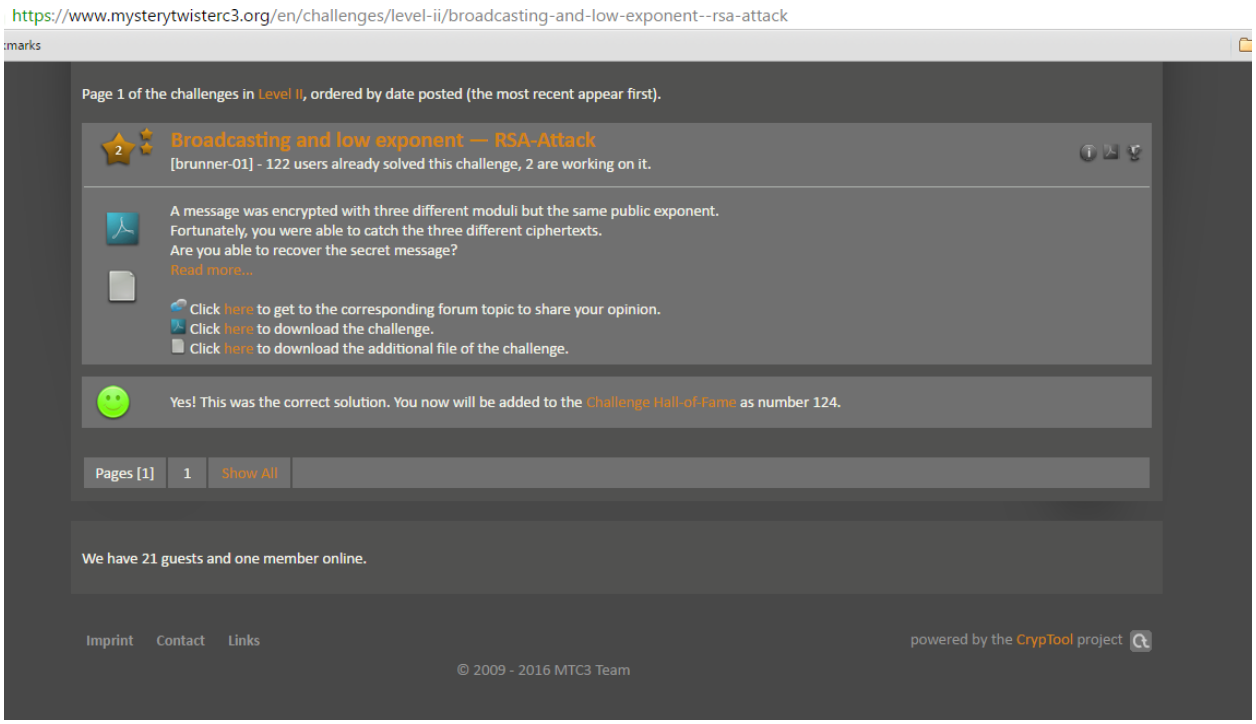
```
THE DECRYPTED PALIN TEXT IS :
"Das fuer morgen festgelegte Meeting muss unbedingt stattfinden!"
```

Thus, the final solution obtained for the challenge is :
"Das fuer morgen festgelegte Meeting muss unbedingt stattfinden!"

We submitted the solution on the website which confirmed that the solution is correct.

Madhura Kaple
madhuramukesh.kaple@sjsu.edu

Vigneshwari Chandrasekaran
vigneshwari.chandrasekaran@sjsu.edu



7. PREVENTIVE MEASURES:

In order to prevent this attack the large values of p and q must be chosen. Large values will make the mathematical computations difficult. Hence, it will be more secure. Also, another way to protect the attack is by salting of message m. Salting refers to adding random bits to the message. These random bits must be different for every message.

8. REFRENCES:

[1] https://www.mysterytwisterc3.org/images/challenges/mtc3-brunner-01-rsacrt-en.pdf

[2] http://www.di-mgt.com.au/crt.html

[3] https://en.wikipedia.org/wiki/Chinese_remainder_theorem

Madhura Kaple
madhuramukesh.kaple@sjsu.edu

Vigneshwari Chandrasekaran
vigneshwari.chandrasekaran@sjsu.edu