The importance of enough random padding bytes in RSA when Mallory has access to the ciphertext of two or more related messages

The CA teachers

October 28, 2022

1 Introduction

To maximize the amount of space for a text message, Alice had the very dubious idea of requiring that all text messages sent to her be padded using only one random byte. The format of the plaintext messages sent to her is

```
 | 0 | random_byte | padding_zeros | text_message |    --- the bytes
  n-1      n-2        n-3          k   k-1        0     --- the byte numbers
  MSB                                           LSB
```

where

- `n` is the number of bytes of her public modulus, i.e., its number of bits divided by 8 (rounded upwards),
- `random_byte` is a single byte with a random value (0 to 255),
- `padding_zeros` are one or more bytes with value zero, and
- `text_message` is a UTF-8 encoded text message (the first character of the text message is stored in the least significant byte of the message, see example below). The text message, without the NUL C-style string terminator, is stored in `k` bytes.

2 Alice's public data

Alice's 2048-bit public modulus is

$$
\begin{aligned}
n =\,&322565848538816688196432267494834625445172807861961952555826780741255228413909\\
&565407829620144084315659098775442457181409009228314141139330943826858041810287\\
&156287891738485415520908563819595478001093462509180230774113164867790592099055\\
&021421494634969156754251717799408021412368025391004464458649484690099297454249\\
&961612370685073608213864469821870612710050353962139785366616150009167075971394\\
&191063112097643885160238788274258938864097016532774176469790006966196152841359\\
&394464677416793143495812340592927067214756828744957792258547598849849659332782\\
&49977898357852295765822796204816960068087631764580573733332129094366877
\end{aligned}
$$

and her public exponent is

$$e = 65537 = 2^{16} + 1.$$

3 The mission

Bob sent Alice the following ciphertext (using Alice's public modulus and exponent)

$$
\begin{aligned}
c_1 =\,&1399970625549014455421308400736210030945662558362953572598382444074245011451 00\\
&2192362849938580065576870469751387389273086206217906450301744222857785063104 81\\
&8003557444229393404173050387741532623600586655046171463075044697765722226994 57\\
&0144187331662652782035424087087418856432695472090608317855177421617414712003 76\\
&1481102837113900141860890235282302483616444962916326328759432975306029490358 30\\
&2629740992800354180841301004276530602003470566643500357632172084094528868862 30\\
&1495303453451552190565880607255925026878441544105927652895555098514816004352 74\\
&00785300898790453936675517505792768625755579501439957445247311207034994.
\end{aligned}
$$

That was a message Mallory was interested in, so he intercepted it and forced Bob to resend the message again (but with a different padding byte). The ciphertext of the second message was

$$
\begin{aligned}
c_2 =\,&7330646540408462499487732988734435881882600143917250125234725645295265436844 85\\
&5956345867015460257851867152226082992604583380722353979123823769153663750127 37\\
&8381181947124303124717022628317971445602980523815915703319718627892152151763 22\\
&4999403696019840134053621522434775044673990084838715747559195164385146719276 03\\
&5186448119274230106962051610865462775732255984526858086851588534412481383870 18\\
&3265812676914984402719899236047964431291778087348829328052439726843671068196 57\\
&5220476292945021334692065439186080314715245574910218285041885631078655049765 02\\
&91012236511723360867268166045860510741130298992826697432917919005743 76.
\end{aligned}
$$

Your mission, should you choose to accept it[1], is to decipher the message.

4 A complete smaller example

Before tackling the true mission, develop code that solves the following smaller example. Although we provide here all information for this smaller example (including Alice's private data and the actual padding bytes and original plaintext message), recover the plaintext using only n, e, c_1, and c_2.

Private data:

$$p = 91385177868893188439606205446657433154550775543347736767839078918307136071207$$

and

$$q = 126919665154092992838405484421396465545153294878449134175022513263373469020623.$$

Public data:

$$
\begin{aligned}
n =\,&11598576175167152956424461782394541439955617267494049729274868685493179955542 9\\
&4917981073047476525570384963660073212215539247436959605944225206667427950196 1
\end{aligned}
$$

and

$$e = 17.$$

[1]Shameless plagiarism of the Mission: Impossible unforgettable quote from the shows and films. As in the shows and films, this mission is not impossible, but it is not trivial to do `:-)`

The plaintext message is "Test", which becomes (in ASCII and UTF-8, 'T' is 84, 'e' is 101, 's' is 115 and 't' is 116)

$$m = 84 + 256 \times 101 + 256^2 \times 115 + 256^3 \times 116 = 1953719636.$$

The padding random bytes of the two messages are 133 and 147, and so the properly padded messages to be encrypted are, respectively,

$$\begin{aligned} m_1 =& m + 256^{62} \times 133 \\ =& 272100594281366792944846393548794179528008370077329760625798594972991121826138 \\ & 57237467182642459078306316745561810343090264118445037672329755795339306324 \end{aligned}$$

(times 256^{62} because the modulus has 512 bits, i.e., 64 bytes), and

$$\begin{aligned} m_2 =& m + 256^{62} \times 147 \\ =& 300742762100458034307461803396035672109903987980206577533777394443832292544679 \\ & 47472990043973244244443823771410421958152397183544515322048677457800947028. \end{aligned}$$

The corresponding encrypted messages are

$$\begin{aligned} c_1 =& 456895437139824131278804667966197337020713629180647352821881395579670182426998 \\ & 81750940143160979352750713318253423828978300269287068062832615768482159380 45 \end{aligned}$$

and

$$\begin{aligned} c_2 =& 107747774400912358212692904047235026033556883693845055626932654797224397687121 \\ & 01419248994746280675715013811585517386981365362526469147816190061167310265864. \end{aligned}$$

Try the true mission only when you are confident you got everything right. The solution method used by one of the teachers (hint: Coppersmith, Franklin, Patarin, and Reiter), coded in `pari-gp`, required about 1 GiB of memory and took about 4 hours to finish on a relatively old laptop (2015). The smaller example, mainly due to the smaller encryption exponent, is much easier (less than one second to solve).

To avoid copy and paste errors, all data is also provided in two text files: `mission.txt` and `training.txt`.