

# O7R: A Cryptography Challenge for CryptoCTF 2024

factoreal<sup>1\*</sup>

rooney<sup>2</sup>

<sup>1</sup>ASIS

factoreal@asis.sh

<sup>2</sup>EURECOM

solmaz.salimi@eurecom.fr

## Abstract

There is an available oracle that, for any input consisting of a string of numbers, converts each digit into a 7-segment representation and then randomly corrupts it with a probability of one-half. The corruption is done by randomly removing one of the segments in the given digit. The parameters and outputs of the RSA algorithm are provided to this oracle, and some resulting outputs are obtained. In this challenge, your task is to find the flag value using any of the provided parameters.

**Keywords:** RSA, 7-segment, Recovery

## 1 Introduction

The RSA partial key recovery algorithm is a cryptographic method used to potentially retrieve parts of the private key in the RSA encryption scheme. This algorithm takes advantage of vulnerabilities or weaknesses in the RSA implementation or key generation process. By analyzing the available information such as ciphertext and corresponding plaintext pairs, or the public key and its parameters, it attempts to deduce specific components of the private key, such as the prime factors of the modulus. With these partial key components, an attacker may be able to perform further calculations to fully reconstruct the private key and gain unauthorized access to encrypted data. The RSA partial key recovery algorithm underscores the importance of secure key generation and proper implementation practices to prevent potential attacks.

In this challenge, there is an oracle that generates RSA private keys with a public key modulus of 1024 bits. However, there is a flaw in the oracle's process, resulting in the creation of corrupted values for the prime factors  $p$  and  $q$  and the modulus,  $n$ , of the private key. Despite this flaw, we have access to the exact ciphertext,  $c$ , and the encrypted message, which have not been corrupted.

Given this situation, our objective is to retrieve the secret message by leveraging the available information. To achieve this, we can employ various techniques such as factorization algorithms or mathematical methods specifically designed to recover RSA private keys from partial or incorrect information. By analyzing the corrupted values of  $p$ ,  $q$ , and  $n$  and applying appropriate mathematical computations, we can deduce the correct values of  $p$  and  $q$ . With the correct values of  $p$  and  $q$ , we can then calculate the private key and use it to decrypt the ciphertext, revealing the secret message.

### 1.1 Preliminaries

The seven-segment presentation of numbers is a commonly used method to display numerical digits using a combination of seven individually controllable segments. These segments are typically arranged in a pattern resembling the number 8, with a horizontal bar at the top, bottom, and middle, and vertical bars on the left and right sides, as well as diagonal bars forming a V shape in the middle. By selectively illuminating or turning off these segments, different numerical digits (0-9) can be visually represented. The seven-segment display is widely used in various applications such as digital clocks, calculators, electronic signs, and other devices where numerical information needs to be communicated visually.

For example, the number 1337 can be represented in seven-segment format as shown in the following picture 1.1.

---

\*The authors were not supported by any grants; they did it out of love 💕💕



Figure 1: The seven-segment representation of 1337

The RSA algorithm is a widely used asymmetric encryption algorithm that relies on the mathematical properties of prime numbers. It involves the following steps:

1. Key Generation:

- Choose two distinct prime numbers,  $p$  and  $q$ .
- Calculate the modulus  $n$  by multiplying  $p$  and  $q$ .
- Compute Euler's totient function  $\varphi$  using the formula:  $\varphi(n) = (p-1)(q-1)$ .
- Select a public exponent  $e$  such that  $1 < e < \varphi(n)$  and  $\gcd(e, \varphi(n)) = 1$ .
- Calculate the private exponent  $d$  using the formula:  $d \equiv \varphi(n)^{-1} e^{-1}$ .

2. Encryption:

- To encrypt a message  $m$ , convert it to a numerical representation.
- Apply the encryption formula:  $c \equiv m^e \pmod{n}$ .

3. Decryption:

- To decrypt the ciphertext  $c$ , apply the decryption formula:  $m \equiv c^d \pmod{n}$ .

The security of RSA relies on the difficulty of factoring large numbers into their prime factors. The public key (consisting of the modulus  $n$  and the public exponent  $e$ ) is used to encrypt messages, while the private key (consisting of the modulus  $n$  and the private exponent  $d$ ) is used to decrypt the ciphertext and recover the original message. In summary, RSA provides a secure method for encryption and decryption by using a pair of mathematically related keys. The public key can be freely shared, while the private key must be kept secret.

An oracle is available that takes any digital input, transforms each digit into a 7-segment representation, and then introduces random corruption with a 50% probability. The corruption involves randomly removing one segment from each digit.

## 1.2 Findings!

The oracle outputs the following RSA parameters along with the encrypted message  $c$ . Please note that the secret message is converted to an integer using the `bytes.to_long` function in Python.

```
p = 7326E7397677567711U3G '59029527757860037207
    R9625435 18 10 1139E95206035595235J06 183226
    5 186EE3234 195A328570482543U 1 1638027G 7263
    406729736 ' 727952A4 12764825' 18 14597
```

```
q = 70756253743433 12393650 1362111 18772 15275 ' ,
    18' 164 '023' 123 19235552342904438 18 10630842G
    0263 '5 127327G0666 155533A2779566242579960
    7464237 16255703946346 1940977806702
```

$n$  = 5 1840E57766 1552E755570E7811 ' 0506546 1.867  
 60857080040 1343500297022092E2552A38 '6879  
 82 050426 ' 1544265 18002947 43965437879769  
 0236739578325A8 1000264555225744 ' 14758 135  
 877 '8595'100 185376924705067 74E '8E2049925  
 157063 17E604A54 12736578963'1002888 134 140A  
 8E'1905535002609555 1 '6780 1673829 15457595 1  
 E35 12 '942478240 17'1076373 1273

$n^2$  = 2697'1532829007 13E630E8957A258 19568803'175  
 377936A3052746656 1 '25779374500 160 136 1 10  
 59 1400340462 1E 18247268E38E046943374A ' 145  
 7544A4 18339E53280E045 '22807494900 10663964  
 9754268 196509005'184752446488 ' 10080966844  
 7E47352 '11 '687009627827 '06E '582560939'1630  
 40023'14A280E203'128 17A237 '3728655 0684385  
 75726377 '8 1 104 12 1 '080 '959335005350625'154  
 563E5454629953E750 226465080502 742 1 '14 1  
 596394702280 195528 1E64535602 104282083430  
 36 16497980000746879 15227080 '577724690040  
 888 1076E7 2580693 135368244 1295364 1 107559  
 3302 17674228564 1567 04 1627'1332 1005033652  
 5 14948974785 1086787952 144879 1024404 17277  
 65A534830025929400527A36A '53348090089377  
 5752272634700523

$c =$  356266379 1006368034 14 1977288433903435480  
72975425624327 17245282035804727709930944  
862803683 105436 1338225674352589 180744398  
366 145966524390 15554048 10 1795255948 17342  
1 1594 100 175954740770365378 142205787 13473  
566032030757092759793436802892 1066574 182  
499247623829999744420 194857583 189 192229 1  
3 180080089822 15564064059476