O7R: A Cryptography Challenge for CryptoCTF 2024

factoreal[1*] rooney[2]

[1]ASIS
factoreal@asis.sh
[2]EUROCOM
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 ♡

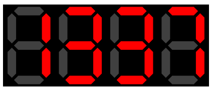


Figure 1: The seven-segment representation of 1337

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 φ 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: $d \overset{\varphi(n)}{\equiv} e^{-1}$.
2. Encryption:
   - To encrypt a message m, convert it to a numerical representation.
   - Apply the encryption formula: $c \overset{n}{\equiv} m^e$.
3. Decryption:
   - To decrypt the ciphertext c, apply the decryption formula: $m \overset{n}{\equiv} c^d$.

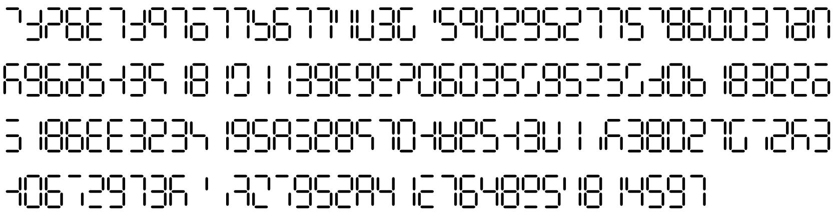
The security of RSA relies on the difficulty of factoring large numbers into their prime factors. The public key (consisting of the modulus n and the public exponent e) is used to encrypt messages, while the private key (consisting of the modulus n and the private exponent d) is used to decrypt the ciphertext 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.

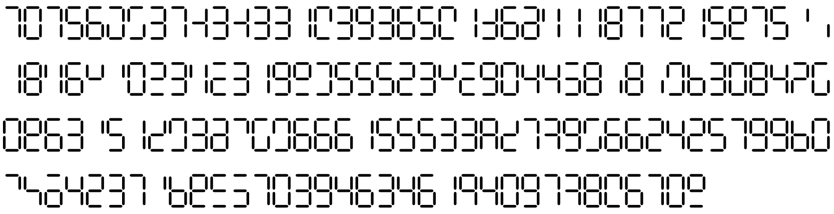
1.2 Findings!

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.

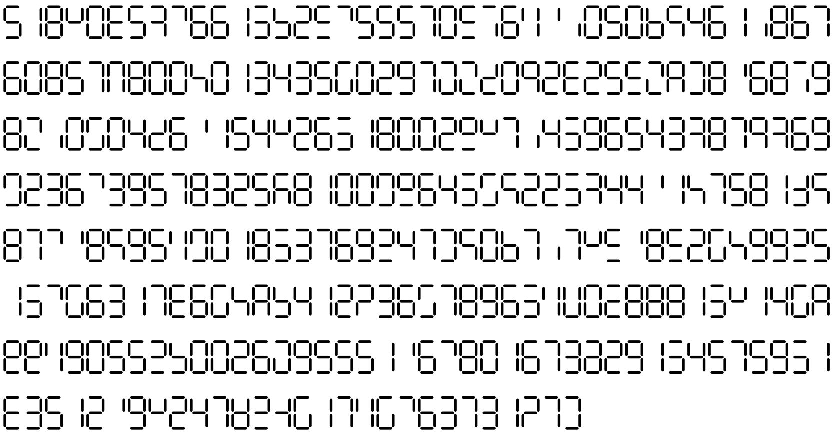
p =



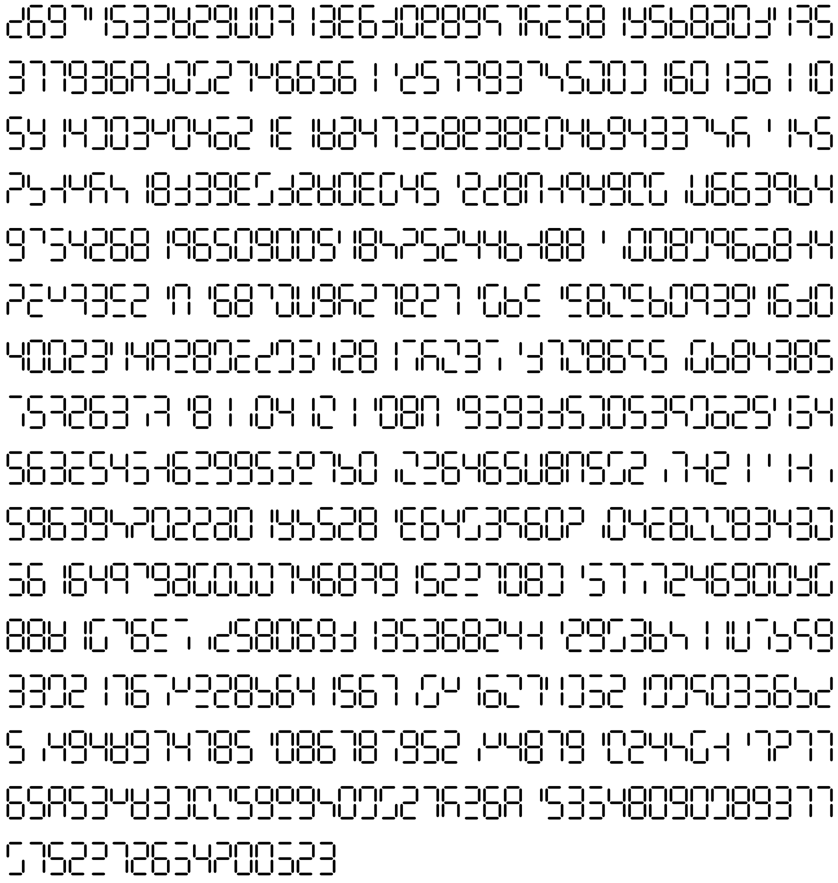
q =



$n =$



$n^2 =$



$$\begin{aligned} c = {} & 3562663791006368034141977288433903435480 \\ & 7297542562432717245282035804727709930944 \\ & 8628036831054361338225674352589180744398 \\ & 3661459665243901555404810179525594817342 \\ & 1159410017595474077036537814220578713473 \\ & 5660320307570927597934368028921066574182 \\ & 4992476238299997444201948575831891922291 \\ & 318008008982215564064059476 \end{aligned}$$