Writeup:

Option1: Ciphertext only , CVE-2017-15361 RSA ROCA attack, vulnerability comes from RSALib, used by BitLocker with TPM 1.2, YubiKey 4 (before 4.3.5) PGP key generation and so on.

$$N = \overbrace{(k * M + 65537^a \bmod M)}^{p} * \overbrace{(l * M + 65537^b \bmod M)}^{q},$$

for $a, b, k, l \in \mathbb{Z}$. The previous identity implies

$$N \equiv 65537^{a+b} \equiv 65537^c \bmod M,$$

Input : N, M', m, t
Output: p – factor of N
$c' \leftarrow \log_{65537} N \bmod M'$ ▷ Use Pohlig–Hellman alg;
$ord' \leftarrow ord_{M'}(65537)$ ▷ See Section 2.6 for method;
forall $a' \in \left[\frac{c'}{2}, \frac{c'+ord'}{2}\right]$ **do**
  $f(x) \leftarrow x + (M'^{-1} \bmod N) * (65537^{a'} \bmod M') \pmod N$;
  $(\beta, X) \leftarrow (0.5, 2 * N^{\beta}/M')$ ▷ Setting parameters;
  $k' \leftarrow Coppersmith(f(x), N, \beta, m, t, X)$;
  $p \leftarrow k' * M' + (65537^{a'} \bmod M')$ ▷ Candidate for a factor;
  if $N \bmod p = 0$ **then**
    return p
  end
end

Reference:
https://acmccs.github.io/papers/p1631-nemecA.pdf
https://crypto.stackexchange.com/questions/53906/how-does-the-roca-attack-work

Option2: Building Carmichael numbers with a large number of prime factors to bypass Fermat primality test in a high possibility, then factor N by the leaked Z using Coppersmith attack.

3.1 The idea

The idea of [23, 40, 14, 1] is to search for Carmichael numbers C with a fixed value of $\Lambda = \lambda(C)$. Let

$$S(\Lambda) = \{p, p \text{ prime}, p \nmid \Lambda, p - 1 \mid \Lambda\}.$$

It is clear that a squarefree product N of elements of $S(\Lambda)$ satisfies

$$\lambda(N) \mid \Lambda$$

as well as property (2). Suppose one wants to find Carmichael numbers with r factors built up with the primes of $S := S(\Lambda)$. It is enough to look for r distinct elements $p_1, \ldots, p_r$ of S such that

$$C := p_1 \times \cdots \times p_r \equiv 1 \bmod \Lambda.$$

If this is the case, we have $C \equiv 1 \bmod \lambda(C)$ since $\lambda(C) \mid \Lambda$.

3.2 An informal description of the algorithm

Let $t = \text{Card}(S)$ and

$$P(S) = \prod_{p \in S} p.$$

Suppose one is looking for a Carmichael number with r prime factors, $r < t$ and $u = t - r$ small. One looks for u primes of S, say $\Theta = \{\theta_1, \ldots, \theta_u\}$ such that

$$\Pi(\Theta) := P(S)/(\theta_1 \cdots \theta_u) \equiv 1 \bmod \Lambda.$$

We examine all u-tuples of primes θ_i in S in order to find a good one. Let us describe the program.

procedure FindCarmichael$(S(\Lambda), r)$
(* one wants r-factor numbers *)
1. Set $t = \mathrm{Card}(S)$, $u = t - r$.
2. Compute $P_\Lambda = \prod_{p \in S} p \bmod \Lambda$.
3. For all u-tuples of distinct primes $\theta_1, \ldots, \theta_u$, check whether

$$\theta_1 \times \theta_2 \times \cdots \times \theta_u \equiv P_\Lambda \bmod \Lambda. \tag{3}$$

Löh and Niebuhr used this idea to find a 81488-digit number with 10058 prime factors.

Reference:
https://hal.inria.fr/inria-00076980/document
https://acmccs.github.io/papers/p1631-nemecA.pdf
https://github.com/mimoo/RSA-and-LLL-attacks

Option3: choose plaintext attack, we are given an encrypt oracle with no padding, so just encrypt 2,4,8…to leak N1, then noticed that p and q are xored with two special primes, and we know its product N2, we can assume:

$N3 = a * b * 10^{1000} + a^2 * 10^{500} + b^2 * 10^{500} + a * b$

where a and b are both $< 2^{500}$, noticed that:

$N3 \equiv a * b (\mathrm{mod}\ 10^{500})$

$$\left\lfloor \frac{N3}{10^{1500}} \right\rfloor = \frac{a * b}{10^{500}} + 1$$

So we have:

$n - 10^{1000} * a * b - a * b = 10^{500} * (a^2 + b^2)$

Solve this quadratic equation of a^2 to factor N2, then we can get the two special primes and then get p and q, once we get p, noticed that p-1 is smooth, so we can factor p-1 to get these small primes, then solve DLP for each small prime and finally to reconstruct e using CRT. Once we get e, we get everything we need, we can easily calculate d and decrypt the ciphertext.

Reference:
https://crypto.stackexchange.com/questions/68916/rsa-find-public-exponent

Option4: choose ciphertext attack, we are given an decrypt oracle which will leak the LSB of the corresponding plaintext, just use RSA LSB attack to recover the shuffled msg, then use Z3 to implement constraint solving method to get the original msg, cuz the same plaintext prefix will got the same ciphertext prefix, we only need to recover a small part of the plaintext, then we can get the full plaintext cuz it is not meaningless.

Option5: We need to input 64 numbers, then our inputs will first do the reverse of DCT, then do the FFT, after the two transformation we will get a number, this number will then xor with msg1, msg2, msg3 and msg4 one by one, if the final result is equal to the hardcoded number, then the server will ask us to input the xor result of 3 random number, which we can get from option 2, 3 and 4, if this random result is also correct, the server will print the flag at last. Noticed that both DCT and FFT is reversible, so this means the whole process is reversible.