# CyberTek Crypto Challenges Writeup

## A Detailed Analysis of Cryptographic Puzzles

**Author: MRx0rd**

May 5, 2025

Prepared for CyberTek CTF

# Contents

# 1 Introduction

Welcome to an exciting journey through six CyberTek cryptography challenges: **ezRSA+**, **syb3lik**, **ezRSA**, **hash101**, **ezMATH**, and **QUANTUM-BB84**. These puzzles test skills in RSA, elliptic curves, complex number cryptography, and quantum key distribution. This writeup provides clear explanations, mathematical derivations, and code snippets to solve each challenge.

# 2 Challenge 1: ezRSA+

## 2.1 Overview

The **ezRSA+** challenge is an RSA-based puzzle with a non-standard setup. We are given:

- **Modulus** $n$: A 2048-bit product of primes $p$ and $q$.
- **Gift**: lcm$(p-1, q-1)$, the least common multiple of $p-1$ and $q-1$.
- **Ciphertext** $c$: The encrypted flag.
- **Public Exponent** $e = 54722$: An even number.

The goal is to decrypt $c$ to recover the flag.

## 2.2 Solution

The `gift` is lcm$(p-1, q-1)$. The RSA totient is:

$$\phi = (p-1)(q-1). \tag{1}$$

LCM and GCD are related:

$$\text{lcm}(a, b) = \frac{a \cdot b}{\text{gcd}(a, b)}. \tag{2}$$

Thus:

$$\text{gift} = \text{lcm}(p-1, q-1) = \frac{(p-1)(q-1)}{\text{gcd}(p-1, q-1)} = \frac{\phi}{\text{gcd}(p-1, q-1)}. \tag{3}$$

Rearranging:

$$\phi = \text{gift} \cdot \text{gcd}(p-1, q-1). \tag{4}$$

The solution script sets $\phi = \text{gift} \cdot 2$, implying gcd$(p-1, q-1) = 2$. Since $p$ and $q$ are 1024-bit primes (odd), $p-1$ and $q-1$ are even, so their GCD is at least 2. The challenge ensures gcd$(p-1, q-1) = 2$.

The exponent $e = 54722 = 2 \cdot 27361$ is even, so gcd$(e, \phi) = 2$. However, $e/2 = 27361$ is coprime with $\phi$. The ciphertext is:

$$c = m^e \mod n = (m^2)^{e/2} \mod n. \tag{5}$$

Compute the private key for $e/2$:

$$d = (e/2)^{-1} \mod \phi. \tag{6}$$

Decrypt to get $m^2$:

$$m^2 = c^d \mod n. \tag{7}$$

Take the square root:

$$m = \sqrt{m^2}. \tag{8}$$

Convert $m$ to bytes.

### 2.3 Key Code

```python
phi = gift * 2
e = 54722
d = inverse(e//2, phi)
print(long_to_bytes(gmpy2.iroot(pow(c, int(d), n), 2)[0]))
```

### 2.4 Flag

Securinets{diff1cult_rsa_1s_e@sy_xxxxxxxxxxxxxxxxxx}

## 3 Challenge 2: syb3lik

### 3.1 Overview

**syb3lik** is an elliptic curve-based challenge. We must decrypt three AES-encrypted messages. We are given:

- An elliptic curve over prime $p$ with generator $G$, parameters $a$, $b$.
- Ability to choose a point $P$.
- Server-generated point $Q$.
- Three ciphertexts (IV + AES-CBC encrypted messages).

### 3.2 Solution

The challenge uses a weak RNG:

```python
class RNG:
    def __init__(self, seed, P, Q):
        self.seed = seed
        self.P = P
        self.Q = Q
    def next(self):
        s = E.multiply(self.P, self.seed).x
```

```
        self.seed = s
        r = E.multiply(self.Q, s).x
        return r & ((1 << 128) - 1)
```

Keys are derived as:

$$\text{key} = \text{sha1}(\text{str}(r))[: 16]. \tag{9}$$

The solution assumes the first $s$ (after one iteration) is small ($0$, $1$, or $2$). For small $s$, $r = (s \cdot Q).x \mod 2^{128}$ is predictable. We:

1. Choose a valid $P$ (provided).
2. Receive $Q$.
3. Compute keys for $s = 0, 1, 2$.
4. Try each key to decrypt the three ciphertexts.
5. Send decrypted messages to the server.

### 3.3 Key Code

```
possible_s = [0, 1, 2]
possible_keys = []
for s in possible_s:
    output = E.multiply(Q, s).x & ((1 << 128) - 1)
    key = hashlib.sha1(str(output).encode()).digest()[:16]
    possible_keys.append(key)
```

### 3.4 Flag

Securinets{D0ubl2_Tr0ubl2015749448494984 74}

## 4 Challenge 3: ezRSA

### 4.1 Overview

**ezRSA** splits the flag into two RSA-encrypted parts:

- **Part 1**: $n_1$, $c_1$, $\text{hint}_1 = x_1 p + y_1 q - 0x114$, $\text{hint}_2 = x_2 p + y_2 q - 0x514$, $x_1, x_2 < 2^{11}$, $y_1 < 2^{114}$, $y_2 < 2^{514}$, $e = 65537$.
- **Part 2**: $n_2$, $c_2$, $\text{hint} = (514p - 114q)^{n-p-q} \mod n$, $e = 65537$.

## 4.2 Solution

### 4.2.1 Part 1

Hints:

$$\text{hint}_1 = x_1 p + y_1 q - 0x114, \tag{10}$$
$$\text{hint}_2 = x_2 p + y_2 q - 0x514. \tag{11}$$

Rewrite:

$$\text{hint}_1 + 0x114 = x_1 p + y_1 q, \tag{12}$$
$$\text{hint}_2 + 0x514 = x_2 p + y_2 q. \tag{13}$$

Eliminate $q$:

$$(\text{hint}_1 + 0x114) \cdot x_2 - (\text{hint}_2 + 0x514) \cdot x_1 = (y_1 x_2 - y_2 x_1) \cdot q. \tag{14}$$

Brute-force $x_1, x_2 < 2^{11}$ to find $p = \gcd(\text{temp}, n_1)$. Compute $q = n_1/p$, $\phi$, $d$, and decrypt $c_1$.

### 4.2.2 Part 2

Hint:

$$\text{hint} = (514p - 114q)^{n-p-q} \mod n. \tag{15}$$

Since $n - p - q = \phi$:

$$\text{hint} \cdot (514p - 114q)^{-1} = 1 \mod n. \tag{16}$$

Solve:

$$514p - 114q = \text{hint}^{-1} \mod n_2, \tag{17}$$
$$p \cdot q = n_2. \tag{18}$$

Compute $\phi$, $d$, and decrypt $c_2$.

## 4.3 Key Code

```python
# Part 1
for i in range(2**11 + 1):
    for j in range(2**11 + 1):
        temp = (hint1_1 + 0x114) * i - (hint1_2 + 0x514) * j
        g = GCD(temp, n1)
        if g != 1 and g != n1:
            p = g
            q = n1 // p
            phi = (p - 1) * (q - 1)
            d = inverse(e, phi)
            flag1 = long_to_bytes(pow(c1, d, n1))
            break
# Part 2
```

```
temp = inverse(hint2, n2)
p, q = symbols('p q')
equation1 = Eq(514 * p - 114 * q, temp)
equation2 = Eq(p * q, n2)
solutions = solve((equation1, equation2), (p, q))
```

## 4.4 Flag

Securinets{$\sim: L1n34r\_Pr1m3E\_114!!!!$}Securinets{$\sim: L1n34r\_Pr1m3E\_114!!!!$}Securinets{$\sim: L1n34r\_Pr1$

# 5 Challenge 4: hash101

## 5.1 Overview

**hash101** uses RSA over complex numbers and ChaCha20:

- $n = p \cdot q$, $e = 3$.
- mh $= [(m.\text{re} \gg 128 \ll 128), (m.\text{im} \gg 128 \ll 128)]$.
- $C = [c.\text{re}, c.\text{im}]$, where $c = m^3 \mod n$.
- enc: ChaCha20-encrypted flag with key sha256$(\text{str}(m.\text{re} + m.\text{im}))$.

## 5.2 Solution

Let $m.\text{re} = a_{\text{high}} + a_{\text{low}}$, $m.\text{im} = b_{\text{high}} + b_{\text{low}}$, where $a_{\text{low}}, b_{\text{low}} < 2^{128}$. From $c.\text{re}$:

$$(a_{\text{high}} + x)^3 - 3(a_{\text{high}} + x)(b_{\text{high}} + y)^2 - c.\text{re} = 0 \mod n. \tag{19}$$

Use Coppersmith's method to find $x, y < 2^{128}$. Compute $m.\text{re}$, $m.\text{im}$, the key, and decrypt with ChaCha20.

## 5.3 Key Code

```
P.<x,y>=PolynomialRing(Zmod(n))
f = (a_high+x)^3-3*(a_high+x)*(b_high+y)^2-hint1
a_low, b_low = small_roots(f, [2^128, 2^128], 3)[0]
key = hashlib.sha256(str(a + b).encode()).digest()
cipher = ChaCha20.new(key=key, nonce=b'Pr3d1ctmyxjj')
flag = cipher.decrypt(enc)
```

## 5.4 Flag

Securinets{h4sh3d_w1th_l0v3_and_0ff_by_0ne_err0rs}

# 6 Challenge 5: ezMATH

## 6.1 Overview

**ezMATH** is an RSA challenge with:

- $n = p \cdot q$, $e = 65537$.
- $c$: Encrypted flag.
- hint $= (2024p + 2025)^q \mod n$.

## 6.2 Solution

The hint:

$$\text{hint} = (2024p + 2025)^q \mod n. \tag{20}$$

Compute:

$$\text{hint} - 2025^q \mod n. \tag{21}$$

Since $(a + b)^q \equiv a^q + b^q \mod q$:

$$(2024p + 2025)^q \equiv 2025^q \mod q. \tag{22}$$

Thus, $q$ divides hint $- 2025^n \mod n$. Compute:

$$p = \gcd(n, \text{hint} - 2025^n \mod n). \tag{23}$$

Then $q = n/p$, compute $\phi$, $d$, and decrypt $c$.

## 6.3 Key Code

```
p = GCD(n, hint - pow(2025, n, n))
print(long_to_bytes(pow(c, inverse(e, (p-1)*(n//p-1)), n)))
```

## 6.4 Flag

Securinets{n0_m0r3_m4th_plz_just_g1v3_m3_th3_fl4g}

# 7 Challenge 6: QUANTUM-BB84

## 7.1 Overview

**QUANTUM-BB84** simulates the BB84 protocol with:

- `qubits`: 100,000 qubits as complex numbers.
- `bob_bases`: Bob's measurement bases ($+$ or $x$).
- `ciphertext`: XOR-encrypted flag.
- Fixed random seed (999999999).

## 7.2 Solution

In BB84, Alice generates bits and bases:

- $+$-basis: $0 = |0\rangle = (0, 1)$, $1 = |1\rangle = (1, 0)$.
- $x$-basis: $0 = |+\rangle = (1/\sqrt{2}, 1/\sqrt{2})$, $1 = |-\rangle = (1/\sqrt{2}, -1/\sqrt{2})$.

Bob measures in random bases. If bases match, Bob gets Alice's bit; otherwise, a random bit. The shared key is Alice's bits where bases match.

The fixed seed makes all random choices deterministic. Given `qubits` and `bob_bases`, we:

1. Decode `qubits`:
   - $+$-basis: $(0, 1) \to 0$, $(1, 0) \to 1$.
   - $x$-basis: $(1/\sqrt{2}, 1/\sqrt{2}) \to 0$, $(1/\sqrt{2}, -1/\sqrt{2}) \to 1$.
2. Extract shared key where `bob_bases` match Alice's bases.
3. XOR with ciphertext bits to recover the flag.

## 7.3 Key Code

```python
for i, base in enumerate(bob_bases):
    qubit = qubits[i]
    real = qubit['real']
    imag = qubit['imag']
    if base == '+':
        if imag == 1.0:
            alice_bit = 0
        elif real == 1.0:
            alice_bit = 1
        else:
            continue
    elif base == 'x':
        if math.isclose(real, 1 / math.sqrt(2)) and
           math.isclose(imag, 1 / math.sqrt(2)):
            alice_bit = 0
        elif math.isclose(real, 1 / math.sqrt(2)) and
           math.isclose(imag, -1 / math.sqrt(2)):
            alice_bit = 1
        else:
            continue
    shared_key_bits.append(alice_bit)
```

## 7.4 Flag

Securinets{QKD_zzzzzzzzzzzzzzzzzzMrx0rd}

# 8 Conclusion

These challenges showcase diverse cryptographic techniques:

- **ezRSA+**: Non-standard RSA with a gift.
- **syb3lik**: Weak elliptic curve RNG.
- **ezRSA**: Linear algebra for RSA.
- **hash101**: Coppersmith's method.
- **ezMATH**: GCD-based factoring.
- **QUANTUM-BB84**: Deterministic BB84.

Each puzzle is educational and engaging.

**Author: MRx0rd for CyberTek CTF**