

Thesis Title:

Solving CTF Cryptography Problems with LLMs

Sebastian Newberry

November 7, 2025

Abstract

Open-source large language models (LLMs) have closed much—but not all—of the gap with frontier proprietary systems, and their relative standing varies sharply by benchmark family. On knowledge-heavy and long-horizon reasoning tasks (e.g., Humanity’s Last Exam, GPQA-Diamond, MMLU-Pro), top closed models such as GPT-5, Claude Sonnet 4.5, and Grok 4 generally retain an edge, aided by larger training mixtures and aggressive test-time compute. By contrast, recent open models (DeepSeek, GLM-4.6, Kimi K2, Qwen) often match or beat peers in coding and web-agent tasks (e.g., LiveCodeBench, SWE-bench Verified, BrowseComp), especially when tool use is allowed. Still, performance is benchmark-sensitive: DeepSeek R1/V3 trails on human-preference arenas despite strong math/code, GLM-4.6 shines on code but not always on long-form reasoning, and Kimi K2 leads some agentic evaluations yet lags top closed-source models on others. These disparities highlight how architecture (dense vs. MoE), alignment strategy, tool-use policies, and test-time compute budget shape results—and why single “leaderboards” can mask important modality- and task-specific trade-offs. In this report, I will be focusing on testing these LLMs in trying to solve complex cybersecurity challenges. These challenges are all multi-step problems that require lots of thinking and effort to solve correctly.

Purpose

Purpose: Companies today are turning to automated solutions like large language models to penetration test their infrastructure.

SmileyCTF 2025 – SaaS (Cryptography)

Difficulty: Easy

Setup and oracle

The server selects primes p, q with

$$p \equiv q \equiv 3 \pmod{4}, \quad n = pq, \quad e = 65537.$$

For an input $x \in \mathbb{Z}_n$ the oracle computes square roots modulo each prime and combines them by the Chinese Remainder Theorem (CRT). Concretely, let

$$r_p \equiv x^{(p+1)/4} \pmod{p}, \quad r_q \equiv x^{(q+1)/4} \pmod{q},$$

which are square roots of x modulo p and q respectively (when x is a quadratic residue). The oracle picks independent signs $a, b \in \{\pm 1\}$ and returns the CRT recombination

$$r = ar_p \cdot A + br_q \cdot B \pmod{n},$$

where we define the CRT basis elements

$$A \equiv q \cdot (q^{-1} \pmod{p}), \quad B \equiv p \cdot (p^{-1} \pmod{q}),$$

so that

$$A \equiv 1 \pmod{p}, \quad A \equiv 0 \pmod{q}, \quad B \equiv 0 \pmod{p}, \quad B \equiv 1 \pmod{q}.$$

Thus the oracle’s output is one of the four values

$$R = \{ ar_p A + br_q B \pmod{n} : a, b \in \{\pm 1\} \}.$$

Square roots modulo n and factor recovery

Let p and q be distinct primes with $p \equiv q \equiv 3 \pmod{4}$, and let $n = pq$. For an input $x \in \mathbb{Z}_n$, the oracle computes square roots modulo each prime and recombines them using the Chinese Remainder Theorem (CRT).

Square roots modulo a prime

Let p be an odd prime with $p \equiv 3 \pmod{4}$. If x is a quadratic residue modulo p (and $x \not\equiv 0 \pmod{p}$), Euler's criterion gives

$$x^{\frac{p-1}{2}} \equiv 1 \pmod{p}.$$

Consider

$$\left(x^{\frac{p+1}{4}}\right)^2 = x^{\frac{p+1}{2}} = x \cdot x^{\frac{p-1}{2}} \equiv x \cdot 1 \equiv x \pmod{p},$$

so $x^{(p+1)/4}$ is indeed a square root of x modulo p :

$$\boxed{\left(x^{(p+1)/4}\right)^2 \equiv x \pmod{p}}.$$

If x is not a quadratic residue modulo p , the same computation yields

$$\left(x^{(p+1)/4}\right)^2 \equiv -x \pmod{p},$$

Thus in all non-degenerate cases the value r_p produced by the local exponentiation satisfies

$$r_p^2 \equiv \pm x \pmod{p},$$

and similarly $r_q^2 \equiv \pm x \pmod{q}$.

CRT recombination of square roots modulo n

Let

$$r_p \equiv x^{(p+1)/4} \pmod{p}, \quad r_q \equiv x^{(q+1)/4} \pmod{q},$$

and define the CRT basis elements

$$A \equiv q \cdot (q^{-1} \bmod p), \quad B \equiv p \cdot (p^{-1} \bmod q),$$

so that

$$A \equiv 1 \pmod{p}, A \equiv 0 \pmod{q}, \quad B \equiv 0 \pmod{p}, B \equiv 1 \pmod{q}.$$

For independent signs $a, b \in \{\pm 1\}$, the four CRT recombinations are

$$r_{a,b} \equiv ar_p A + br_q B \pmod{n}.$$

Reducing modulo p and q shows that each $r_{a,b}$ is a square root of x modulo n :

$$r_{a,b}^2 \equiv r_p^2 \equiv x \pmod{p}, \quad r_{a,b}^2 \equiv r_q^2 \equiv x \pmod{q} \implies r_{a,b}^2 \equiv x \pmod{n}.$$

Recovering n from complementary roots

If two roots correspond to opposite signs, say (a, b) and $(-a, -b)$, then

$$r_{a,b} + r_{-a,-b} \equiv 0 \pmod{n}.$$

Taking integer representatives in $[0, n-1]$, this means

$$r_{-a,-b} \equiv n - r_{a,b}.$$

Hence, after sorting the four integer roots, the sum of the smallest and largest root recovers n :

$$\boxed{n = r_{\min} + r_{\max}}.$$

Extracting a prime via a GCD

Different sign choices produce distinct roots. Consider a pair differing only in the q -component:

$$r_{a,b} = ar_p A + br_q B, \quad r_{a,-b} = ar_p A - br_q B.$$

Their difference is

$$r_{a,b} - r_{a,-b} = 2br_q B.$$

Reducing this difference modulo the primes gives the two congruences

$$r_{a,b} - r_{a,-b} \equiv 0 \pmod{p}, \quad r_{a,b} - r_{a,-b} \equiv 2br_q \pmod{q}.$$

Thus the difference is divisible by p and congruent to $2br_q$ modulo q . Therefore the greatest common divisor yields the prime factor:

$$\boxed{\gcd(r_{a,b} - r_{a,-b}, n) = p.}$$

Symmetrically, taking two roots that differ only in the p -component reveals q . In practice one collects the four CRT roots, picks an appropriate pair, and computes

$$p = \gcd(r_i - r_j, n), \quad q = n/p.$$

Forge the signature

Once p, q are known we compute

$$\varphi(n) = (p-1)(q-1), \quad d \equiv e^{-1} \pmod{\varphi(n)}.$$

Given the challenge's printed value m the RSA signature is forged as

$$\boxed{s \equiv m^d \pmod{n}.}$$

Submitting s to the server satisfies $s^e \equiv m \pmod{n}$ and retrieves the flag.

Summary of steps used in the exploit

1. Repeatedly query the oracle with a fixed quadratic residue (the solver uses $x = 4$) to collect several of the four square roots.
2. From the four roots compute n as the sum of the smallest and largest root.
3. Use a difference of an appropriate pair and $\gcd(\cdot, n)$ to recover one prime p (the other is $q = n/p$).
4. Compute $d = e^{-1} \pmod{\varphi(n)}$ and then $s = m^d \pmod{n}$.