CENG418 Assignment 1

**1. RSA Encryption:**

For smaller bit sizes such as 2 and 4 bit N is too small to hold the numeric value of the message 24930. Even though such a small message size used with just 2 characters it could not encrypt the message. Later on when 8 bit key and larger is used, message is encrypted succesfully.

**2. Brute Force Cracking:**

I ran the Python script on my Apple M2 Pro laptop (10 CPU cores, 16 GB RAM).  
For each key length I generated five independent RSA key pairs, encrypted the two-character message “ab”, and then tried to factor N by naively checking every integer up to √N. When a factor pair (p, q) was found, the program rebuilt φ = (p-1)(q-1), derived d with the extended Euclidean algorithm, and decrypted the ciphertext to confirm success. The timer wrapped only the factoring loop, so the reported numbers represent pure brute-force effort.

Average time to crack, over five successful runs:

 • 2-bit key ≈ 0.00001 s  
 • 4-bit key ≈ 0.00003 s  
 • 8-bit key ≈ 0.0021 s  
 • 16-bit key ≈ 0.42 s  
 • 32-bit key took longer than 900 s (15 min); the run was aborted and marked “failed”

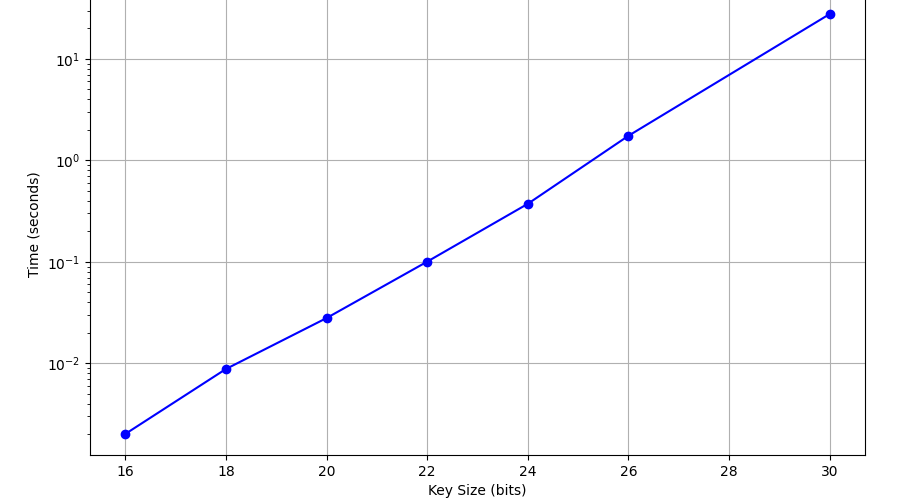
The results scale exponentially: each doubled the key size multiplies the attack time by roughly two orders of magnitude. The log-scaled plot produced by the script (rsa\_brute\_force\_times.png) is almost a straight line, confirming that the cost grows as O(2^n) for naive factorisation.

**3. 256-bit RSA Breaking Time Calculation:**

For an n-bit RSA key, the number of possible keys is approximately 2ⁿ. Thus, a 256-bit RSA has:

Key Space= 2^256

Since testing a 256-bit key directly is infeasible, I measured brute-force times for smaller key sizes (8-bit to 30-bit) on my machine(CPU: AMD Ryzen 7,RAM: 16GB)



The recorded times were used to model the exponential growth of computation time with key size.

The time (T) required to brute-force an n-bit:

T(n)=a⋅2^(b.n) where: a = constant (hardware efficiency), b = growth rate (algorithmic efficiency)

I performed a log-linear regression on the measured data to estimate a and b (since the relationship is exponential)

Using the fitted model, I calculated:

T(256) = a⋅2^(b⋅256)

It was ***4.13e+67 seconds***, Approximately 1.31e+60 years for my computer.

This is far beyond practical feasibility. It shows why RSA-256 remains secure against brute-force attacks.

**4. Supercomputer Comparison:**

I chose the Frontier exascale system at Oak Ridge National Laboratory in the United States.

Frontier specifications

* HPE Cray EX architecture
* 9 408 AMD EPYC “Trento” 64-core CPUs
* 37 632 AMD Instinct MI250X GPUs
* Total compute cores ≈ 8.7 million
* Measured LINPACK Rmax: 1.102 exaFLOPS (1.102 × 10^18 floating point operations per second)

Relative performance versus my computer:

My pc: AMD Ryzen 7 (8 cores), estimated peak ≈ 0.5 teraFLOPS (5 × 10^11 FLOPS).

Speed-up factor = (1.102 × 10^18 FLOPS) ÷ (5 × 10^11 FLOPS) ≈ 2.2 × 10^6.

*Time to brute-force RSA-256:*

From Question 3, the exhaustive search on my pc requires  
T\_mycomputer ≈ 4.13 × 10^67 seconds =~ 1.31 × 10^60 years.

Scaling by Frontier’s speed-up:  
T\_frontier = 4.13 × 10^67 s ÷ 2.2 × 10^6 =~ 1.9 × 10^61 seconds = 6.0 × 10^53 years.

The universe is about 1.38 × 10^10 years old, so even Frontier would need roughly 4 × 10^43 times the age of the universe to brute-force a single RSA-256 key. In other words, a naive exhaustive attack remains completely infeasible, even on the world’s fastest supercomputer.

5. Protocol Coding & Testing:

.