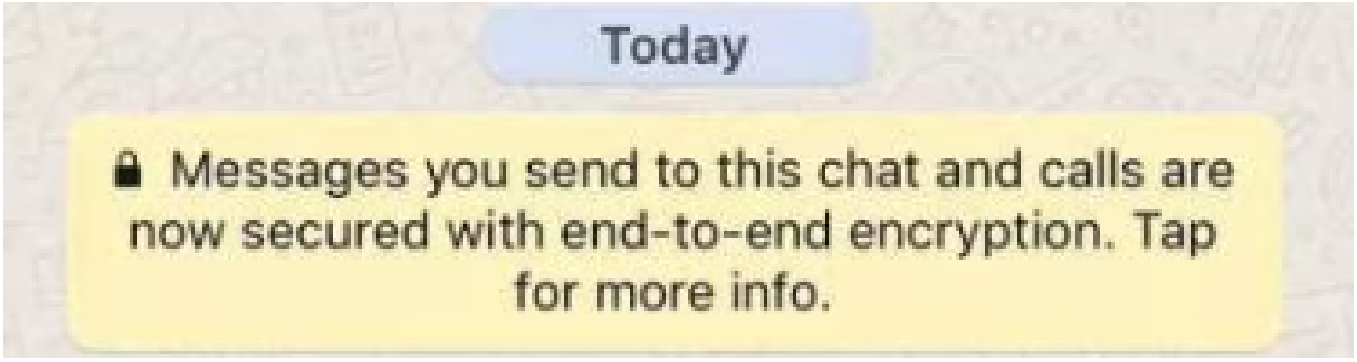
How to Win a 1,000,000$ (or at Least a Cookie) through Cryptography: Computational Analysis of the Security of the RSA Encryption Algorithm

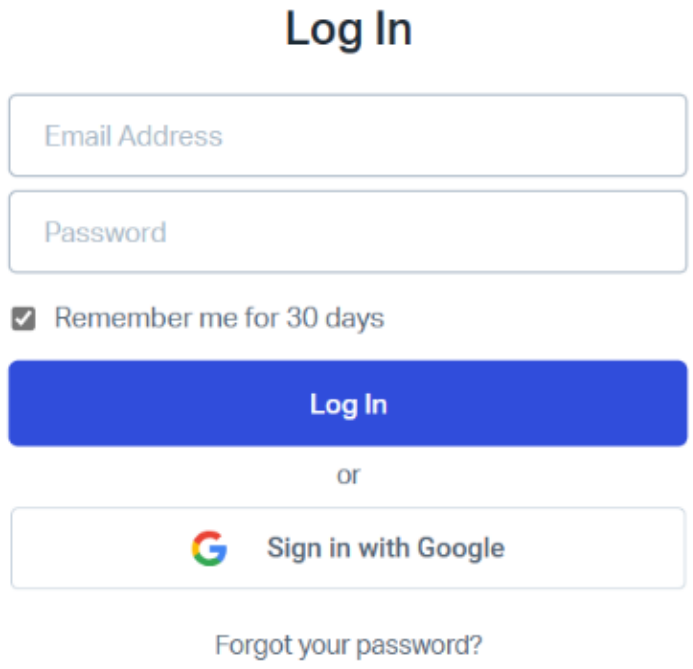
Andrey Ambartsumov

Supervised by Professor Takei

May 8th, 2024

Why Cryptography

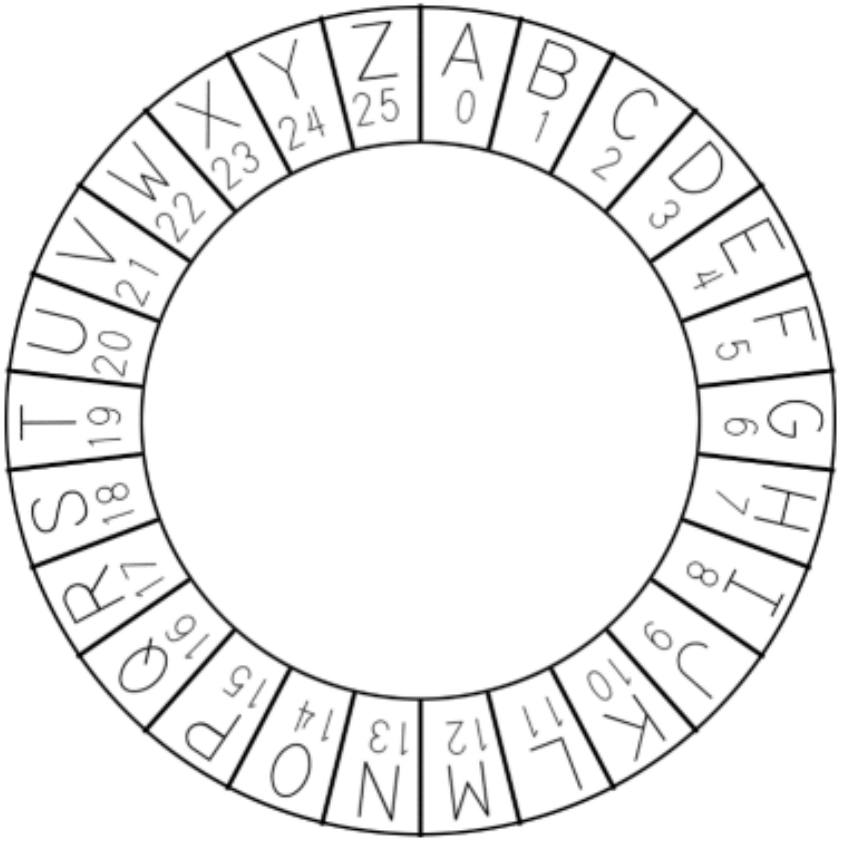


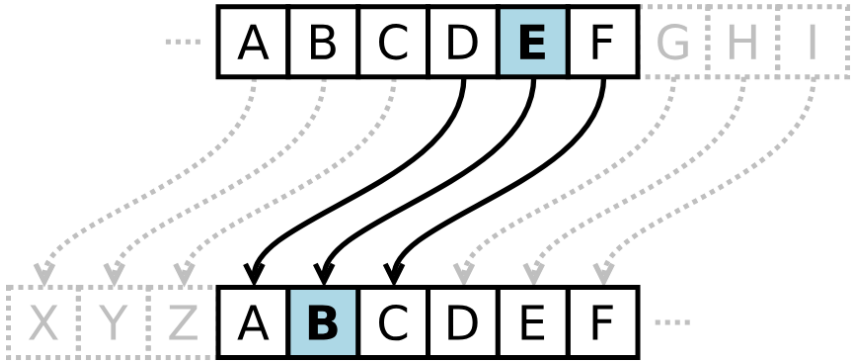


Caesar Cipher - Julius Caesar, the first Cryptographer

Julius Caesar invented the Caesar Cipher in 100BC.

Caesar Cipher - Example





https://ceasarcipher.onrender.com/

RSA - Analogy

RSA - Analogy


THEYARD.NYC
TD
TD
TD Bank
Open 7 Days
TD
TD Bank


HARDENED
Guliffen
Guliffen

RSA - Analogy

RSA - Analogy


TD
TD
TD Bank
Open 7 Days
TD


HARDENED
Guliffen
Guliffen


HARDENED

RSA - Explanation

Public Key Pair

Private Key

RSA - Send a Secret

Bob's Public Keys: (33,7)

Alice: G $\rightarrow$ 6

RSA - Send a Secret

Bob's Public Keys: $(33,7)$

Alice: G $\rightarrow$ 6

$$
\begin{aligned}
6^7 &= 279936 \\
&= 33 \cdot 8482 + 30 \\
&\equiv 30 \pmod{33}
\end{aligned}
$$

RSA - Send a Secret

Bob's Public Keys: $(33,7)$

Alice: G $\rightarrow$ 6

$$6^7 = 279936$$
$$= 33 \cdot 8482 + 30$$
$$\equiv 30 \pmod{33}$$

On Bob's side, he has the secret key 3:

$$30^3 = 27000$$
$$\equiv 6 \pmod{33}$$

RSA - Choosing the Keys

- Encryption modulus: $33 = 3 \times 11$
- Encryption and Decryption Exponents:

$$\begin{aligned} 7 \cdot 3 &= (3-1)(11-1)n + 1 \\ 21 &= 20n + 1 \\ 21 &\equiv 1 \pmod{20} \end{aligned}$$

RSA - Security

1. n^{th} roots in modular

$$m^7 = 33n + 30$$

$$\sqrt[7]{33n + 30}$$

$$6 = \sqrt[7]{33 \cdot 8482 + 30}$$

RSA - Security

1. n^{th} roots in modular

$$m^7 = 33n + 30$$

$$\sqrt[7]{33n + 30}$$

$$6 = \sqrt[7]{33 \cdot 8482 + 30}$$

2. Factorize the encryption modulus into p and q

$$33 = 3 \text{ x } 11$$

RSA - Security

1. n^{th} roots in modular

$$m^7 = 33n + 30$$

$$\sqrt[7]{33n + 30}$$

$$6 = \sqrt[7]{33 \cdot 8482 + 30}$$

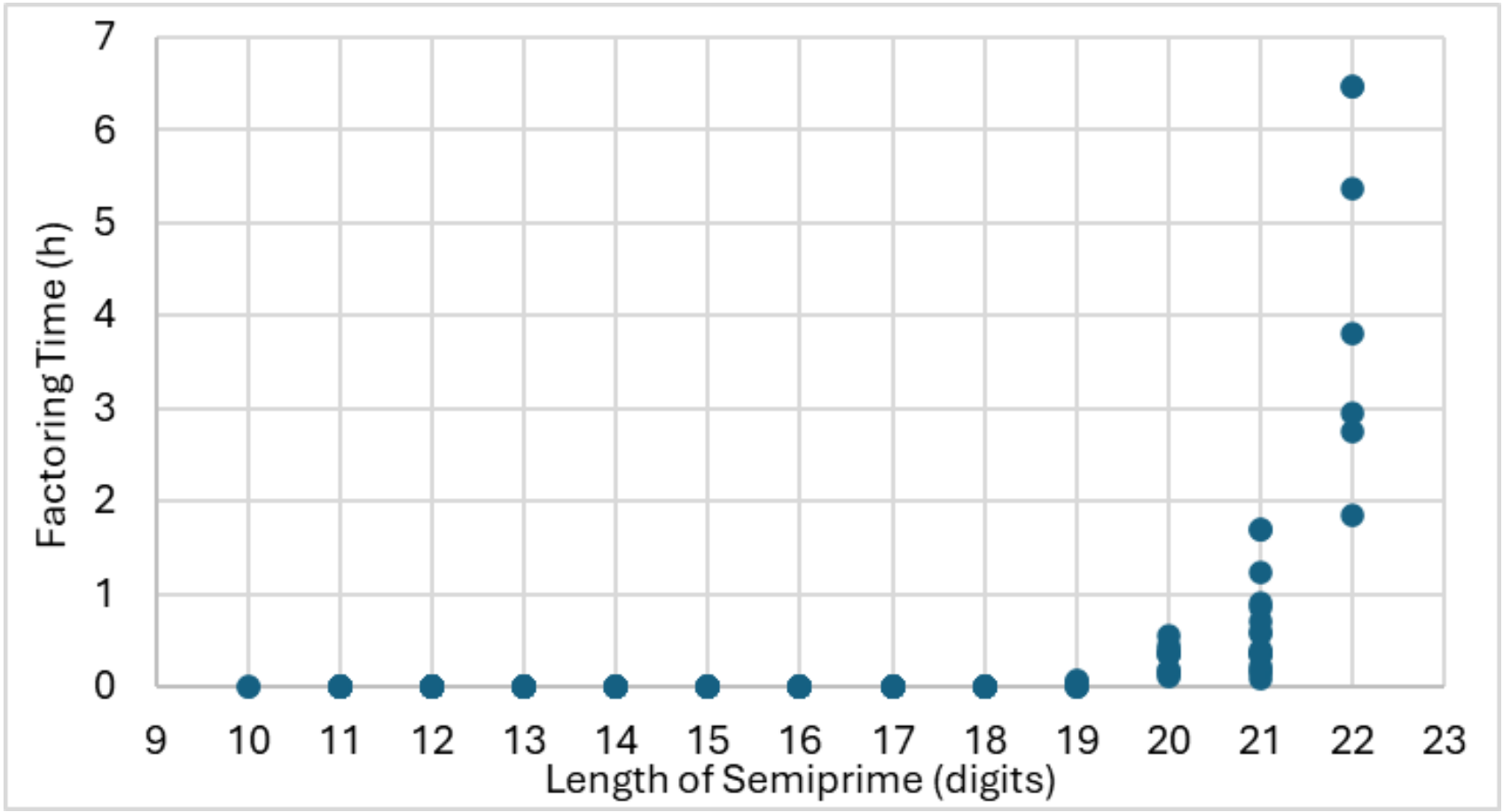
2. Factorize the encryption modulus into p and q

$$33 = 3 \text{ x } 11$$

$$7 \cdot key = (3 - 1)(11 - 1)n + 1$$

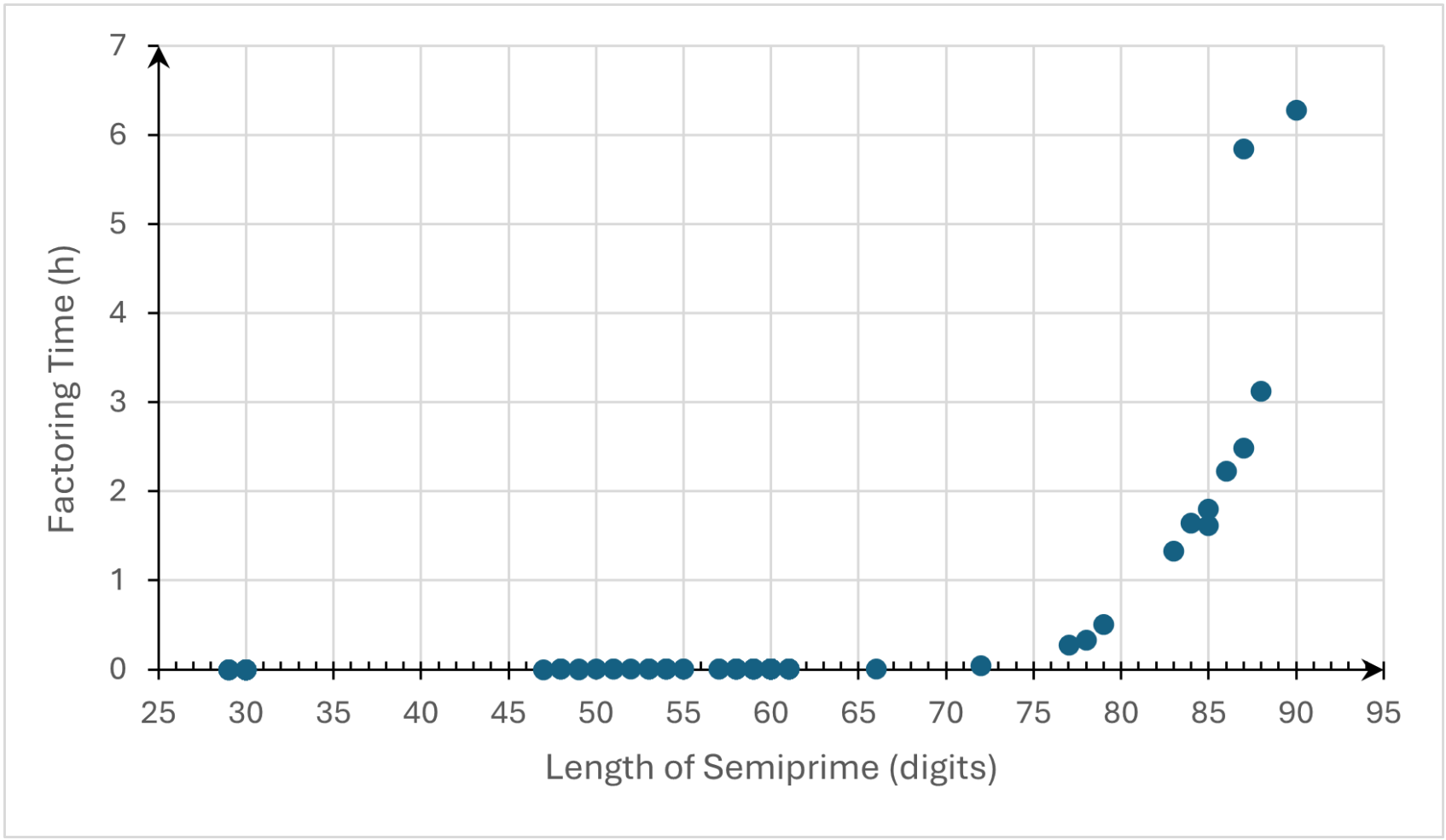
$$key = 3$$

RSA - Trail and Error Factorization using Python



Factorization time of semiprimes using trial and error

RSA - Quadratic Sieve using C++



Factorization time of semiprimes using the Quadratic Sieve

RSA - 90 digits semiprime

This semi-prime took 7 hours to factorize:

601348911492967474716626562591162015444606322948636213
084003067539621391421673708075577
= 880748131971770591613567759127557026653582 09 x
68277057840213697348901222523555727513545 53

However, to encrypt it takes:

0.00099730491638183 6 seconds

RSA - 309 digits Encryption Modulus

An industry standard would be:

1437832489048230458268954124427515656991812670435271598107249824368398178285945362131773362655027756794178034116111605432812511130074934056582606343715763469499504203051365348502657140237133463474343604517177518695762038284173906148107290849160323765547206418773039754189427597424671098104124210954876063372 09

Decrypting Challenges!

Head onto:

www.cookiechallenge.live

To have a chance to win cookies!

Good luck!