CURVAS ELIPTÍCAS

Gian Ferrari (22100619)
Guilherme Adenilson de Jesus (22100620)
Gustavo Konescki Fuhr (22203675)

Introdução

Objetivos e Motivação

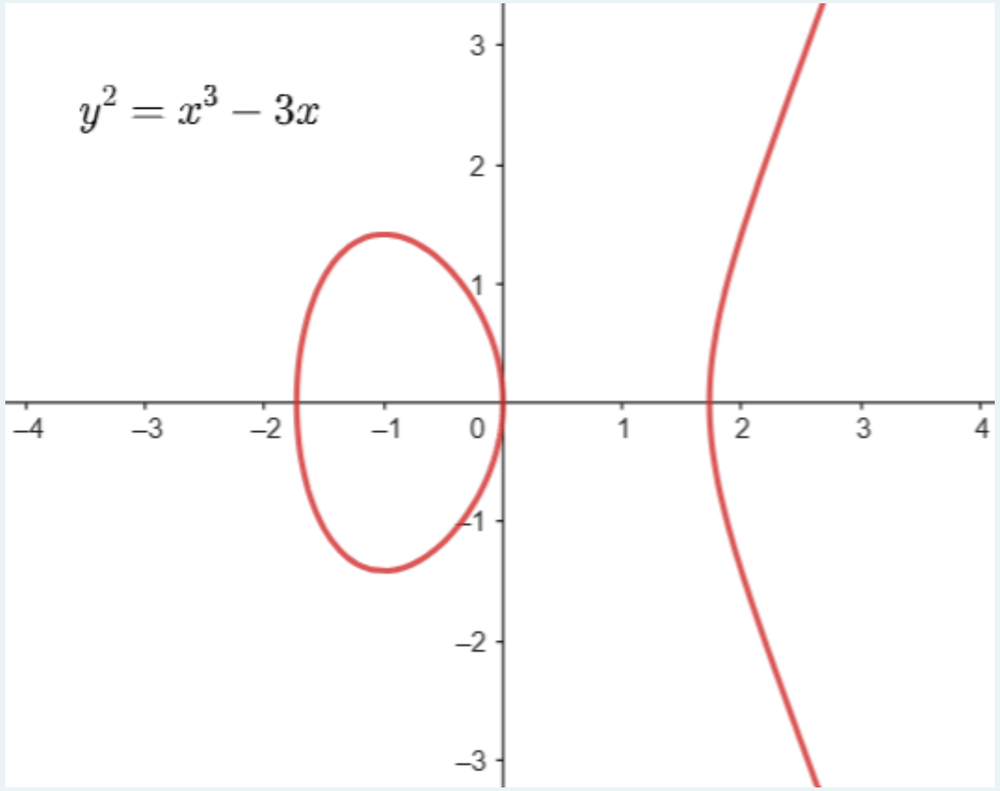
- Curvas Elípticas
  - Força Criptográfica
- Estudo Comparativo
  - Tempo de execução
  - Curvas vs. Curvas
  - Curvas vs. RSA
- Uso de Curvas Elípticas
  - ECDH

Desenvolvimento

Curvas Elípticas

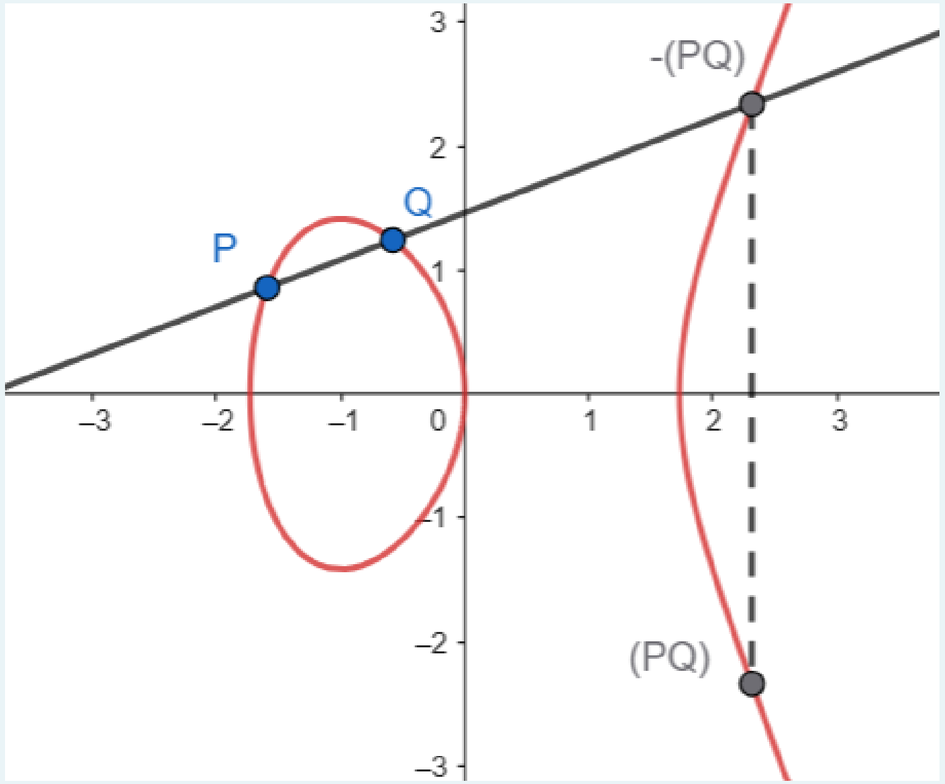
- Criptografia de Chave Pública
  - Protocolo Diffie-Hellman
  - Logaritmo Discreto
- Definição de Curva Elíptica
  - Definidas sobre um corpo K

$$E_K(a,b) = \{(x,y) \in K^2 : y^2 = x^3 + ax + b\} \cup \{\infty\}$$



Curvas Elípticas

- Operação de Grupo
  - Definida Geometricamente
  - Elemento Neutro
  - Contrapartida Algébrica
  - Definição de mP
- Codificação de Mensagens
- Dificuldade Computacional
  - Dados P e Q
  - Encontrar x tal que Q = xP
  - Analogia ao Logaritmo Discreto



Uso de Curvas Elípticas

- Protocolo ECDH
  - Análogo ao Diffie-Hellman usual
  - Curva e ponto gerador público
  - Cada parte escolhe um número aleatório
    - Digamos, Parte A escolhe n e parte B escolhe u
  - Chave de A: nP, Chave de B: uP
  - nP e uP são transmitidos
  - nuP = unP é calculada em cada parte

Padrões de Curvas Elípticas

- Diferenças:
  - Corpo subjacente
  - Parâmetros (a, b) da curva
  - Ponto gerador
- NIST
  - secp256r1, secp384r1, secp521r1
- SECG
  - secp256k1
- Brainpool
  - brainpoolP256r1, brainpoolP512r1
- ICP-Brasil

Experimento

Link Google Colab

Análises realizadas

- Google Colab (Python)
- Curvas da biblioteca Cryptography
- RSA de 1024, 2048 e 4096 bits
- Comparativo
  - Geração de chave
  - Assinatura
  - Verificação de assinatura
  - Diffie-Hellman (apenas curvas)

Nome da Curva	Tamanho da Chave	Origem/Padrão
SECP256K1	256 bits	SECG (SEC 2)
SECP256R1	256 bits	NIST P-256
SECP384R1	384 bits	NIST P-384
SECP521R1	521 bits	NIST P-521
BrainpoolP256r1	256 bits	Brainpool (RFC 5639)
SECT163K1	163 bits	NIST K-163
SECT163R2	163 bits	NIST B-163
SECP192R1	192 bits	NIST P-192
SECP224R1	224 bits	NIST P-224
SECT233K1	233 bits	NIST K-233
SECT233R1	233 bits	NIST B-233
SECT283K1	283 bits	NIST K-283
SECT283R1	283 bits	NIST B-283
BrainpoolP384R1	384 bits	Brainpool (RFC 5639)
SECT409K1	409 bits	NIST K-409
SECT409R1	409 bits	NIST B-409
BrainpoolP512R1	512 bits	Brainpool (RFC 5639)
SECT571K1	571 bits	NIST K-571
SECT571R1	571 bits	NIST B-571

Função de Medição

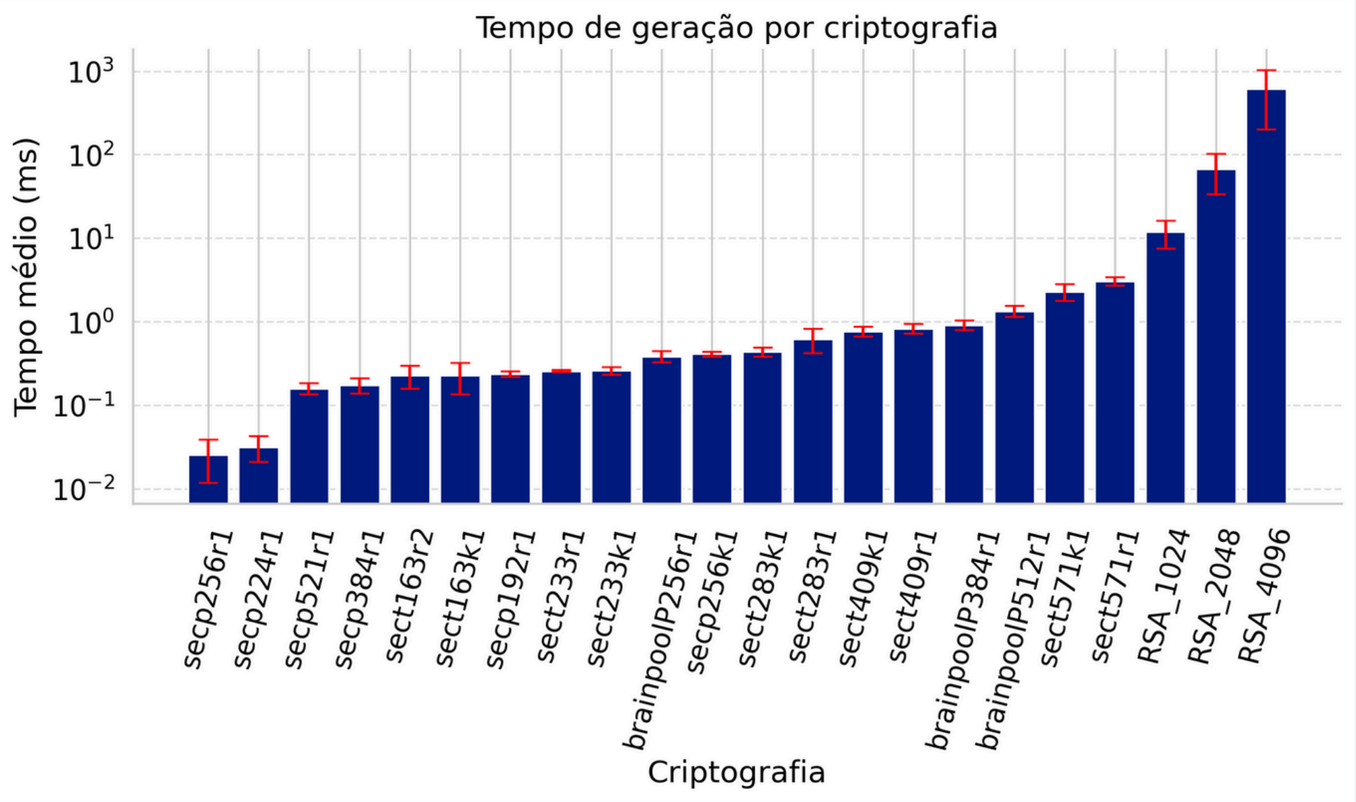
```
1    def realizar_medicao(op, args_op, n):
2        tempos = []
3        for _ in range(n):
4            start = time.process_time_ns()
5            op(*args_op)
6            end = time.process_time_ns()
7            tempos.append(end - start)
8
9        return np.array(tempos)
```

Geração de chave – Código

```
n = 100
tempos = dict()

for curva in curvas:
    tempo_curva = realizar_medicao(ec.generate_private_key, [curva
    ], n) / 1e6
    tempos[curva.name] = tempo_curva

for rsa_config in rsa_configs:
    tempo_curva = realizar_medicao(rsa.generate_private_key,
    rsa_config, n) / 1e6
    tempos[f'RSA_{rsa_config[-1]}'] = tempo_curva

resultados = pd.DataFrame(tempos)
resultados.describe()
```

Geração de chave – Resultado

- Curva secp256r1 (NIST P-256)
  - Menor tempo médio: 0,03 ms
  - Usa número primo pseudo-Mersenne
  - Permite reduções modulares rápidas
- RSA_4096
  - Maior tempo médio: 618,51 ms
  - 20.617 vezes maior que secp256r1
  - Exige a geração de dois números primos com teste de primalidade

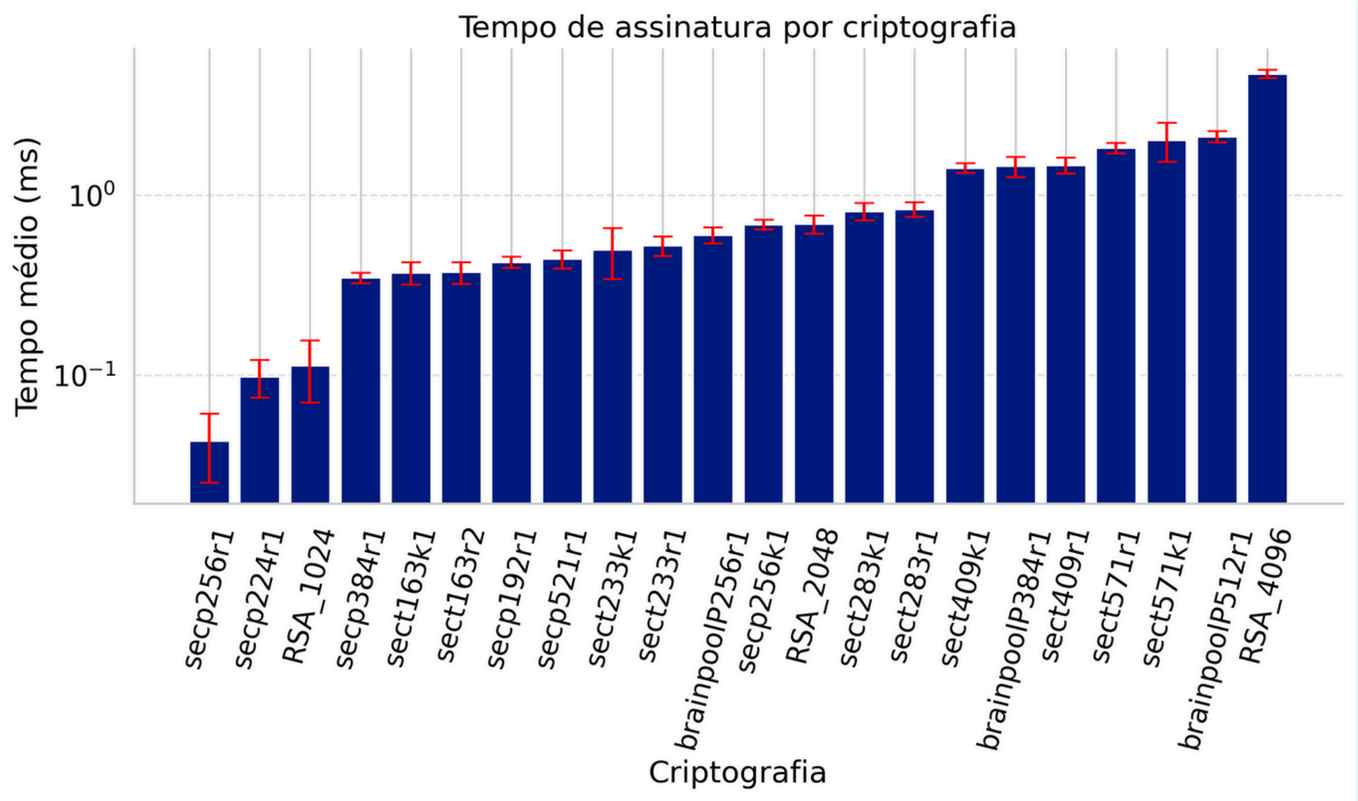


Assinatura – Código

```python
n = 100
tempos = dict()
data = b"Uma mensagem a ser assinada"
for curva in curvas:
  private_key = aux_curve_sign(curva)
  tempo_curva = realizar_medicao(private_key.sign, [data, ec.
   ECDSA(hashes.SHA256())], n) / 1e6
  tempos[curva.name] = tempo_curva

for rsa_config in rsa_configs:
  private_key = aux_rsa_sign(rsa_config)
  tempo_curva = realizar_medicao(private_key.sign, [data, padding
   .PSS(mgf=padding.MGF1(hashes.SHA256()), salt_length=padding.PSS
   .MAX_LENGTH), hashes.SHA256()], n) / 1e6
  tempos[f'RSA_{rsa_config[-1]}'] = tempo_curva

resultados = pd.DataFrame(tempos)
resultados.describe()
```

Assinatura – Resultado

- Curva secp256r1 (NIST P-256)
  - Menor tempo médio: 0,04 ms
  - Mesmo motivo que na geração de chaves
- RSA_1024
  - Terceiro menor tempo: 0,11 ms
  - Mais rápido que varias curvas
    - Curvas elípticas envolvem cálculos complexos
  - Possui tamanho de chave menor
    - Exponenciação modular mais rápida

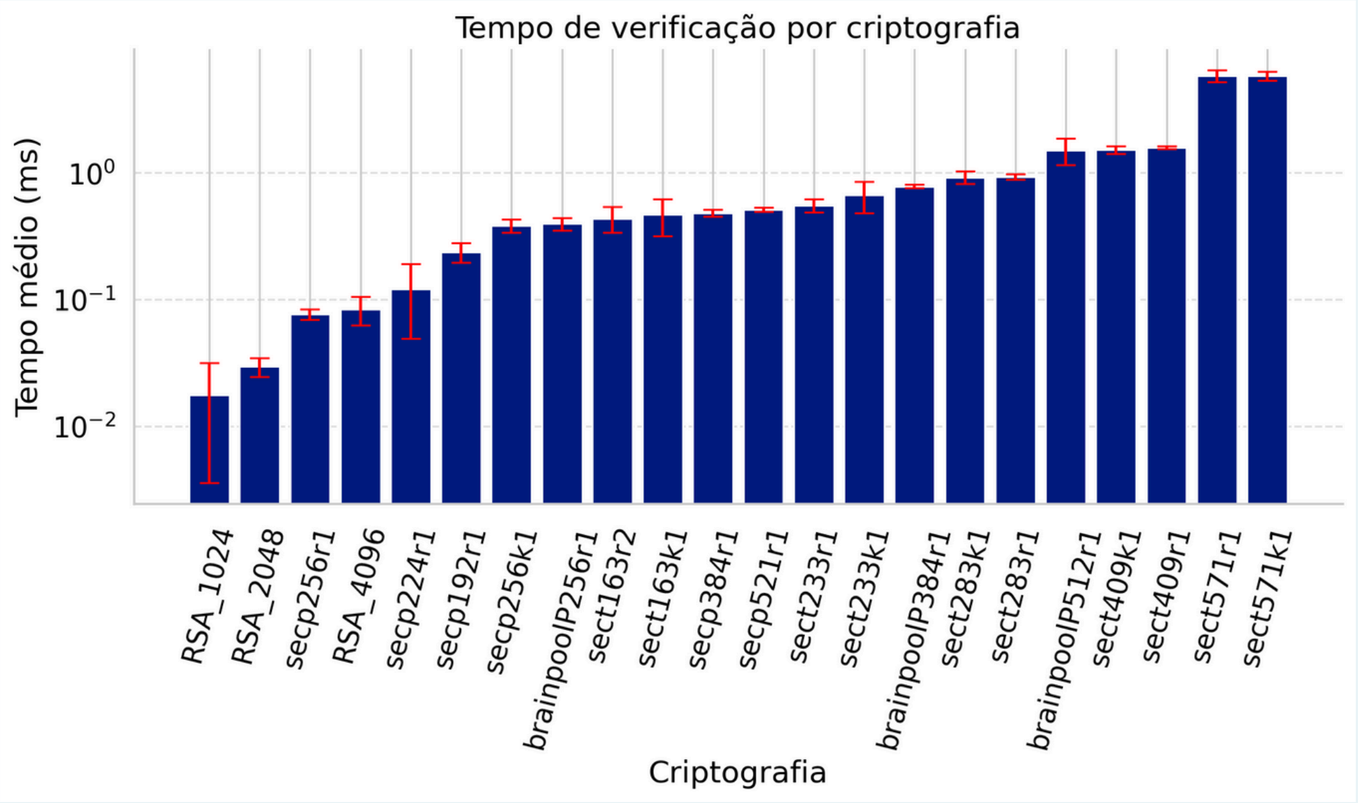


Verificação – Código

```
n = 100
tempos = dict()
data = b"Uma mensagem a ser assinada"
for curva in curvas:
  public_key, sig = aux_curve_verify(curva, data)
  tempo_curva = realizar_medicao(public_key.verify, [sig, data,
   ec.ECDSA(hashes.SHA256())], n) / 1e6

  tempos[curva.name] = tempo_curva

for rsa_config in rsa_configs:

  public_key, sig = aux_rsa_verify(rsa_config, data)
  tempo_curva = realizar_medicao(public_key.verify, [sig, data,
   padding.PSS(mgf=padding.MGF1(hashes.SHA256()), salt_length=
   padding.PSS.MAX_LENGTH),hashes.SHA256()], n) / 1e6

  tempos[f'RSA_{rsa_config[-1]}'] = tempo_curva

resultados = pd.DataFrame(tempos)
resultados.describe()
```

Verificação – Resultado

- RSA’s mais eficiente
  - Verificação mais simples
  - $m = s^e \mod n$
- Curvas elípticas
  - Operações mais complexas
    - Inversão modular

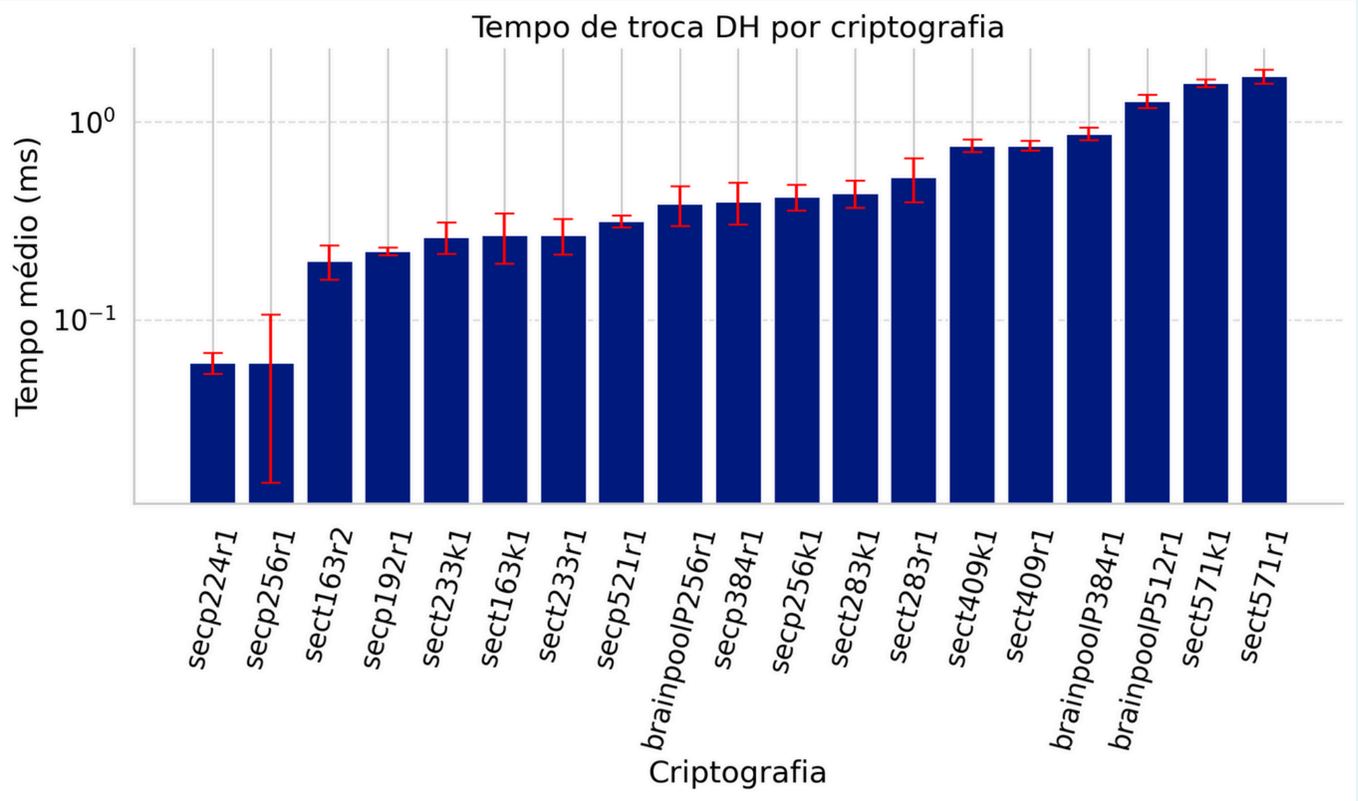


Diffie Hellman – Código

```
n = 100
tempos = dict()
for curva in curvas:
  server_private_key, peer_public_key = aux_curve_DH(curva)
  tempo_curva = realizar_medicao(server_private_key.exchange, [ec
   .ECDH(), peer_public_key], n) / 1e6

  tempos[curva.name] = tempo_curva

resultados = pd.DataFrame(tempos)
resultados.describe()
```

Diffie Hellman – Resultado

- Curvas secp224r1 e secp256r1
  - Menores tempos médios
    - ~0,06 ms
  - Primos de pseudo-Mersenne
  - Operações modulares mais rápidas
- Tamanho da chave influencia
  - Chaves maiores tendem a ser mais lentas



Conclusão

- Curvas elípticas são melhores para geração de chaves
- RSA se sobressai em verificação de assinatura
- O primo utilizado pode influenciar no desempenho da curva
- Primos de pseudo-Mersenne são mais otimizados

Referências

COELHO, M. A. Workshop Padrões e Tecnologias da ICP-Brasil. 2022. Acesso em: 01 jul. 2025. Disponível em: <https://workshopicpmercosul.paginas.ufsc.br/files/2022/04/Workshop_MERCOSUL_Padr%C3%B5es_e_Tecnologias_da_ICP_Brasil_Curvas_El%C3%ADpticas.pdf>. Citado na página 6.

CRYPTOGRAPHY. Elliptic curve cryptography. 2025. Acesso em: 28 jun. 2025. Disponível em: <https://cryptography.io/en/latest/hazmat/primitives/asymmetric/ec/#elliptic-curves>. Citado na página 7.

HANKERSON, D.; MENEZES, A.; VANSTONE, S. Guide to Elliptic Curve Cryptography. New York: Springer, 2004. Livro de referência sobre curvas elípticas em criptografia. ISBN 978-0387952734. Citado na página 5.

ITI. PADRÕES E ALGORITMOS CRIPTOGRÁFICOS DA ICP-BRASIL DOC ICP-01.01. 2022. Acesso em: 01 jul. 2025. Disponível em: <https://repositorio.iti.gov.br/instrucoes-normativas/IN2022_22_DOC-ICP-01.01.htm>. Citado na página 6.

KOBLITZ, N. Elliptic curve cryptosystems. Mathematics of Computation, American Mathematical Society, v. 48, n. 177, p. 203–209, 1987. Citado nas páginas 3 e 4.

LOCHTER, M.; MERKLE, J. Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation. 2010. Acesso em: 28 jun. 2025. Disponível em: <https://datatracker.ietf.org/doc/html/rfc5639>. Citado na página 6.

MARTINA, J. E.; IDALINO, T. B. Criptografia Assimétrica e Integridade. 2025. Slides de Aula da disciplina INE5429, Segurança em Computação, disponível via Moodle. Acessado em 29 jun. 2025. Citado na página 3.

NIST. FIPS 186-5: Digital Signature Standard (DSS). 2013. Acesso em: 28 jun. 2025. Disponível em: <https://csrc.nist.gov/pubs/fips/186-5/final>. Citado na página 5.

SECG. SEC 2: Recommended Elliptic Curve Domain Parameters. 2010. Acesso em: 28 jun. 2025. Disponível em: <https://www.secg.org/sec2-v2.pdf>. Citado na página 6.