INSTITUTO SUPERIOR POLITÉCNICO DE TECNOLOGIA E CIÊNCIAS

ENGENHARIA INFORMÁTICA

DISCIPLINA DE MATEMÁTICA DISCRETA

CURVAS ELÍPTICAS E CRIPTOGRAFIA

(IMPLEMENTAÇÃO EM PYTHON)

Nome: Jofre Jaime Jamuinda Mutumbungo

Nº: **20230394**

Turma: **EINF4_T3**

Docente**:** Valter Paulo Tomé

Luanda, junho de 2025

INDICE

1. INTRODUÇÃO TEÓRICA

Uma **curva elíptica** é uma curva plana definida por uma equação específica do tipo:

$$y^2 = x^3 + ax + b$$

Essa equação se chama **equação de Weierstrass**, e gera curvas suaves e simétricas — sem buracos, pontas ou cruzamentos.

Apesar do nome, **não tem relação com elipses**. O nome "elíptica" vem de sua conexão com **funções elípticas**, estudadas na matemática avançada. Embora tenham sido estudadas originalmente por matemáticos como **Euler**, **Gauss** e **Weierstrass**, as curvas elípticas passaram a ter grande utilidade em áreas práticas, principalmente:

Matemática:

- Estudo de equações diofantinas;
- Teoria dos números;
- Geometria algébrica;
- Resolução do Último Teorema de Fermat (por Andrew Wiles!);
- Criptografia (nos últimos 50 anos).

Criptografia (nos últimos 50 anos):

Curvas elípticas fornecem uma base segura para sistemas criptográficos modernos. Permitem criar **protocolos de chave pública** com **segurança alta usando chaves pequenas**, o que é ideal para dispositivos móveis, redes, e IoT.

Equação de Weierstrass e Condição de Validade

A equação usada em criptografia é:

$$y^2 = x^3 + ax + b$$

Onde:

- a e b são constantes.
- A equação define todos os pontos (x,y) que satisfazem essa relação.

Mas **nem toda escolha de a e b** produz uma curva válida! Para a curva ser "não singular", ela precisa satisfazer a seguinte condição:

$$4a^3 + 27b^2 \neq 0$$

Se essa condição **não for satisfeita**, a curva pode ter auto-interseções ou pontas agudas, o que inviabiliza seu uso em criptografia.

O Grupo dos Pontos da Curva

O que faz as curvas elípticas especiais para criptografia é que **seus pontos podem ser somados entre si**, formando o chamado **grupo aditivo de pontos da curva**.

Como funciona a adição de pontos?

- Dados dois pontos P e Q na curva, podemos calcular um terceiro ponto R = P+Q, que também pertence à curva.
- Existe uma fórmula algébrica e uma interpretação geométrica para essa operação.
- Existe um **ponto neutro** chamado **ponto no infinito** $\mathcal{O}$, que age como o "zero" da operação:

$$P + \mathcal{O} = P$$

Essa estrutura de grupo é a **base matemática dos sistemas criptográficos** com curvas elípticas.

Curvas Elípticas sobre Corpos Finitos

Na prática da criptografia, não usamos curvas elípticas sobre os **números reais R** (como em gráficos), mas sim sobre **corpos finitos**. O mais comum é o corpo finito dos inteiros módulo p, com p primo.

Definição:

Um **corpo finito F_p** é um conjunto com p elementos:

$$F_p = \{0,1,2,\dots,p-1\}$$

com as operações de:

- **soma**: a + b mod p
- **multiplicação**: a x b mod p
- **inverso multiplicativo**: existe a – 1 mod p, sempre que a ≠ 0

Equação sobre F_p:

$$y^2 1 \equiv x^3 + ax + b \; mod \; p$$

Aqui, todos os valores e operações são **módulo p**. Isso faz com que o número de pontos da curva seja **finito**, ideal para implementação computacional e análise criptográfica.

Exemplo prático:

Vamos considerar **p = 17, a = 2, b = 2**. A equação vira:

$$y^2 \equiv x^3 + 2x + 2 \; mod \; 17$$

Para encontrar os pontos da curva, testamos todos os valores de **x** e verificamos se o lado direito da equação é um **quadrado módulo 17** (isto é, se existe um y tal que $y^2 \equiv resultado \; mod \; 17$ **).**

2. FUNDAMENTOS MATEMÁTICOS

2.1 O Problema do Logaritmo Discreto em Curvas Elípticas (ECDLP)

O que é o ECDLP?

O Elliptic Curve Discrete Logarithm Problem (ECDLP) é a base da segurança dos sistemas de criptografia com curvas elípticas. Ele é o equivalente do problema do logaritmo discreto usado no algoritmo Diffie-Hellman clássico, mas aplicado sobre o grupo de pontos de uma curva elíptica.

Analogia simples

Imagine que temos um ponto **G** (conhecido como **ponto base**) em uma curva, e que multiplicamos esse ponto várias vezes:

$$\boldsymbol{P = n\ \ x\, G}$$

Onde:

G: ponto base público

n: número secreto (chave privada)

P: ponto público resultante

A **multiplicação escalar** aqui significa **somar G com ele mesmo nnn vezes** (não é multiplicação comum!).

Problema difícil:

Dado **G e P**, **descobrir n** é computacionalmente **muito difícil**, mesmo com computadores modernos.

Esse é o **ECDLP**, e a dificuldade de resolvê-lo é o que garante a segurança do ECC (Criptografia com Curvas Elípticas).

Por que é difícil: Porque, diferente da multiplicação comum, **não há uma fórmula direta ou reversa eficiente** para "dividir pontos" em uma curva elíptica. Só se pode resolver tentando todos os valores possíveis de n, o que é inviável para curvas grandes.

ex: **256 bits** → $\boldsymbol{2^{256}}$ possibilidades!

Comparação: Curvas sobre os Reais R vs. Corpos Finitos Fp.

Característica	Curvas sobre R	Curvas sobre Fp
Visualização	Curva contínua e suave	Conjunto de pontos discretos
Número de pontos	Infinito	Finito
Uso	Matemática pura, estudo teórico	Criptografia, implementação real
Operações	Reais, ponto flutuante	Inteiros módulo p
Segurança	Não usado diretamente	Base da ECC

Curvas sobre **R** são ótimas para estudo visual e teórico, mas **impraticáveis para computadores** por causa da imprecisão com números reais.

Já curvas sobre $\mathbf{F_p}$ são **perfeitas para computação discreta**, pois usam apenas inteiros e podem ser **armazenadas, transmitidas e calculadas com precisão**.

3. Curvas sobre $\mathbf{F}_{2^n}$

Além de $\mathbf{F_p}$, curvas elípticas também podem ser definidas sobre campos binários do tipo $\mathbf{F}_{2^n}$, muito usadas em hardware (smart cards, dispositivos IoT).

Vantagens:

- Mais rápidas em circuitos eletrônicos.
- Operações baseadas em bits.

Desvantagens:

- Mais complexas de entender e implementar.
- Algumas curvas $\mathbf{F}_{2^n}$ foram enfraquecidas por ataques criptográficos.

Por isso, atualmente, a preferência em segurança digital está em curvas sobre $\mathbf{F_p}$ como **secp256k1 (Bitcoin)** ou **Curve25519 (Signal, WhatsApp)**.

4. Criptografia com curvas Elípticas(ECC)

3.1 O que é?

ECC significa Elliptic Curve Cryptography (Criptografia de Curva Elíptica). É um conjunto de algoritmos de criptografia de chave pública baseados na dificuldade do ECDLP (Problema do Logaritmo Discreto em Curvas Elípticas).

Caracteristicas:

- Chaves menores para o mesmo nível de segurança comparado a RSA e DH.
- Mais leve e rápido, ideal para dispositivos móveis, cartões inteligentes, e IoT.
- Alta segurança com baixo custo computacional.

3.2 Protocolo ECDH (Elliptic Curve Diffie-Hellman)

ECDH é uma variação do **protocolo de troca de chaves Diffie-Hellman**, mas usando **curvas elípticas**.

Ele permite que duas partes (Alice e Bob) criem uma **chave secreta compartilhada**, mesmo que se comuniquem por um canal inseguro.

Como funciona?

1. **Parâmetros públicos (conhecidos por todos):**
   - Fp: corpo finito
   - Curva elíptica: $\boldsymbol{y^2 = x^3 + ax + b \bmod p}$
   - Ponto base **G** na curva.
2. **Alice:**
   - Gera chave privada a
   - Calcula chave pública: $\boldsymbol{A = a \ x \ G}$
   - Envia **A** para **Bob**
3. **Bob:**
   - Gera chave privada **b**
   - Calcula chave pública: $\boldsymbol{B = b \ x \ G}$
   - Envia B para **Alice**
4. **Ambos calculam a mesma chave secreta:**

- Alice: $\boldsymbol{K = a\ x\ B = a\ x\ (b\ x\ G)}$
- Bob: $\boldsymbol{K = b\ x\ A = b\ x\ (a\ x\ G)}$
- Resultado: $\boldsymbol{K = a.b\ x\ G}$

Como o cálculo inverso (descobrir **a** ou **b**) é difícil, o segredo está protegido.

3.3 Esquema de Assinatura Digital ECDSA

ECDSA (Elliptic Curve Digital Signature Algorithm) é a versão de curvas elípticas do algoritmo de assinatura digital (semelhante ao DSA).

Permite:

- Garantir **autenticidade** (quem enviou).
- Garantir **integridade** (não foi modificado).
- Prevenir **repúdio** (não pode negar que assinou).

Como funciona?

1. **Geração de chaves:**
   - Usuário escolhe chave privada **d**
   - Calcula chave pública $\boldsymbol{Q = d\ x\ G}$
2. **Assinatura de uma mensagem mmm:**
   - Gera número aleatório **k**
   - Calcula $\boldsymbol{R = k\ x\ G}$, usa coordenada $\boldsymbol{Xr}$
   - Calcula assinatura: dois valores **(r, s)**
   - Envia **(m, r, s)**
3. **Verificação da assinatura:**
   - Verifica que **r** e **s** correspondem à chave pública **Q**
   - Se a verificação for bem-sucedida → assinatura válida

Se $\boldsymbol{k}$ or reutilizado ou fraco, a chave privada pode ser descoberta! Isso já aconteceu em ataques a carteiras Bitcoin mal implementadas.

3.4 Comapração entre ECC e RSA

Característica	ECC	RSA
Segurança base	ECDLP	Factoração de primos
Tamanho da chave para 128 bits de segurança	**256 bits**	**3072 bits**
Velocidade (assinatura e verificação)	Rápida	Lenta
Tamanho da assinatura	Pequena	Grande
Ideal para	Dispositivos móveis, IoT	Servidores, arquivos grandes

ECC é mais eficiente, mais leve e mais moderno que RSA. Por isso, está sendo amplamente adotado em:

- WhatsApp, Signal, iMessage.
- Certificados TLS (HTTPS modernos).
- Bitcoin, Ethereum e criptomoedas.
- Dispositivos embarcados e chips.

5. 4. IMPLEMENTAÇÃO EM PYTHON

Parte 4 – Implementação em Python

A seguir, são apresentados os principais trechos da implementação do projeto, que foi desenvolvido em Python, com estrutura modular e entrada interativa de dados. A implementação respeita os fundamentos matemáticos das curvas elípticas sobre corpos finitos Fp e demonstra seu uso em criptografia, incluindo o protocolo ECDH e a assinatura digital ECDSA.

1. Entrada de Dados

```
def get_curve_parameters():
    """Lê os parâmetros da curva elíptica do usuário."""
    p = int(input("Digite o valor de p (primo): "))
    a = int(input("Digite o valor de a: "))
    b = int(input("Digite o valor de b: "))
    return p, a, b
```

Esta função solicita ao usuário os parâmetros da curva elíptica na forma $\mathbf{y^2 = x^3 + ax + b}$ **mod p.**

2. Adição de Pontos

```
def point_add(P, Q, a, p):
    """Soma dois pontos P e Q na curva elíptica sobre Fp."""
    if P is None: return Q
    if Q is None: return P
    x1, y1 = P
    x2, y2 = Q
    if x1 == x2 and y1 != y2:
        return None  # P + (-P) = ponto no infinito
    if P != Q:
        m = (y2 - y1) * pow(x2 - x1, -1, p) % p
    else:
        m = (3 * x1**2 + a) * pow(2 * y1, -1, p) % p
    x3 = (m**2 - x1 - x2) % p
    y3 = (m * (x1 - x3) - y1) % p
    return (x3, y3)
```

Implementa a operação de adição de dois pontos em uma curva elíptica sobre **Fp.**

4. Multiplicação Escalar (Double and Add)

```
def scalar_mult(k, P, a, p):
    """Multiplica o ponto P por um escalar k usando o método
double-and-add."""
    R = None
    N = P
    while k > 0:
        if k & 1:
            R = point_add(R, N, a, p)
        N = point_add(N, N, a, p)
        k >>= 1
    return R
```

Calcula k * P de forma eficiente, fundamental para operações **criptográficas seguras.**

4. Protocolo ECDH (Elliptic Curve Diffie-Hellman)

```
def ecdh(p, a, b, G, privA, privB):
    """Simula o protocolo ECDH entre Alice e Bob."""
    pubA = scalar_mult(privA, G, a, p)
    pubB = scalar_mult(privB, G, a, p)
    sharedA = scalar_mult(privA, pubB, a, p)
    sharedB = scalar_mult(privB, pubA, a, p)
    return pubA, pubB, sharedA, sharedB
```

Permite que duas partes troquem chaves secretas com segurança usando curvas elípticas.

5. Assinatura Digital com ECDSA (via biblioteca)

```
from ecdsa import SigningKey, SECP256k1

def run_ecdsa_demo():
    message = input("Digite a mensagem a ser assinada: 
").encode()
    sk = SigningKey.generate(curve=SECP256k1)
    vk = sk.verifying_key
    signature = sk.sign(message)
    print(f"Assinatura: {signature.hex()}")
    if vk.verify(signature, message):
        print(" Assinatura válida.")
    else:
        print(" Assinatura inválida.")
```

Demonstra a assinatura e verificação de uma mensagem usando ECDSA e a curva SECP256k1.

6. Resultado Esperado (ECDH)

Exemplo de saída após executar a opção do menu para simulação do protocolo ECDH:

```
Chave pública de Alice: (154, 4379)
Chave pública de Bob:   (217, 893)
Chave compartilhada de Alice: (7934, 3342)
Chave compartilhada de Bob:   (7934, 3342)
Sucesso: As chaves compartilhadas coincidem!
```

```
====== MENU - CRIPTOGRAFIA COM CURVAS ELÍPTICAS(Jofre - 20230394)======
1. Adição de dois pontos (P + Q)
2. Multiplicação escalar de um ponto (n * P)
3. Simulação do protocolo ECDH
4. Assinatura digital com ECDSA (lib ecdsa)
0. Sair
```

No final temos um meno como este em cima, para fazer todos os testes pretendidos.

[Ver o codigo completo apartir do meu Github]

6. Parte 5 – Análise e Discussão

5.1 Complexidade Computacional das Operações

As principais operações realizadas em curvas elípticas sobre corpos finitos **Fp** são:

- **Adição de Pontos:** Executada em tempo constante, depende de algumas operações modulares (adição, multiplicação, inverso modular).
- **Multiplicação Escalar (k * P):** Utiliza o algoritmo **double-and-add**, com complexidade **logarítmica** em relação ao escalar kkk, ou seja:

$$O(\log k)$$

  Isso torna a operação eficiente mesmo com valores grandes de **k** (como 256 bits).

A eficiência dessa operação é a base da segurança e desempenho da criptografia **ECC**.

5.2 Vulnerabilidades e Ataques Conhecidos

Embora a ECC seja muito segura, alguns cuidados são essenciais:

Ataques Teóricos

- **Ataque ao ECDLP:** Até hoje, não existe algoritmo eficiente (nem com computadores quânticos conhecidos) para resolver o **ECDLP** em curvas bem escolhidas.
- **Ataques com Baby-Step Giant-Step ou Pollard's Rho:** Possíveis para curvas pequenas ou chaves curtas (usadas apenas para fins didáticos).

Ataques Práticos e de Implementação

- **Reutilização de `k` no ECDSA:** Se o número aleatório usado na assinatura (k) for reutilizado ou mal gerado, a **chave privada pode ser descoberta**. Isso já aconteceu em falhas de carteiras de Bitcoin.
- **Ataques por canal lateral:** Observar o tempo de execução, consumo de energia ou radiação eletromagnética durante operações criptográficas para extrair chaves.

A escolha segura de curvas e boas práticas de implementação são fundamentais.

5.3 Comparação entre Curvas: NIST P-256 vs. Curve25519

Característica	NIST P-256	Curve25519
Segurança	128 bits	128 bits
Eficiência	Boa	Excelente (mais rápida)
Implementação	Propensa a erros	Projetada para evitar falhas
Origem	NIST (EUA)	Daniel J. Bernstein (independente)
Uso comum	Certificados SSL/TLS, governos	Signal, WhatsApp, Tor, criptomoedas
Resistência a ataques	Boa, mas com suspeitas de backdoors	Forte, com foco em segurança prática

A curva **Curve25519** é preferida atualmente em aplicações modernas pela sua eficiência, segurança prática e facilidade de implementação segura. Já a **NIST P-256** ainda é muito utilizada por compatibilidade e padronização governamental.

7. CONCLUSÃO E REFLEXÃO

Conclusão

A implementação de operações com curvas elípticas em **Python** proporcionou uma compreensão prática dos fundamentos matemáticos por trás da criptografia moderna. Durante o desenvolvimento, foi possível observar a importância da precisão nas operações modulares, da escolha adequada dos parâmetros e da segurança na geração de chaves e números aleatórios.

Apesar da simplicidade visual de uma curva elíptica, os algoritmos envolvidos são sensíveis a falhas de implementação, o que reforça a necessidade de uma abordagem cuidadosa, mesmo ao trabalhar com bibliotecas externas como a `ecdsa`.

O projeto também demonstrou, por meio da simulação do protocolo **ECDH** e da assinatura digital **ECDSA**, como as curvas elípticas oferecem uma solução prática e eficiente para comunicação segura.

Reflexão

Curvas elípticas representam uma das maiores revoluções da criptografia moderna. Elas permitem obter o mesmo nível de segurança de algoritmos como **RSA**, mas com **chaves muito menores**, reduzindo o custo computacional e facilitando a aplicação em dispositivos com recursos limitados, como smartphones, sensores e sistemas embarcados.

Além disso, o estudo das curvas elípticas mostrou como a matemática abstrata — como teoria dos grupos e corpos finitos — tem aplicações concretas e essenciais no mundo digital, protegendo transações financeiras, mensagens privadas e identidades online.

A realização deste trabalho reforçou a importância da matemática discreta como base para a segurança da informação, além de incentivar o aprofundamento em temas avançados como criptografia pós-quântica, curvas alternativas (como **Curve25519**) e protocolos criptográficos modernos.