Proceeding Series of the Brazilian Society of Computational and Applied Mathematics, v. 12, n. 1, 2026.

Trabalho apresentado no XLIV CNMAC, Fundação Getulio Vargas - Rio de Janeiro - RJ, 2025.

Proceeding Series of the Brazilian Society of Computational and Applied Mathematics

Criptografia Baseada em Curvas Simétricas Rotacionais

Manolo R. Heredia[1], Cecilia O. Castro[2]
CPAQ-UFMS, Aquidauana, MS

Resumo. Investigamos a definição de uma estrutura algébrica sobre curvas simétricas rotacionais para criptografia, adaptando o protocolo de ElGamal. Propomos uma operação binária entre pontos da curva e justificamos matematicamente a escolha dos parâmetros para evitar autointerseções. Implementamos e testamos computacionalmente o método, explorando sua viabilidade como alternativa às abordagens tradicionais baseadas em RSA e curvas elípticas.

Palavras-chave. Curvas Simétricas Rotacionais, Criptografia, ElGamal, Logaritmo Discreto.

1 Introdução

A criptografia moderna fundamenta-se na complexidade computacional de problemas matemáticos, sendo utilizados esquemas como RSA e a criptografia baseada em curvas elípticas (ECC) [10, 13]. Além dessas abordagens, propostas alternativas exploram estruturas geométricas e algébricas, como toros algébricos [11], curvas hiperelípticas [8], curvas de Lissajous [9] e variações de curvas elípticas como as curvas de Edwards [4], curvas de Hessian [2] e curvas Doche–Icart–Kohel [3].

Neste trabalho, propomos um esquema criptográfico alternativo baseado em curvas simétricas rotacionais (CSR), curvas planares caracterizadas por propriedades geométricas de simetria e congruências modulares. O sistema proposto utiliza uma operação binária inspirada no esquema ElGamal [5], diretamente relacionada ao problema do logaritmo discreto no grupo multiplicativo finito $\mathbb{F}_p^*$. Logo, nossa abordagem herda a segurança de esquemas clássicos, com resistência comparável ao RSA-2048 e Diffie-Hellman-2048 frente ao ataque clássico conhecido (NFS-DL) [1, 10]. Assim como os esquemas mencionados, também é vulnerável ao algoritmo quântico de Shor [12].

Além da formulação teórica, validamos a proposta com uma implementação computacional, demonstrando consistência, desempenho e possível aplicabilidade prática em contextos como criptografia visual, dispositivos com hardware restrito e ensino de matemática aplicada. Os resultados computacionais confirmam o escalonamento esperado em função de d, e reforçam que, ao eliminar exponenciações em favor de somas modulares e tabelas pré-calculadas, o CSR é atraente em cenários como IoT, criptografia visual e ensino.

2 Curvas Simétricas Rotacionais e Suas Propriedades

A construção da curva descrita em [7] estabelece sua simetria rotacional. Para $M = 3$ temos a "curva misteriosa" e, generalizando essa construção para M rodas no plano complexo, temos

$$\gamma(t) = \sum_{j=1}^{M} r_j \exp(ia_j t), \tag{1}$$

onde $t \in [0, 2\pi)$, $M \geq 1$, $r_j > 0$ são os raios e $a_j \in \mathbb{Z}$ deve satisfazer $k \equiv a_j \pmod{m}$ para $k \in \mathbb{Z}$.

A curva γ é periódica com período 2π e, a cada $2\pi/m$, sofre uma rotação de $2k\pi/m$ radianos.

[1]manolo.r@ufms.br
[2]cecilia.o@ufms.br

Teorema 2.1. *Se m, k e a_j são inteiros tais que $k \equiv a_j \mod m$ para $j = 1, \ldots, M$, então, para todo $t \in [0, 2\pi]$, a curva (1) satisfaz:*

$$\gamma\left(\frac{2\pi}{m} + t\right) = \exp\left(ik\frac{2\pi}{m}\right)\gamma(t). \tag{2}$$

Como consequência, a seguinte definição caracteriza a simetria rotacional:

Definição 2.1. *Uma curva γ possui* ***simetria rotacional de ordem m*** *se existe um inteiro k, com $\gcd(m, k) = 1$, tal que*

$$\gamma\left(\frac{2\pi}{m} + t\right) = \Delta_{k\frac{2\pi}{m}}\gamma(t).$$

Também, a curva pode satisfazer a **simetria espelhada**, conforme [6]. Em notação matricial,

$$\gamma(-t) = \Delta_{2\theta}R_0\gamma(t) = R_\theta\gamma(t), \tag{3}$$
$$\gamma(-t) = R_0\gamma(t), \tag{4}$$

onde

$$R_\theta = \begin{bmatrix}\cos(2\theta) & \sin(2\theta)\\ \sin(2\theta) & -\cos(2\theta)\end{bmatrix} \quad \text{e} \quad \Delta_\theta = \begin{bmatrix}\cos(\theta) & -\sin(\theta)\\ \sin(\theta) & \cos(\theta)\end{bmatrix}. \tag{5}$$

As relações em (3) e (4) decorrem diretamente da multiplicação sucessiva das matrizes Δ_θ (rotação) e R_θ (reflexão), o que torna imediata a demonstração.

Isto é, R_θ representa uma reflexão em torno de uma reta r que passa pela origem com coeficiente angular θ, e Δ_θ é uma rotação anti-horária de ângulo θ em torno da origem.

3 Resultados

Seja γ uma curva com simetria rotacional de ordem m, conforme a Definição 2.1, e considere $t \in [0, 2\pi)$. Definimos $\gamma|_{(0,2\pi/m)}$ como a restrição da curva ao intervalo $(0, 2\pi/m)$. Mostramos que essa restrição é suficiente para descrever toda a curva. De fato, para $t \in (0, 2\pi/m)$, utilizando (2) e a propriedade de composição de rotações $\Delta_{\theta_1}\Delta_{\theta_2} = \Delta_{\theta_1+\theta_2}$, verifica-se que, para $j = 1, 2, \ldots, m-1$,

$$\gamma\left(j\frac{2\pi}{m} + t\right) = \Delta_{jk\frac{2\pi}{m}}\gamma(t). \tag{6}$$

Assim, para $s \in (0, 2\pi)$, temos $s = j\frac{2\pi}{m} + t$, com $j\frac{2\pi}{m} < s < (j+1)\frac{2\pi}{m}$. Substituindo em (6)

$$\gamma(s) = \Delta_{jk\frac{2\pi}{m}}\gamma(t). \tag{7}$$

Essa relação permite reconstruir γ a partir da sua restrição ao intervalo $(0, 2\pi/m)$. A seguir, determinamos condições sob as quais γ pode ser gerada a partir da restrição ao intervalo $(0, \pi/m)$.

Proposição 3.1. *Se m, k e a_j são inteiros tais que $k \equiv a_j \mod m$ para $j = 1, \ldots, M$, e $\gcd(m, k) = 1$, então a curva definida em (1) satisfaz, para cada $t \in [0, 2\pi)$ e para todo $t \in [0, \pi/m)$ com $j = 1, \ldots, 2m-1$,*

$$\gamma\left(\frac{2\pi}{m} - t\right) = R_{k\frac{\pi}{m}}\gamma(t) \quad \text{e} \quad \gamma\left(j\frac{\pi}{m} + t\right) = R_{jk\frac{\pi}{m}}\gamma\left(j\frac{\pi}{m} - t\right), \tag{8}$$

respectivamente.

DOI: 10.5540/03.2026.012.01.0270

Demonstração. Pela equação (4), $\gamma(-t) = R_0\gamma(t)$, e pelo Teorema 2.1, temos:

$$\gamma\left(\frac{2\pi}{m} - t\right) = \Delta_{k\frac{2\pi}{m}}\gamma(-t). \tag{9}$$

Usando $\Delta_{2\theta}R_0 = R_\theta$, segue que $\Delta_{k\frac{2\pi}{m}}R_0 = R_{k\frac{\pi}{m}}$, e, substituindo em (9), obtemos:

$$\gamma\left(\frac{2\pi}{m} - t\right) = R_{k\frac{\pi}{m}}\gamma(t). \tag{10}$$

Para $j = 1$ e para $j \geq 2$, aplicando a equação anterior em $s = \frac{\pi}{m} - t$ e aplicando a identidade $R_{2\theta}R_\theta = \Delta_{2\theta}$ e iterando a relação, segue que

$$\gamma\left(\frac{\pi}{m} + t\right) = R_{k\frac{\pi}{m}}\gamma\left(\frac{\pi}{m} - t\right) \quad \text{e} \quad \gamma\left(j\frac{\pi}{m} + t\right) = R_{jk\frac{\pi}{m}}\gamma\left(j\frac{\pi}{m} - t\right), \tag{11}$$

respectivamente. Concluímos a demonstração. □

Definição 3.1. *Uma curva γ é **simétrica reflexiva** em relação à reta $y = \tan(jk\pi/m)x$ se, para cada $j = 1, \ldots, 2m-1$, a imagem do ponto refletido sobre essa reta pertence à curva. Ou seja,*

$$\gamma\left(j\frac{\pi}{m} + t\right) = R_{jk\frac{\pi}{m}}\gamma\left(j\frac{\pi}{m} - t\right). \tag{12}$$

Com base na Proposição 3.1, a curva pode ser gerada iterativamente a partir de reflexões sucessivas, conforme descrito no Algoritmo 1.

Algoritmo 1: Geração de Curvas Parametrizadas Usando Reflexões

Dados: Constantes M, m e k. Curva parametrizada γ_1.
Resultado: $\gamma = \gamma_1 \cup \gamma_2 \cup \ldots \cup \gamma_{2m-1} \cup \gamma_{2m}$.

```
γ|_[0,π/m) ← γ_1
para j ← 1 to 2m − 1 faça
    Defina a reta L_j por y = tan(jk π/m)x;
    Reflita γ_j em relação a L_j para obter γ_{j+1};
    γ|_((j π/m,(j+1) π/m)) ← γ_{j+1}
```

3.1 Construção dos Pontos na Curva

Apresentamos a construção dos pontos Q_j usados na cifragem e decifragem de mensagens. Os pontos Q_j são construídos explicitamente a partir dos pontos P_j e P_{j+1}, distribuídos em torno de um círculo unitário. A posição desses pontos P_j é derivada das potências de $g \pmod p$,

$$P_j = \left(\cos\left((g^j \mod p)\frac{2\pi}{p}\right), \sin\left((g^j \mod p)\frac{2\pi}{p}\right)\right), \tag{13}$$

para $j \mod (p-1)$. Com os pontos P_j e P_{j+1}, o ponto Q_j é dado por:

$$Q_j = P_j + \frac{\lambda}{g+1}\left(P_{j+1} - P_j\right), \tag{14}$$

onde $\lambda \in \mathbb{R}$ é um parâmetro que ajusta a posição de Q_j ao longo do segmento entre P_j e P_{j+1}.

A parametrização da curva $\varphi : [0, 2\pi] \mapsto \mathbb{R}^2$ é dada por

$$\varphi(t) = \left(\left(1 - \frac{\lambda}{g+1}\right)\cos t + \frac{\lambda}{g+1}\cos(gt), \left(1 - \frac{\lambda}{g+1}\right)\sin t + \frac{\lambda}{g+1}\sin(gt)\right) \tag{15}$$

onde λ é um parâmetro ajustável. Observamos que, se $t_j = (g^j \pmod p)(2\pi/p)$, então $\varphi(t_j) = Q_j$. Dessa maneira, para entender as propriedades deste conjunto de pontos, devemos estudar a curva φ.

DOI: 10.5540/03.2026.012.01.0270

3.2 Critério para Evitar Autointerseções

Considerando que a curva φ é simétrica reflexiva, a matriz da Definição 3.1 está dada por $R_{j\theta}$, onde $\theta = \pi/(g-1)$. Em particular, para $j = 1$, os valores próprios são 1 e -1 e os vetores próprios associados estão gerados por $(1, \tan(\theta))$ e $(1, -\cot(\theta))$, respectivamente.

Como consequência, possíveis autointerseções devem ocorrer ao longo das direções de simetria da curva. Além disso, segundo [6], uma CSR é invariante sob reflexão em relação ao eixo x, garantindo que interseções também devem ocorrer ao longo da reta $y = 0$.

Proposição 3.2. *Se $\varphi : [0, 2\pi] \to \mathbb{R}^2$ é uma curva parametrizada por (15) e $\theta = \pi/(g-1)$, para que a curva φ não possua autointerseções, é necessário que λ satisfaça a condição $\lambda \in I_g$, onde*

$$I_g = \left(-\frac{g+1}{g-1}, 1 \right). \tag{16}$$

Demonstração. Para evitar autointerseções, analisamos as interseções da curva φ com as retas $y = \tan(\theta)x$ e $y = 0$, onde $\theta = \pi/(g-1)$. Consideramos $t = \theta$, pois $g\theta = \pi + \theta$ implica:

$$(x(\theta), y(\theta)) = \left(\left(1 - \frac{2\lambda}{g-1} \right) \cos(\theta), \left(1 - \frac{2\lambda}{g-1} \right) \sin(\theta) \right), \tag{17}$$

o que confirma que $\varphi(\theta)$ pertence a $y = \tan(\theta)x$.

Outras possíveis interseções são determinadas pelo comportamento da função

$$\lambda(t) = (g+1) \frac{\sin t - \tan(\theta)\cos t}{\sin t - \tan(\theta)\cos t - (\sin(gt) - \tan(\theta)\cos(gt))}. \tag{18}$$

Os valores de t que anulam o denominador são:

$$t_k = \frac{2\theta}{g+1} + \frac{(2k+1)\pi}{g+1}, \quad t_m = -2m\theta. \tag{19}$$

Se $g \geq 3$, então $t_k = \theta$ é o único em $(0, 2\theta)$.

Para a reta $y = 0$, consideramos

$$\lambda_0(t) = (g+1) \frac{\sin t}{\sin t - \sin(gt)}, \tag{20}$$

com descontinuidades em $t_k = ((2k+1)\pi)/(g+1)$ e $t_m = -2m\theta$. No intervalo $(-\alpha, \alpha)$ com $\alpha = \pi/(g+1)$, avaliamos λ_0 em $t = 0$ usando L'Hôpital: $\lambda_0(t)$ se aproxima de $-(g+1)/(g-1)$ quando t tende a 0. Como $\lambda_0'(0) = 0$ e $\lambda_0''(0) < 0$, concluímos que $t = 0$ é um **máximo local**. Para $\lambda(\theta)$, obtemos que $\lambda(t)$ se aproxima de 1 quando t se aproxima de θ.

Com $\lambda'(\theta) = 0$ e $\lambda''(\theta) \geq 0$, segue que $t = \theta$ é um **mínimo local**. Assim, $\lambda \in I_g$. □

3.3 Definição de uma Estrutura Algébrica sobre os Pontos da Curva

Os pontos P_j podem ser considerados em qualquer círculo de raio $R > 0$, levando à definição:

$$\hat{P}_j = R \left(\cos\left((g^j \mod p) \frac{2\pi}{p} \right), \sin\left((g^j \mod p) \frac{2\pi}{p} \right) \right), \tag{21}$$

para $j \mod (p-1)$. Definimos o ponto $\hat{Q}_j$ como:

$$\hat{Q}_j = \hat{P}_j + \frac{\lambda}{g+1} \left(\hat{P}_{j+1} - \hat{P}_j \right), \tag{22}$$

e a parametrização da curva $\hat{\varphi} : [0, 2\pi] \to \mathbb{R}^2$ por $\hat{\varphi}(t) = R\varphi(t)$. Logo, se $t_j = (g^j \pmod p)(2\pi/p)$, vale $\hat{\varphi}(t_j) = \hat{Q}_j$, garantindo que as propriedades que evitam autointerseções de $\hat{\varphi}$ sejam preservadas.

Para definir uma estrutura algébrica sobre a curva, escolhemos os seguintes parâmetros:

$$\lambda_j = \ell + \frac{j(u-\ell)}{p-1}, \quad R_j = \left(\frac{1 + cj^2 \mod p}{x\left(t_j \mod (p-1)\right)^2 + y\left(t_j \mod (p-1)\right)^2} \right)^{1/2}, \tag{23}$$

onde $c \in \mathbb{Z}_{p-1}^*$, $\ell = -(g+1)(g-1)+\epsilon$, $u = 1-\epsilon$ e $\epsilon > 0$. Também, definimos as operações binárias:

$$\lambda_j * \lambda_k = \lambda_{j+k \mod (p-1)}, \quad R_j * R_k = R_{j+k \mod (p-1)}. \tag{24}$$

Finalmente, definimos a operação sobre os pontos $\hat{Q}_j$ da curva $\hat{\varphi}$ por

$$Q_j * Q_k = Q_{(j+k) \mod (p-1)}. \tag{25}$$

Verificamos que essa estrutura satisfaz as propriedades de um grupo abeliano.

3.4 Método de Cifragem e Decifragem Baseado na Curva

A nossa proposta de criptografia baseada em CSR segue um esquema adaptado do protocolo de ElGamal, estruturado sobre operações geométricas.

3.4.1 Processo de Cifragem

1. **Definição de parâmetros:** escolhe-se: um número primo p, define-se um número $g \mod p$, que pode ser uma raiz primitiva de p; uma chave privada j_{priv} e uma chave aleatória k.
2. **Codificação da mensagem:** A mensagem M é representada na base p por meio de
$$M = q_d p^d + n_{d-1} p^{d-1} + \cdots + n_1 p + n_0, \tag{26}$$
onde n_i são os restos sucessivos da divisão de M por p, e q_d é o último quociente obtido.
3. **Cálculo dos pontos cifrados:** Para cada índice j associado a um coeficiente da mensagem, calculam-se: o parâmetro angular t_j; o parâmetro de ajuste λ_j com $\epsilon = 1/(g-1)$; as coordenadas do ponto $Q_j = (x_j, y_j)$; o fator de escala R_j, com $c = g^{k \cdot j_{\text{priv}}} \pmod p$. Os pontos cifrados são então obtidos pela transformação:
$$x_{\text{enc}} = R_{j+k\cdot j_{\text{priv}}} \cdot x_{j+k\cdot j_{\text{priv}}}, \quad y_{\text{enc}} = R_{j+k\cdot j_{\text{priv}}} \cdot y_{j+k\cdot j_{\text{priv}}}. \tag{27}$$
Um erro associado à cifragem é calculado para verificação na decifragem:
$$\text{erro}_{\text{enc}} = x_{\text{enc}}^2 + y_{\text{enc}}^2 - \left(1 + g^{k\cdot j_{\text{priv}}}(j + k \cdot j_{\text{priv}})^2 \pmod p\right). \tag{28}$$
4. **Envio da mensagem cifrada:** Para cada índice j, a terna $(x_{\text{enc}}, y_{\text{enc}}, \text{erro}_{\text{enc}})$ é enviada como mensagem cifrada.

3.4.2 Processo de Decifragem

1. **Cálculo do valor j:** O valor de j é recuperado resolvendo:
$$r = \frac{x_{\text{enc}}^2 + y_{\text{enc}}^2 - (1 + \text{erro}_{\text{enc}})}{g^{k\cdot j_{\text{priv}}} \mod p} \mod p, \quad \text{e} \quad (j + k \cdot j_{\text{priv}})^2 = r \pmod p. \tag{29}$$
Logo, a raiz correta é escolhida minimizando o erro entre y_j e y_{enc}.
2. **Reconstrução da mensagem M:** Com os coeficientes j recuperados, a mensagem original é reconstruída usando (26).

DOI: 10.5540/03.2026.012.01.0270

3.5 Desenvolvimento Computacional e Discussões

Para validar a proposta, implementamos todo o esquema em Python no Google Colab. As bibliotecas usadas foram `mpmath`, `sympy`, `secrets` e `pycryptodome`. Os primos p com

$$p = \text{nextprime}(10^d - 1)\,, \tag{30}$$

foram gerados em célula separada do Colab e não foram incluídos nas medições de desempenho.

Testamos três tamanhos de mensagem: curta (12 C), média (49 C) e longa (137 C), onde C denota um caractere. Para cada combinação (mensagem, d), executamos 10 repetições da cifragem e da decifragem, coletando média e variância dos tempos de execução das funções puras de cifra/decifra.

Tabela 1: Tempo de cifragem/decifragem CSR para diferentes dígitos de p (10 execuções).

Dígitos	**Mensagem (C)**	**Tempo (s)**	
		Cifra (média/var)	Decifra (média/var)
101	12	0,0156 / 5 e-6	0,0490 / 6 e-6
101	49	0,0321 / 3,5 e-5	0,0961 / 6 e-6
101	137	0,0892 / 5,7 e-4	0,2836 / 5,4 e-3
301	12	0,0270 / 0	0,1417 / 6,6 e-5
301	49	0,0414 / 8,4 e-5	0,2013 / 2,8 e-3
301	137	0,0545 / 2 e-6	0,2744 / 7,2 e-5
601	12	0,0864 / 3,3 e-5	0,3186 / 5,2 e-5
601	49	0,0843 / 5 e-6	0,3223 / 6,1 e-5
601	137	0,1077 / 5,2 e-4	0,3919 / 7,2 e-3

A Tabela 2 mostra os tempos médios e variâncias de: CSR, `encrypt_indices`/`decrypt_indices`; RSA-2048, `rsa_encrypt`/`rsa_decrypt` (OAEP+AES-GCM); ECC-P256, `ecc_encrypt`/`ecc_decrypt` (ECDH+HKDF+AES-GCM).

Tabela 2: Comparativo de desempenho: CSR-601 díg. vs. RSA-2048 vs. ECC-P256 (10 execuções).

Algoritmo	**Mensagem (C)**	**Tempo (s)**	
		Cifra (média/var)	Decifra (média/var)
CSR-601 díg.	12	0,0864 / 3,3 e-5	0,3186 / 5,2 e-5
	49	0,0843 / 5 e-6	0,3223 / 6,1 e-5
	137	0,1077 / 5,2 e-4	0,3919 / 7,2 e-3
RSA-2048	12	0,0085 / 1,9 e-5	0,0091 / 5 e-6
	49	0,0077 / 5,6 e-6	0,0101 / 1,3 e-5
	137	0,0068 / 0	0,0086 / 1 e-6
ECC-P256	12	0,0101 / 0	0,0078 / 2,8 e-6
	49	0,0109 / 2,1 e-6	0,0074 / 0
	137	0,0103 / 0,2 e-6	0,0079 / 1,9 e-6

Nos testes, medimos apenas os tempos de cifra e decifra, sem incluir geração de chaves, inicialização de bibliotecas ou I/O. Teoricamente, cada operação CSR é $O(\log p)$, igual ao ECC e melhor que o RSA ($O((\log p)^3)$), e inverter $\star$ equivale ao DLP em $\mathbb{F}_p^*$ com complexidade $L_p[1/3, 1.923]$ para primos de 600 dígitos (≈ 112 bits de segurança), equiparável a RSA-2048 e DH-2048.

Também, o CSR pode gerar máscaras $M(x, y)$ por somas modulares e consulta a tabelas trigonométricas, o que o torna indicado para criptografia visual, para dispositivos IoT/embarcados

DOI: 10.5540/03.2026.012.01.0270

(ocupando apenas $\approx$ 8 kB de memória para tabelas) e para ensino de criptografia aplicada, ao ilustrar simetria rotacional e álgebra de grupos sem depender de bibliotecas de precisão arbitrária.

4 Considerações Finais

O CSR é um esquema baseado em curvas simétricas rotacionais cuja operação $\star$ é isomorfa a $(\mathbb{Z}/(p-1)\mathbb{Z}, +)$ e cuja segurança se reduz ao DLP em $\mathbb{F}_p^*$, com esforço $L_p[1/3, 1.923]$ em primos de 600 dígitos ($\approx$ 112 bits). Em Python, seu tempo de decifragem cresce linearmente de $\approx 0,05$ s a $\approx 0,32$ s (101 a 601 dígitos), enquanto RSA-2048 e ECC-P256 mantêm $\approx$0,007–0,010 s, mas com escalabilidade cúbica e constantes maiores, respectivamente.

A implementação atual carece de otimizações de baixo nível e não contempla geração de chaves ou ataques de canal lateral ou em contexto quântico, além do algoritmo de Shor, o que seria corrigido portando módulos críticos para C/GMP (`gmpy2`).

Referências

[1] E. Barker e A. Roginsky. **Transitioning the Use of Cryptographic Algorithms and Key Lengths**. Rel. técn. NIST Special Publication 800-131A Revision 2, 2019. URL: `https://doi.org/10.6028/NIST.SP.800-131Ar2`.

[2] D. J. Bernstein e T. Lange. "Faster Addition and Doubling on Elliptic Curves". Em: **International Conference on the Theory and Application of Cryptology and Information Security**. Springer, 2007, pp. 29–50.

[3] C. Doche, T. Icart e D. R. Kohel. "Efficient scalar multiplication by isogeny decompositions". Em: **Public Key Cryptography – PKC 2006, LNCS** 3958 (2006), pp. 191–206.

[4] H. M. Edwards. "A normal form for elliptic curves". Em: **Bulletin of the American Mathematical Society** 44.3 (2007), pp. 393–422.

[5] T. ElGamal. "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms". Em: **IEEE Transactions on Information Theory** 31.4 (1985), pp. 469–472.

[6] F. A. Farris. **Creating Symmetry: The Artful Mathematics of Wallpaper Patterns**. Princeton University Press, 2015.

[7] F. A. Farris. "Wheels on Wheels on Wheels-Surprising Symmetry". Em: **Mathematics Magazine** 69.3 (1996), pp. 185–189.

[8] N. Koblitz. "Hyperelliptic cryptosystems". Em: **Journal of Cryptology** 1.3 (1989), pp. 139–150.

[9] S. K. B. J. Kumar e K. Vijay. "Symmetric Key based Encryption and Decryption using Lissajous Curve Equations". Em: **International Journal of Electrical and Computer Engineering (IJECE)** 7.4 (2017), pp. 2173–2180. DOI: `10.11591/ijece.v7i4.pp2173-2180`.

[10] A. J. Menezes, P. C. van Oorschot e S. A. Vanstone. **Handbook of Applied Cryptography**. Boca Raton, FL, USA: CRC Press, 1996. ISBN: 0849385237.

[11] K. Rubin e A. Silverberg. "Torus-based cryptography". Em: **Lecture Notes in Computer Science** 3357 (2004), pp. 349–365.

[12] P. W. Shor. "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer". Em: **SIAM Review** 41.2 (1999), pp. 303–332.

[13] J. H. Silverman. **The Arithmetic of Elliptic Curves**. 2nd. Springer, 2009.

