Capítulo 01

Uma Abordagem Educacional da Criptografia Híbrida de Textos Via Decomposição LU com Geradores Pseudoaleatórios

An Educational Approach to Hybrid Text Cryptography via LU Decomposition with Pseudorandom Generators

Joilson Batista de Almeida Rego

ECT/UFRN.

Resumo: Este trabalho apresenta uma abordagem educacional aplicada à criptografia híbrida de textos por meio da decomposição LU associada a geradores pseudoaleatórios. O objetivo principal consiste em demonstrar a aplicabilidade da fatoração matricial em algoritmos de segurança, articulando conceitos de álgebra linear, criptografia e pensamento computacional em um contexto de formação acadêmica. A pesquisa caracteriza-se como aplicada, exploratória e experimental, sendo desenvolvida por meio da implementação de um protótipo em Python 3.10+ com apoio da biblioteca NumPy. O processo proposto contempla a conversão de textos em estruturas matriciais, sua decomposição em componentes triangulares e a posterior aplicação de procedimentos de criptografia e decriptografia. Os resultados obtidos indicam que a abordagem apresenta viabilidade computacional, preserva a integridade dos dados na etapa de recuperação e amplia a complexidade estrutural do processo criptográfico quando comparada a modelos lineares estáticos. Conclui-se que a proposta, além de funcional do ponto de vista técnico, constitui um recurso pedagógico relevante para a contextualização prática de conteúdos matemáticos e computacionais aplicados à segurança digital.

Palavras-chave: criptografia; decomposição LU; cifra de Hill; álgebra linear; Python.

Abstract: This study presents an educational approach to hybrid cryptography for text using LU decomposition combined with pseudo-random generators. The main objective is to demonstrate the practical applicability of matrix factorization in security algorithms while integrating concepts from linear algebra, cryptography, and computational thinking in an academic context. The research is characterized as applied, exploratory, and experimental, with the development of a computational prototype in Python 3.10+ using the NumPy library. The proposed procedure includes the conversion of text data into matrix structures, their decomposition into lower and upper triangular components, and the subsequent application of encryption and decryption operations. The results indicate that the approach is computationally viable, preserves data integrity during recovery, and increases the structural complexity of the cryptographic process when compared with static linear models. It is concluded that the proposed model is not only technically functional but also pedagogically relevant for the practical teaching of mathematical and computational concepts applied to digital security.

Keywords: cryptography; LU decomposition; Hill cipher; linear algebra; Python.

DOI: 10.47573/aya.5379.3.51.1

INTRODUÇÃO

A necessidade de resolver sistemas de equações lineares aparece em uma grande gama de aplicações na engenharia e matemática aplicada. Na álgebra linear computacional, os métodos de resolução direta conduzem a soluções exatas e com menor erro de arredondamento introduzido pela máquina. Neste contexto, apresenta-se uma aplicação prática de um método computacional em que é possível decompor uma matriz quadrada no produto de duas matrizes: uma triangular inferior (L) e outra triangular superior (U). A decomposição LU é tradicionalmente utilizada para resolver sistemas de equações lineares de forma eficiente. Em criptografia e segurança, ela ganha destaque na aplicação em esquemas de encriptação de textos, onde os dados são organizados em estruturas matriciais decompostas em L e U para criptografia separada, reduzindo o comprimento da cifra e o risco de interceptação. Essa abordagem combina eficiência numérica e computacional com segurança, apresentando inovações em segurança cibernética.

O objetivo geral deste trabalho é demonstrar a aplicação da decomposição LU em algoritmos de criptografia de textos, explorando sua capacidade de fatorar matrizes em componentes separáveis para otimizar a criptografia, reduzir o comprimento da cifra e aprimorar a segurança contra interceptações e ataques cibernéticos. Para tanto, alguns objetivos específicos se fazem necessários: apresentar os fundamentos teóricos da decomposição LU e implementar um protótipo computacional em Python.

Essa metodologia não apenas motiva a pesquisa autônoma sobre o tema, mas também permite a contextualização de conteúdos matemáticos fundamentais — como álgebra linear, teoria dos números e funções — com a realidade cotidiana dos estudantes, promovendo maior engajamento e retenção de conceitos. Além disso, ao simular cenários práticos de codificação e decodificação, a criptografia transforma abstrações matemáticas em problemas reais e solucionáveis, fortalecendo habilidades como raciocínio lógico, resolução de problemas e pensamento computacional.

O problema prático na intersecção entre álgebra linear computacional e algoritmos de criptografia de textos refere-se à necessidade de algoritmos que sejam eficientes e seguros na transmissão de dados em canais públicos ou não confiáveis. Em outros termos: como a decomposição LU pode ser aplicada para fatorar matrizes de texto, otimizando o processo de criptografia, reduzindo o comprimento de cifras e aprimorando a segurança contra intercepções e ataques?

Apresenta-se um trabalho de natureza aplicada e exploratória, com abordagem experimental, visando validar modelos e a viabilidade do método proposto no processo educacional de incentivar os alunos a criptografar textos por meio de algoritmos computacionais implementados em ambiente Python com a biblioteca *NumPy*. O estudo está estruturado da seguinte forma: fundamentação teórica, implementação computacional contemplando o processo de criptografia/decriptografia e, por último, a avaliação dos resultados obtidos.

FUNDAMENTAÇÃO TEÓRICA

No âmbito educacional, a criptografia revela-se uma ferramenta valiosa no processo de ensino-aprendizagem da Matemática, ao despertar o interesse e a curiosidade dos alunos por meio de abordagens didáticas, lúdicas e desafiadoras. Nas disciplinas de álgebra linear e computação numérica, a decomposição LU representa uma fatoração fundamental, em que uma matriz quadrada **A** é expressa como o produto de uma matriz triangular inferior **L** (com 1s na diagonal principal) e uma matriz triangular superior **U**. Essa técnica deriva diretamente da eliminação gaussiana e é amplamente utilizada para resolver sistemas de equações lineares de forma eficiente, evitando recálculos em múltiplos vetores do lado direito.

Considerando o sistema linear na forma:

$$\mathbf{Ax = b} \tag{1}$$

Onde **A** é a matriz do sistema (quadrada e de ordem n × n), **x** é o vetor de parâmetros desconhecidos (dimensão n × 1) e **b** é o vetor solução (dimensão n × 1), o método da decomposição LU consiste em fatorar a matriz **A** como o produto de duas matrizes:

$$\mathbf{A = LU} \tag{2}$$

De modo que **L** seja triangular inferior e **U** seja triangular superior. O sistema resultante reduz-se a:

$$\mathbf{Ax = b} \rightarrow \mathbf{LU = b} \rightarrow \underbrace{LUx}_{y} = b \tag{3}$$

Ou seja, o objetivo é decompor a matriz **A** em duas matrizes triangulares. Por exemplo:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix} \tag{4}$$

A resolução ocorre em duas etapas: substituição progressiva (**Ly = b**) e substituição regressiva (**Ux = y**), permitindo a obtenção eficiente da solução **x**. Tomando o exemplo apresentado, mostraremos como é efetuado o cálculo da decomposição LU, onde a matriz **A**, pode ser escrito como o seguinte produto matricial (Chapra, 2008),

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} u_{11} & u_{12} & u_{1} \\ l_{21}u_{11} & l_{21}u_{12}+u_{22} & l_{21}u_{13} \\ l_{31}u_{11} & l_{31}u_{12}+l_{32}u_{22} & l_{31}u_{13}+l_{32} \end{pmatrix} \tag{5}$$

Daí, podemos tirar as seguintes relações,

$$\begin{gathered} a_{1n}=u_{1n}, n=1,2,3 \\ a_{21}=l_{21}.u_{11} \rightarrow l_{21}=\frac{a_{21}}{u_{11}} \\ a_{22}=l_{21}.u_{12}+u_{22} \rightarrow u_{22}=a_{22}-l_{21}.u_{12} \\ a_{23}=l_{21}.u_{13}+u_{23} \rightarrow u_{23}=a_{23}-l_{21}.u_{13} \end{gathered} \tag{6}$$

e,

$$a_{31}=l_{31}.u_{11} \rightarrow l_{31}=\frac{a_{31}}{u_{11}}$$
$$a_{32}=l_{31}.u_{12}+l_{32}.u_{22} \rightarrow l_{32}=\frac{a_{32}-l_{31}.u_{12}}{u_{22}} \quad (7)$$
$$a_{33}=l_{31}.u_{13}+l_{32}.u_{23}+u_{33} \rightarrow u_{33}=(a_{33}-l_{31}.u_{13})-l_{32}.$$

Cifra de Hill

A cifra de Hill, proposta por Lester S. Hill em 1929, é um método de criptografia simétrica fundamentado em álgebra linear, no qual blocos de texto são representados por vetores numéricos e transformados por meio da multiplicação por uma matriz-chave em aritmética modular. Esse método amplia a lógica das cifras monoalfabéticas ao permitir a codificação simultânea de múltiplos caracteres, o que aumenta a complexidade da transformação aplicada à mensagem original (Hill, 1929).

Em termos formais, considerando uma matriz de texto plano **T**, uma matriz-chave **K** e um módulo *m*, a cifra pode ser expressa por **C = KT** mod *m*, em que **C** representa o texto cifrado. Para que o processo seja reversível, a matriz **K** deve possuir inversa no sistema modular adotado, o que exige que MDC(det(**K**), *m*) = 1.

Apesar de sua relevância histórica e de seu valor didático, a cifra de Hill tradicional apresenta limitações importantes do ponto de vista da segurança, especialmente em razão do uso de uma matriz-chave fixa, o que pode favorecer a ocorrência de padrões repetidos no texto cifrado e torná-la vulnerável a ataques de texto conhecido. Nesse sentido, sua utilização neste trabalho não se restringe à apresentação de um modelo clássico, mas serve como base conceitual para uma abordagem ampliada, em que a decomposição LU é incorporada como estratégia de organização matricial e de aumento da complexidade estrutural do procedimento criptográfico (Stallings,2015).

METODOLOGIA

O presente estudo caracteriza-se como aplicado, exploratório e de abordagem experimental, tendo como propósito investigar a viabilidade do uso da decomposição LU em procedimentos de criptografia de textos. O estudo foi desenvolvido com ênfase tanto no desempenho computacional do método quanto em seu potencial pedagógico, considerando a integração entre álgebra linear, criptografia clássica e implementação computacional em ambiente acadêmico.

Para os experimentos com texto, adotou-se um alfabeto expandido com 29 símbolos, incluindo letras, espaço e caracteres adicionais, permitindo a codificação de mensagens em vetores numéricos. Em seguida, os dados foram organizados em blocos compatíveis com a ordem da matriz-chave, de modo a possibilitar a aplicação

de operações inspiradas na cifra de Hill, porém associadas à decomposição LU como mecanismo adicional de estruturação e segurança. A escolha das matrizes de chave considerou a necessidade de invertibilidade no sistema modular adotado, garantindo a possibilidade de decriptografia e validação do processo (Rego, 2014).

A seguir, apresenta-se um exemplo didático de utilização da cifra de Hill para criptografar o texto “Resolver sistema”. Para m = 29 símbolos (letras, espaço e caracteres especiais),

Passo 01 — Codificação do texto.

A = 1	B = 2	C = 3	D = 4	E = 5	F = 6	G = 7	H = 8	I = 9	J = 10	K = 11	L = 12
M = 13	N = 14	O = 15	P = 16	Q = 17	R = 18	S = 19	T = 20	U = 21	V = 22	W = 23	X = 24
Y = 25	Z = 26	Espaço = 27	# = 28	@ = 0							

Para o texto escolhido, temos que: resolver o sistema equivale ao seguinte conjunto numérico – 18, 5, 19, 15, 12, 22, 5, 18, 27, 19, 9, 19, 20, 5, 13, 1, organizado na forma matricial **T**.

(8)

Passo 02 — Escolha da matriz-chave K

Escolhe-se, então, uma matriz-chave K tal que MDC(det(**K**), *m*) = 1, garantindo a existência de $\mathbf{K}^{-1}$ mod *m*.

$$K=\begin{pmatrix}1 & 1 & 1 & 1\\ 2 & 4 & 8 & 16\\ 3 & 9 & 27 & 81\\ 4 & 16 & 64 & 256\end{pmatrix} \tag{9}$$

Passo 03 — Cifração: C = (K·T)(mod m)

O texto cifrado C é obtido pelo produto matricial C = KT mod m. Ou seja, um produto matricial seguido de uma redução modular com m = 29.

$$C=\begin{pmatrix}1 & 1 & 1 & 1\\ 2 & 4 & 8 & 16\\ 3 & 9 & 27 & 81\\ 4 & 16 & 64 & 256\end{pmatrix}=\begin{pmatrix}18 & 5 & 19 & 15\\ 12 & 22 & 5 & 18\\ 27 & 19 & 9 & 19\\ 20 & 5 & 13 & 1\end{pmatrix}=\begin{pmatrix}77 & 51 & 46 & 53\\ 620 & 330 & 338 & 270\\ 2511 & 1131 & 1398 & 801\\ 7112 & 2868 & 4060 & 1820\end{pmatrix}_{mod29}$$

$$C=\begin{array}{c}77=2x29+19\\ 620=21x29+11\\ 2511=86x29+17\\ \vdots\\ 1820=62x29+22.\end{array}\begin{pmatrix}19 & 22 & 17 & 24\\ 11 & 11 & 19 & 9\\ 17 & 0 & 6 & 18\\ 7 & 26 & 0 & 22\end{pmatrix}=\begin{pmatrix}S & V & Q & X\\ K & K & S & I\\ Q & @ & F & R\\ G & Z & @ & V\end{pmatrix} \tag{10}$$

O texto cifrado resultante é: SVQXKKSIQ@FRGZ@V.

Passo 04 — Decomposição LU da matriz K

A matriz-chave **K** é decomposta em **K = LU**, onde **L** é triangular inferior (com 1’s na diagonal) e **U** é triangular superior, obtidas via equações 5, 6 e 7:

$$K=\begin{pmatrix}1 & 0 & 0 & 0\\ 2 & 1 & 0 & 0\\ 3 & 3 & 1 & 0\\ 4 & 6 & 4 & 1\end{pmatrix}\begin{pmatrix}1 & 1 & 1 & 1\\ 0 & 2 & 6 & 14\\ 0 & 0 & 6 & 36\\ 0 & 0 & 0 & 24\end{pmatrix} \tag{11}$$

Passo 05 — Criptografar o texto cifrado por meio de: $X = (L^{-1} \cdot C)(\text{mod } m)$

Aplica-se a inversa da matriz **L** sobre o texto cifrado **C** para gerar a matriz (texto) criptografada **X**:

$$X=\begin{pmatrix}1&0&0&0\\-2&1&0&0\\3&-3&1&0\\-4&6&-4&1\end{pmatrix}\begin{pmatrix}19&22&17&24\\11&11&19&9\\17&0&6&18\\7&26&0&22\end{pmatrix}=\begin{pmatrix}19&22&17&24\\-27&-33&-15&-39\\41&33&0&63\\-71&4&22&-99\end{pmatrix}_{mod29}$$
$$=\begin{pmatrix}19&22&17&24\\11&11&19&9\\17&0&6&18\\7&26&0&22\end{pmatrix}=\begin{pmatrix}S&V&Q&X\\B&Y&N&S\\L&D&@&E\\P&D&V&X\end{pmatrix}\rightarrow SVQXBYNSLD@EPDVX. \quad (12)$$

Portanto, SVQXBYNSLD@EPDVX corresponde efetivamente ao texto criptografado.

Antes de mostrar o processo inverso (decriptografia), faz-se necessário o entendimento da matriz inversa modular. Sabendo que o determinante da matriz **K** corresponde ao produto dos autovalores da matriz **K**, onde os mesmos correspondem aos elementos da diagonal na matriz **U**, temos que:

$$det(K) = \prod_{i=1}^{n} u_{ii} \quad (13)$$

E o *MDC*(det(**K**),*m*) = 1. De modo a garantir que a matriz **U** possua inversa modular (módulo *m*). No exemplo apresentado, temos que det(**K**) = 288, assim, *MDC*(288, 29) = 1. Portanto, a matriz **U** é invertível módulo 29. Calcular,

$$det(K)det(K)^{-1} \equiv 1(mod m) \rightarrow 288x \equiv 1(mod 29). \quad (14)$$

Onde x corresponde ao inverso multiplicativo modular de 288. Assim,

$$\begin{gathered} x = 1 \rightarrow 288 = 1x29 + 259 \\ x = 2 \rightarrow 288.2 = 976 = 33x29 + 19 \\ \vdots \\ x = 14 \rightarrow 288.14 = 4032 = 139x29 + 1 \end{gathered} . \quad (15)$$

Passo 06 — Decriptografia: $T_0 = (U^{-1} \cdot X)(\text{mod } m)$

Para a obtenção (recuperação) do texto original, calcula-se $\mathbf{U}^{-1}$ por meio de $\mathbf{U}^{-1} = (\det(\mathbf{K}))^{-1} \cdot \text{adj}(\mathbf{U})$ (mod m), onde adj(**U**) corresponde a matriz adjunta de U.

$$U^{-1}=\begin{pmatrix}4032&-2016&96&1344\\-1008&0&2016&-2016\\0&0&672&-1008\\0&0&0&168\end{pmatrix}_{mod29}=\begin{pmatrix}1&14&10&7\\0&15&14&21\\0&0&5&7\\0&0&0&23\end{pmatrix} \quad (16)$$

De modo que o texto original é recuperado por $\mathbf{T}_0 = \mathbf{U}^{-1} \cdot \mathbf{X}$ (mod 29):

$$T_0=\begin{pmatrix}1&14&10&7\\0&15&14&21\\0&0&5&7\\0&0&0&23\end{pmatrix}\begin{pmatrix}19&22&17&24\\2&25&14&19\\12&4&0&5\\16&4&22&24\end{pmatrix}=\begin{pmatrix}279&440&367&508\\534&515&672&859\\172&48&154&193\\368&93&506&552\end{pmatrix}_{mod19}$$

$$=\begin{pmatrix}18&5&19&15\\12&22&5&18\\27&19&9&19\\20&5&23&2\end{pmatrix}=\begin{pmatrix}R&E&S&O\\L&V&E&R\\\square&S&I&S\\T&E&M&A\end{pmatrix}\rightarrow RESOLVERSISTEMA.$$

Que corresponde ao texto original (Rego, 2014). Portanto, a avaliação metodológica foi organizada em quatro eixos principais: consistência matemática, funcionalidade computacional, integridade da recuperação dos dados e potencial didático. A consistência matemática foi verificada pela correta fatoração das matrizes e pela manutenção das condições necessárias para a reversibilidade do método. A funcionalidade computacional foi observada por meio da execução do protótipo em diferentes entradas de texto, considerando estabilidade, tempo de processamento e capacidade de generalização. A integridade dos dados foi analisada pela comparação entre os conteúdos originais e os resultados obtidos após o processo de inverso (decriptografia). Por fim, o potencial didático foi considerado a partir da possibilidade de empregar o algoritmo como recurso de ensino para demonstrar, de forma aplicada, conceitos abstratos de álgebra linear, computação numérica e segurança da informação.

RESULTADOS E DISCUSSÃO

Os resultados obtidos a partir da implementação do protótipo computacional indicam que a decomposição LU apresenta desempenho satisfatório como base estruturante de um procedimento criptográfico aplicado a textos. Em termos funcionais, verificou-se que a fatoração da matriz original em componentes triangulares inferiores e superiores permitiu organizar o processamento dos dados de forma sistemática, favorecendo tanto a etapa de encriptação quanto a posterior recuperação da informação original no processo de decriptografia. Esse comportamento confirma a viabilidade matemática e computacional do método proposto no contexto do estudo.

Nos experimentos com dados textuais, a conversão da mensagem para representação numérica e sua organização em blocos matriciais compatíveis com a chave adotada possibilitaram a aplicação do algoritmo de forma estável e reprodutível. A estrutura matricial empregada mostrou-se adequada para demonstrar, de maneira didática, como operações de álgebra linear podem ser integradas a procedimentos de codificação, especialmente quando associadas à lógica da cifra de Hill e ao uso de critérios de invertibilidade modular. Além disso, a combinação entre organização em blocos e fatoração matricial contribuiu para reduzir a exposição direta a padrões simples da mensagem original, aspecto relevante do ponto de vista criptográfico.

Do ponto de vista comparativo, a abordagem proposta apresenta vantagens conceituais em relação à cifra de Hill tradicional, especialmente no que se refere

à rigidez estrutural associada ao uso de uma matriz-chave fixa. Enquanto a cifra clássica tende a manter uma transformação linear única para diferentes blocos de dados, o emprego da decomposição LU introduz uma camada adicional de organização matricial que amplia a complexidade do processo e favorece a construção de estratégias mais flexíveis de encriptação. Embora isso não elimine, por si só, todas as vulnerabilidades inerentes a sistemas lineares, o procedimento proposto representa um avanço relevante em termos de modelagem didática e de sofisticação estrutural do algoritmo.

Os testes também sugerem que o método possui compatibilidade com aplicações educacionais, sobretudo em disciplinas que articulam matemática aplicada, programação e segurança da informação. A implementação em Python 3.10 com bibliotecas científicas amplamente utilizadas contribui para a reprodutibilidade do experimento e para sua incorporação em práticas de ensino baseadas em experimentação computacional. Nesse sentido, os resultados reforçam que o modelo não deve ser compreendido apenas como uma solução técnica, mas também como um recurso pedagógico capaz de aproximar o estudante de conceitos abstratos por meio de uma aplicação concreta e interdisciplinar.

Outro aspecto relevante observado diz respeito ao equilíbrio entre simplicidade de implementação e profundidade conceitual. A decomposição LU, por já integrar o repertório clássico da álgebra linear computacional, oferece uma entrada metodológica acessível para estudantes e pesquisadores iniciantes, ao mesmo tempo em que sustenta discussões avançadas sobre estabilidade algorítmica, invertibilidade, aritmética modular e representação matricial de dados. Dessa forma, a proposta contribui para consolidar uma ponte entre fundamentos matemáticos e aplicações contemporâneas em criptografia.

CONSIDERAÇÕES FINAIS

Os resultados obtidos confirmam que a decomposição LU se mostrou adequada como núcleo matemático de uma proposta de criptografia híbrida com finalidade técnica e educacional. O método evidenciou coerência interna entre fundamentação teórica, implementação computacional e aplicação prática, demonstrando que a fatoração matricial pode contribuir tanto para a organização do processo criptográfico quanto para o enriquecimento do ensino de álgebra linear e computação científica.

Apesar dos resultados favoráveis, é importante reconhecer que a presente investigação possui caráter exploratório e, portanto, não esgota as possibilidades de avaliação do método. Estudos futuros podem incorporar métricas quantitativas mais específicas, como tempo de processamento por dimensão matricial, análise de complexidade computacional, sensibilidade a perturbações numéricas e comparação com outros algoritmos criptográficos aplicados a textos. Também se mostra pertinente examinar, em trabalhos posteriores, o impacto do uso de diferentes geradores pseudoaleatórios e de esquemas dinâmicos de chave sobre a robustez global do sistema.

Em síntese, os achados sustentam a relevância da abordagem proposta para contextos de formação acadêmica e para investigações futuras na interface entre matemática aplicada, programação e segurança digital. A proposta se consolida como recurso pedagógico relevante para a contextualização prática de conteúdos matemáticos e computacionais aplicados à segurança da informação.

REFERÊNCIAS

HILL, L. S. **Cryptography in an algebraic alphabet**. American Mathematical Monthly, v. 36, n. 6, p. 306–312, 1929.

STALLINGS, W. **Criptografia e segurança de redes: princípios e práticas.** Tradução de Daniel Vieira. 6. ed. São Paulo: Pearson Education do Brasil, 2015.

CHAPRA, Steven C.; CANALE, Raymond P. **Métodos numéricos para engenharia.** 5. ed. São Paulo: McGraw-Hill, 2008.

REGO, Joilson B. A. **Computação numérica: notas de aula** [recurso eletrônico] / p. 124 – 128, Natal/RN: Caule de Papiro, 2024.

ANEXO A — CÓDIGO-FONTE PYTHON: CIFRA DE HILL COM DECOMPOSIÇÃO LU

O código a seguir implementa, em Python 3.10+, o algoritmo completo de criptografia e decriptografia da cifra de Hill com decomposição LU, conforme descrito nos Passos 01 a 06 da seção de Metodologia. Dependências: *NumPy* e math.

```
"""
Cifra de Hill com Decomposição LU
==================================
Exemplo didático: criptografia e decriptografia do texto "RESOLVER SISTEMA"
Usando a cifra de Hill associada à decomposição LU.
"""

import numpy as np
from math import gcd


#
# ============================================================================
=
# Configuração do alfabeto (m = 29 símbolos)
#
# ============================================================================
=

m = 29

ALPHA = {c: i + 1 for i, c in enumerate('ABCDEFGHIJKLMNOPQRSTUVWXYZ')}
ALPHA[' '] = 27
ALPHA['#'] = 28
ALPHA['@'] = 0
REV = {v: k for k, v in ALPHA.items()}


#
# ============================================================================
=
# Funções auxiliares
#
# ============================================================================
=

def extended_gcd(a, b):
    "Algoritmo de Euclides estendido.
    Retorna (g, x, y) tal que a*x + b*y = g = mdc(a, b)."""
    if a == 0:
        return b, 0, 1
    g, x, y = extended_gcd(b % a, a)
    return g, y - (b // a) * x, x


def mod_inv(a, m):
    """Inverso multiplicativo modular de a mod m."""
    g, x, _ = extended_gcd(a % m, m)
    if g != 1:
        raise ValueError(f"{a} não tem inverso multiplicativo mod {m}")
    return x % m


def lu_no_pivot(A):
    """Decomposição LU por eliminação gaussiana sem pivoteamento.
    Retorna L, U (inteiros) tais que A = L @ U (exato, sem permutação).

    Nota: scipy.linalg.lu retorna P, L, U onde P @ A = L @ U.
    Como P pode não ser a identidade, usamos esta implementação
    manual para garantir A = L @ U diretamente.
    """
    n = A.shape[0]
    L = np.eye(n)
    U = A.astype(float).copy()
    for k in range(n - 1):
```

```
        for i in range(k + 1, n):
            if U[k, k] == 0:
                raise ValueError("Pivô zero encontrado — pivoteamento
necessário.")
            L[i, k] = U[i, k] / U[k, k]
            U[i, :] -= L[i, k] * U[k, :]
    return np.round(L).astype(int), np.round(U).astype(int)


def mat_mod_inv_lower(L, m):
    """Inversa modular de matriz triangular inferior com 1s na diagonal.
    Usa substituição progressiva."""
    n = L.shape[0]
    inv = np.eye(n, dtype=int)
    for i in range(n):
        for j in range(i):
            inv[i] = (inv[i] - L[i, j] * inv[j]) % m
    return inv % m


def mat_to_text(M):
    """Converte matriz de volta ao texto, lendo linha por linha."""
    return ''.join(REV[M[i, j]] for i in range(M.shape[0]) for j in
range(M.shape[1]))


def print_matrix(name, M):
    print(f"{name} =")
    for row in M:
        print("  ", row)
    print()


#
================================================================================
=
# Passo 01 — Codificação do texto
#
================================================================================
=

print("=" * 60)
print("PASSO 01 — Codificação do texto")
print("=" * 60)

texto = "RESOLVER SISTEMA"
nums = [ALPHA[c] for c in texto]
print("Texto:    ", texto)
print("Sequência:", nums)

T = np.array(nums).reshape(4, 4)
print_matrix("\nT", T)


#
================================================================================
=
# Passo 02 — Matriz-chave K
#
================================================================================
=

print("=" * 60)
print("PASSO 02 — Matriz-chave K")
print("=" * 60)

K = np.array([
    [1,  1,  1,   1],
    [2,  4,  8,  16],
    [3,  9, 27,  81],
    [4, 16, 64, 256]
])
```

```
det_K = round(np.linalg.det(K))
print(f"det(K)        = {det_K}")
print(f"MDC({det_K}, {m}) = {gcd(det_K, m)}  → K é invertível mod {m}\n")
print_matrix("K", K)


#
================================================================================
=
# Passo 03 — Cifração: C = (K · T) mod m
#
================================================================================
=

print("=" * 60)
print("PASSO 03 — Cifração: C = (K · T) mod 29")
print("=" * 60)

C = (K @ T) % m
print_matrix("C", C)

cifrado = mat_to_text(C)
print(f"Texto cifrado: {cifrado}\n")


#
================================================================================
=
# Passo 04 — Decomposição LU de K
#
================================================================================
=

print("=" * 60)
print("PASSO 04 — Decomposição LU de K  (sem pivoteamento)")
print("=" * 60)

L, U = lu_no_pivot(K)
print_matrix("L", L)
print_matrix("U", U)
print(f"Verificação L @ U == K: {np.allclose(L @ U, K)}\n")


#
================================================================================
=
# Passo 05 — Criptografia adicional: X = (L⁻¹ · C) mod m
#
================================================================================
=

print("=" * 60)
print("PASSO 05 — Criptografia adicional: X = (L⁻¹ · C) mod 29")
print("=" * 60)

Linv = mat_mod_inv_lower(L, m)
X = (Linv @ C) % m
print_matrix("X", X)

cifrado2 = mat_to_text(X)
print(f"Texto criptografado final: {cifrado2}\n")


#
================================================================================
=
# Passo 06 — Decriptografia: T₀ = (U⁻¹ · X) mod m
#
================================================================================
=
```

```
print("=" * 60)
print("PASSO 06 — Decriptografia: T₀ = (U⁻¹ · X) mod 29")
print("=" * 60)

det_inv = mod_inv(det_K, m)
print(f"(det K)⁻¹ mod {m} = {det_inv}  "
      f"(pois {det_K} × {det_inv} = {det_K * det_inv} ≡ {(det_K * det_inv) %
m} mod {m})")

adj_U = np.round(det_K * np.linalg.inv(U)).astype(int)
Uinv = (det_inv * adj_U) % m
print_matrix("\nU⁻¹", Uinv)

T0 = (Uinv @ X) % m
print_matrix("T₀", T0)

original = mat_to_text(T0)
print(f"Texto original recuperado: {original}")
print(f"Correto: {original == texto}\n")
```