Documentație pentru programul

“Algoritmi de Criptare”

Cuprins

[Către utilizator 2](#_Toc199141860)

[Funcționalitate 2](#_Toc199141861)

[Feedback 2](#_Toc199141862)

[Către dezvoltatori 3](#_Toc199141863)

[Algoritmi implementați 3](#_Toc199141864)

[Structura proiectului 18](#_Toc199141865)

[Bibliografie 19](#_Toc199141866)

[Wikipedia 19](#_Toc199141867)

[Youtube 19](#_Toc199141868)

[Cryptii 19](#_Toc199141869)

[Diverse 19](#_Toc199141870)

# Către utilizator

## Funcționalitate

A screenshot of a computer

AI-generated content may be incorrect.Programul “Algoritmi de criptare” are scopul de a facilita utilizatorilor o interfață grafică prin care își pot cripta/decripta mesaje sau parole, într-un mod intuitiv și ușor. Câmpul text este locul unde se introduce mesajul care va urma să fie modificat. În funcție de butonul apăsat, modificările vor fi afișate în câmpul din partea dreaptă.

Fiecare algoritm are un câmp specific, de cele mai multe ori denumit cheie. Spre exemplu, cifrul Cezar este unul dintre cele mai cunoscute cifre datorită simplității sale. Cheia reprezintă numărul de poziții cu care se va schimba ordinea alfabetului. Litera “A” este prima din alfabet, dar dacă îi atribuim valoarea 2 câmpului cheie, prima litera din alfabet va fi “C”.

Pentru o detaliere completă a modului de funcționare al algoritmilor, puteți conspecta secțiunea “Algoritmi implementați”.

## Feedback

Dacă constatați vreo eroare sau doriți să faceți sugestii referitoare la noi funcționalități sau modificări, puteți să ne contactați la adresa: totallyLegitMail@gmail.com

# Către dezvoltatori

## Algoritmi implementați

|  |  |
| --- | --- |
| Algoritm | Mod de utilizare câmp cheie |
| Caesar | A fost explicat anterior |
| Vigenère | Asemănător cu Caesar, cheia reprezintă un șir de caractere. Fiecărei litere din mesaj îi este schimbată poziția cu ordinea literei corespondentului din cheie. |
| Polybius | Cheia reprezintă o matrice pătratică de ordinul 5 care poate fi extinsă la ordinul 6 dacă sunt necesare și numere sau chiar și la ordinul 7 pentru caractere speciale (pentru a avea litera “j” este nevoie de matricea de ordinul 6). Fiecărui element din acea matrice îi este asociată o pereche, prima cifră fiind rândul la care se află litera în matrice, iar a doua cifră este numărul coloanei. Matricea poate fi modificată, însă nu sunt permise caractere identice în matrice. De asemenea, poate fi apăsat butonul de resetare care aduce matricea la forma inițială (ordinea normală a literelor în alfabet) |
| Bifid | Se folosește de matricea lui Polybius pentru a genera valori numerice bazate pe lini și coloane. Se folosește la fel ca Polybius, însă pentru fiecare literă se formează un sir cu reprezentanții rândului iar apoi se concatenează reprezentanții coloanelor. După aceea, se ia grupe de câte două cifre și se reprezintă litera de pe linia si coloana grupei, prima cifră reprezintă linia, a doua reprezintă coloana. |
| ADFGVX | Cifrul se numește așa datorită faptului ca toate literele unui alfabet sunt puse într-o matrice (de tip Polybius) care proiectează fiecare element al matricei, într-o pereche linie/coloană (formate doar din ADFGVX). Aceste caractere au fost alese datorită distincției dintre ele în codul morse). Rezultă un mesaj fracționat fiecare făcând referire la linia/coloana în care se află în matrice. După care, se alege o cheie de lungime “n”, grupurile fiind așezate pe lini de lungime “n”. Cheia se reordonează în funcție de ordinea literelor din alfabet, astfel se reordonează și coloanele mesajului criptat. Într-un final se citesc elementele de pe coloane în ordine crescătoare. |
| SHA-1 | Textului îi este atribuit valorile binare ale literelor din tabela ASCII (corespondente cu mesajul introdus) după care, sunt introduse un bit cu valoarea 1 urmat de biți de 0 până când lungimea acelui mesaj este congruentă cu restul împărțiri lui 448 la 512, iar la final este introdusă valoarea binară a lungimii mesajului reprezentată pe 64 de biți. Se inițializează 5 variabile, (H0, H1, H2, H3, H4) fiecare de lungime de 32 biți pentru a furniza în final un mesaj de 160 de biți, valorile acestea sunt standard și constante:  H0 = 0x67452301, H1 = 0xEFCDAB89, H2 = 0x98BADCFE, H3 = 0x10325476, H4 = 0xC3D2E1F0.  Urmează aplicarea funcției de compresie, fiecare bloc va avea 80 de iterații. Pentru fiecare bloc de câte 512 biți se divid în câte 16 cuvinte de 32 de biți. De la iterația 16 până la 79 (presupunem ca începem iterațiile de la 0). Cuvintele se calculează astfel:    Unde simbolul dintre cuvinte reprezintă operatorul pe biți logic “sau exclusiv”, iar S la puterea 1 reprezintă mutare la stânga cu o poziție unde exponentul reprezintă blocul la care suntem. Se inițializează 5 variabile A, B, C, D, E cu valorile H0, H1, H2, H3, H4. Pentru 80 de iterații se folosește formula:    Înainte de a trece la pasul următor vom explica anumite aspecte ale ecuației:  f(i; B, C, D) este o funcție ce ia ca parametrii: i – iterația, cuvintele B C D,  atunci se aplică pe biți:  dacă i este între [0, 19] (B ∧ C) ∨ (¬ (B) ∧ D)  dacă i este între [20, 39] B ⊻ C ⊻ D  dacă i este între [40, 59] (B ∧ C) ∨ (B ∧ D) ∨ (C ∧ D)  dacă i este între [60, 79] B ⊻ C ⊻ D  de asemenea, în funcție de i, K ia valorile:  pentru i între [0, 19], K(i) = 5A827999;  pentru i între [20, 39], K(i) = 6ED9EBA1;  pentru i între [40, 59], K(i) = 8F1BBCDC​;  pentru i între [60, 70], K(i) = CA62C1D6,​.  După reatribuim valorile variabilelor: E = D, D = C = S30(B), B = A, A = TEMP.  După cele 80 de iterații ale blocului, însumăm variabilelor inițiale următoarele valori:  H0 = H0 + A;  H1 = H1 + B;  H2 = H2 + C;  H3 = H3 + D;  H4 = H4 + E.  În cazul în care avem mai multe blocuri de câte 512 biți, variabilele inițiale blocului curent devin cele de la blocul anterior după terminarea algoritmului de compresie.  De asemenea, adunarea este calculată prin modulo 232.  După ce au fost procesate toate blocurile de 512 biți, mesajul „digerat” format din 160 de biți ale variabilelor H le sunt aplicate următoarea formulă.  HH = S128(H0​) ∨ S96(H1​) ∨ S64(H2​) ∨ S32(H3​) ∨ H4​.  Se returnează HH. |
| SHA-2 | Familia aceasta de funcții conține SHA-256, SHA-512. Cea mai mare diferență dintre cele două familii de algoritmi de hashing este lungimea mesajului “digerat”. Pentru SHA-256, evident funcția va returna o valoare de 256 de biți, asemănător și pentru SHA-512, funcția returnează un șir format din 512 biți.  SHA-256:  Asemănător în multe aspecte cu SHA-1, mesajul trebuie să fie multiplu de 512 biți, în caz contrar se adaugă un 1 urmat de 0-uri până când lungimea mesajului este congruentă cu 448 modulo 512 pentru a avea 64 de biți pe care putem reprezenta lungimea mesajului inițial. Urmează inițializarea a 8 variabile (cu 3 mai multe decât la SHA-1) H0, H1, H2, H3, H4, H5, H6, H7 de 32 de biți fiecare având valorile părților fracționare ale rădăcinilor pătrate ale primelor 8 numere prime scrise în hexazecimal. Urmează inițializarea unei liste formate din 64 de valori (k[i], unde i ∈ [0, 63]) care reprezintă primele zecimale reprezentate pe 32 biți din partea fracționară a rădăcinii cubice a primelor 64 numere prime. Se divide mesajul în blocuri de 512 biți, fiecare bloc se divide în câte 64 de cuvinte de 32 de biți. Urmează bucla și aplicarea funcției de compresie. Primele 16 cuvinte sunt fragmentele blocului inițial și se extinde până la 64 de cuvinte astfel:  Se inițializează variabilele:  Rn reprezintă schimbare la dreapta cu n poziții, cu introducerea valorilor în “small endian”;  Sn reprezintă schimbare la stânga cu n poziții, cu introducerea valorilor în “small endian”;  rn schimbare la dreapta cu trunchierea valorilor scoase și introducerea unor 0-uri nesemnificative în stânga;  sn schimbare la stânga cu trunchierea valorilor scoase și introducerea unor 0-uri nesemnificative în dreapta.  s0 = (W[i-15] R7) ⊻ (W[i-15] R18) ⊻ (W[i-15] r3)  s1 = (W[i - 2] R17) ⊻ (W[i - 2] R19) ⊻ (W[i - 2] r10)  w[i] = w[i-16] + s0 + w[i-7] + s1  După ce au fost găsite toate cuvintele, se inițializează următoarele variabile astfel:  A = H0, B = H1, C = H2, D = H3, E = H4, F = H5, H = H6;  Începe bucla funcției de compresie care are 64 de iterații:  S1 = (E R6) ⊻ (E R11) ⊻ (e R25)  ch = (E ∧ F) ⊻ ((¬E) ∧ G)  temp1 = H + S1 + ch + k[i] + w[i]  S0 = (A R2) ⊻ (A R13) ⊻ (A R22)  maj = (A ∧ B) ⊻ (A ∧ C) ⊻ (B ∧ C)  temp2 = S0 + maj  Se reatribuie valorile variabilelor de la A la H:  H = G, G = F, E = D + temp1, D = C, C = B, B = A, A = temp1 + temp2  Se adună variabilelor inițiale în modulo 232:  H0 = H0 + A, H1 = H1 + B, H2 = H2 + C, H3 = H3 +D, H4 = H4 + E, H5 = H5 + F, H6 = H6 + G, H7 = H7 + H.  În cazul în care avem mai multe blocuri de câte 512 biți, variabilele inițiale blocului curent devin cele de la blocul anterior după terminarea algoritmului de compresie.  Se concatenează toate variabilele H și se returnează șirul.  SHA-512:  Este identic, din punct de vedere structural doar că are anumite aspecte diferite față de SHA-256.  Blocurile sunt formate din 1024 de biți, rezultă că padding-ul de 1 urmat de 0 trebuie să fie congruent cu restul împărțirii lui 896 la 1024, ultimi 128 de biți sunt folosiți pentru a reprezenta binar lungimea mesajului.  Adunarea variabilelor inițiale se petrece în contextului unui modulo 264.  Se inițializează variabilele de 64 de biți:  H0 = 0x6a09e667f3bcc908;  H1 = 0xbb67ae8584caa73b;  H2 = 0x3c6ef372fe94f82b;  H3 = 0xa54ff53a5f1d36f1;  H4 = 0x510e527fade682d1;  H5 = 0x9b05688c2b3e6c1f;  H6 = 0x1f83d9abfb41bd6b;  H7 = 0x5be0cd19137e2179.  Se inițializează o listă cu 80 de valori K[i] astfel:  A close-up of a computer code  AI-generated content may be incorrect.  Începe bucla pentru calcularea cuvintelor de 64 de biți și aplicarea funcției de compresie, asemănător cu cea de la SHA-256 doar că sunt extinse la 80 de iterați (SHA-256 avea doar 64). De la i >= 16 până la 79 cuvintele sunt calculate astfel:  s0 = (w[i-15] R1) ⊻ (w[i-15] R8) ⊻ (w[i-15] r7)  s1 = (w[i-2] R19) ⊻ (w[i-2] R61) ⊻ (w[i-2] r6)  W[i] = w[i-16] + s0 + w[i-7] + s1  Se inițializează variabilele de la A la H la fel ca la SHA-256.  Funcția de compresie urmează același algoritm cu cel al lui SHA-256 doar că S0 și S1 sunt diferite:  S0 = (a R28) ⊻ (a R34) ⊻ (a R39)  S1 = (e R14) ⊻ (e R18) ⊻ (e R41)  Restul operațiilor sunt identice cu cele de la SHA-256. |
| MD5 | Este la fel ca SHA-1, SHA-2, un algoritm de hashing prin care se introduce un mesaj de lungime n și iese un mesaj de 128 de biți. Se convertesc literele mesajului în numerele reprezentante fiecărui caracter din tabela ASCII. Bineînțeles, aceste valori vor fi reprezentate în binar. Același padding are loc ca și la algoritmii SHA-1 și SHA-256, iar ultimi 64 de biți sunt alocați pentru valoarea binară a lungimii mesajului.  Fiecare bloc este de câte 512 biți, și fiecare bloc este divizat în 16 cuvinte de 32 de biți. Se inițializează 4 vectori: A = 0x01234567, B = 0x89abcdef, C = 0xFEDCBA98, D = 0x76543210.  De asemenea se inițializează o listă K[i], în funcție de i, avem:  K[i] = |sin (i + 1)| x 232 pentru I ∈ [0, 63]  Se inițializează o listă de valori care reprezintă schimbare cu n poziți la stânga  Pentru 0 <= i <= 15:  s = 4 \* [7, 12, 17, 22] (acest șir se repetă de 4 ori și indicele reprezintă poziția numărului, de exemplu, dacă i = 0 => s[0] = 7, dacă i = 2 => s[2] = 17)  Pentru 16 <= i <=31:  s = 4 \* [5, 9, 14, 20]  Pentru 32 <= i <= 47:  s = 4 \* [4, 11, 16, 23]  Pentru 48 <= i <= 63:  s = 4 \* [6, 10, 15, 21]  Începe bucla care creează blocurile M[n], iar ulterior cuvintele (notate W[i]). Se inițializează variabilele a0, b0, c0, d0 care copiază valorile variabilelor inițiale A, B, C, D.  În funcție de numărul iterației, se va aplica o formulă specifică.  Pentru i ∈ [0, 63], dacă 0 <= i <= 15:  F = (B ∧ C) ∨ ((¬B) ∧ D)  g = i  dacă 16 <= i <= 31:  F = (D ∧ B) ∨ ((¬D) ∧C)  g = (5\*i + 1) mod 16  dacă 32 <= i <= 47:  F = B ⊻ C ⊻ D  g = (3\*i + 5) mod 16  dacă 48 <= i <= 63:  F = C ⊻ (B ∨ (¬D))  g = (7\*i) mod 16  “\*”- reprezintă înmulțire  Se adună la fiecare variabilă în funcție de iterație astfel:  F = F + a0 + K[i] + W[g]  a0 = d0  d0 = c0  c0 = b0  b0 = b0 + Ss[i] \* F  După toate cele 64 de iterații se atribuie următoarele valori variabilelor inițiale:  A = A + a0  B = B + b0  C = C + c0  D = D + d0  Toate adunările sunt mod 232.  În cazul în care avem mai multe blocuri de câte 512 biți, variabilele inițiale blocului curent devin cele de la blocul anterior.  După parcurgerea tuturor blocurilor se concatenează cele 4 variabile inițiale (A, B, C, D) și se returnează rezultatul. |
| AES | Algoritmul AES operează pe blocuri de 128 de biți, returnează mesajul digerat pe aceeași lungime de biți și are o cheie de mărimile 128, 192 sau 256 de biți (programul include doar chei de 128 sau 256 de biți) bazându-se pe substituție și permutație. Algoritmul se folosește de o matrice pătratică de ordinul 4, în care elementele mesajului de 128 de biți sunt așezate pe verticală. În funcție de lungimea cheii, sunt specificate numărul de iterații care vor avea loc ulterior: pentru 128 de biți vor avea loc 10 iterații, pentru 192 vor avea loc 12 iterații, iar pentru 156 vor avea loc 14 iterații.  Pentru a nu fi vulnerabil la atacurile diferențiale prin analiza intrărilor și a ieșirilor, o metodă de a sparge cheia, algoritmul se folosește de o coadă de așteptare a cheilor. Algoritmul funcționează în felul următor:  Se definesc următoarele variabile: K (cheia introdusă de utilizator de 128/192/256 de biți), Nk (numărul de cuvinte de 32 de biți) = 4/6/8, Nr (numărul de runde) =10/12/14, Nb (numărul de coloane a matricei) = 4/6/8. Urmează etapa expansiunii cheii care va genera 11 “subchei” pentru că prima cheie (plain text, introdusă de utilizator) îi este aplicată operația logică “sau exclusiv” înainte de a aplica un “key schedule” (coadă de chei), deci, practic înainte de a genera chei specifice iterației, prima cheie este operată prin sau exclusiv cu starea inițială (mesajul așezat pe verticală într-o matrice de dimensiune n) după care starea este supusă unor operațiuni ce vor fi prezentate după ce sunt explicați pașii prin care se obțin cheile specifice pentru fiecare iterație. Coada va genera chei de 32 de biți, iar w[i] reprezintă un cuvânt de 8 biți, unde 0 <= i <= 10. Fiecare cheie ulterioară depinde de o cheie anterioară astfel:  Pentru i > 3:  W[i] = W[I - Nk] ⊻ W[i - 1]  Același lucru se aplică pentru fiecare cheie, K0 =[w0, w1, w2, w3] => K1 format din w0, w1, w2, w3 = w4, w5, w6, w7 și tot așa de 10 ori.  Excepția apare atunci când i este congruent cu 0 mod Nk, adică atunci când trecem la o nouă cheie. În acest caz, aplicăm cuvântului anterior (cel din cheia anterioară, adică w3, de exemplu funcțiile: RotWord, SubWord și Rcon).  RotWord: preia valorile cheii anterioare și le aplică o rotație la stânga  SubWord: preia valorile cuvânt de 32 de biți și le aplică AES S-Box pentru a genera noi valori  AES-S Box (Rijndael S-box):  Reprezintă o tabelă de substituție care proiectează unei intrări de 8 biți o ieșire de opt biți, toate valorile din tabel sunt obținute printr-o formulă care aparține câmpului Galois: GF(28) = GF(2) [x]/(x8 + x4 + x3 + x + 1). Toate valorile se pot reprezenta pe un tabel.    Se calculează inversa operației multiplicative printr-o transformare afină:    Unde operația de adunare este operația logică sau exclusiv pe biți cu o constantă egală cu 99 scrisă în binar pe verticală în “big endian”, înmulțirea este matriceală, și rezultatul este un vector. Pentru decriptare se folosește inversa matricei înmulțită cu coloana vectorilor pentru a afla inversa operației multipliative, iar constanta este egală cu 5 scrisă în binar în “big endian”.  Practic are loc următoarea transformare: s = b ⊻ (S1 b) ⊻ (S2) ⊻ (S3) ⊻ (S4) ⊻ (01100011).  Rcon: este o listă de constante, din care se aleg în funcție de numărul iterației o valoare la care se adaugă încă 3 octeți cu valori 0 în hexazecimal pentru a forma un cuvânt de 32 de biți și care se aplică o operație logică sau exclusiv cuvântului obținut din rezultatele anterioare de forma rconi = [rci 00 00 00] reprezentat în hexazecimal    Astfel se generează toate cheile la fiecare iterație.  După ce a fost generată cheia iterației respective, urmează substituția biților după valorile tabelului S-Box menționat anterior și prin transformare afină.  Până acum am generat confuzie în algoritmul nostru, următorii pași vor reprezenta partea de difuzie a algoritmului.  Rândurile stări (matricei mesajului) sunt mutate în următorul mod:   * Primul rând rămâne neschimbat * Al doilea rând este schimbat la stânga prin rotație cu un bit. * Al treilea rând este schimbat la stânga prin rotație cu doi biți. * Al patrulea rând este schimbat la stânga prin rotație cu trei biți.   Coloanele stării sunt combinate prin intermediul unei transformări liniare inversabilă. Datorită faptului că valorile nu pot depăși valoarea de 255, operațiile vor avea loc, din nou, într-un câmp Galois.  Pentru fiecare coloană are loc următoarea transformare:  A black background with white text  AI-generated content may be incorrect.  Fiecare coloană introdusă din stare (a0j, a1j, a2j, a3j) are elemente coeficienți ale unui polinom de grad 7.  Înmulțirea este modulo polinomului ireductibil x8 + x4 + x3 + x2 + x + 1. Iar adunarea elementelor din matrice este operația logică sau exclusiv pe biți. În cazul în care valoare este peste 255 atunci se aplică o operație sau exclusiv condițională cu 27 reprezentat în binar.  Coloanele din matricea pătratică de ordinul 4 reprezintă coeficienții unui polinom constant, specific standardului AES.  După toate aceste operații se combină starea internă cu cheia respectivei iterații prin operația logica sau exclusiv pe biți.  În concluzie:   * Se introduce mesajul și o cheie care va facilita criptarea/decriptarea mesajului inițial/mesajului digerat; * Urmează procesul de expansiune a cheii; * Se adaugă prima cheie (introdusă de utilizator); * Acum începe bucla de n iterații (depinde de lungimea aleasă); * Se substituiesc biții; * Se schimbă rândurile; * Se combină coloanele; * La finalul fiecărei iterații se adaugă cheia iterației prin operația logică sau exclusiv.   De menționat este faptul că ultima iterație sare peste combinarea coloanelor, întrucât aceasta nu va avea un impact asupra mesajului digerat. |
| RC4 | Este un algoritm care înșiruie bit cu bit, într-un mod pseudo-aleatoriu, caractere care au suferit un proces de prelucrare simplu. De asemenea este simetric, adică, criptarea și decriptarea este bazată pe cheie, decriptarea aceluiași mesaj criptat cu aceeași cheie generează mesajul inițial. Are 2 părți, ca și la AES este prezent un “key schedule” care are loc la începutul algoritmului și criptarea/decriptarea propriu-zisă.  Se inițializează următoarele variabile: S[i] o listă de 256 de caractere cu valori de la 0 la 255 (starea), doi indici i și j, k – cheia, lenk – lungimea cheii inițiale ,K – cheia digerată.  1.Algoritmul de creare a cheii (key schedule):  În primul rând, lungimea unei chei trebuie să fie între 48 și 2048 de biți.  Se creează bucla de 256 de iterații care va introduce cele 256 de valori în stare. (se folosește indexul i). Urmează o buclă de același număr de iterații care va determina indicele j astfel: j = (j + S[i]) + k[i modulo lenk]) mod 256 și va schimba între ele valorile stării de la pozițiile S[i] și S[j], unde j și i sunt inițial egali cu 0.  2.Algoritmul de generare al mesajului digerat pseudo-aleatoriu:  De asemenea se inițializează indicii i și j cu valorile 0. Începe o nouă buclă care are n iterații în funcție de lungimea mesajului introdus unde.  i = (i + 1) mod 256  j = (j + S[i]) mod 256  Se schimbă valorile stării de la pozițiile S[i] și S[j] între ele.  t = (S[i] + S[j]) mod 256  K = S[t]  Și se afișează toți biți lui K până când mesajul digerat are lungimea mesajului inițial. |
| Playfair | Algoritmul se folosește de o matrice pătratică (pătrat Polybius) de ordinul 5 pentru a reprezenta litere din alfabet, pe rânduri. Se poate alege o cheie, al căror elemente nu se repetă, literele fiind așezate primele pe rânduri, după care se înșiruie în tabel literele în ordinea alfabetului fără a se repeta.  Pentru a cripta mesajul, se formează grupe de câte două litere. În cazul în care numărul de litere este impar se adaugă, un “X” la final, se poate alege orice literă care nu este folosită frecvent. Același lucru se întâmplă și în momentul în care un grup este format din aceeași literă, în acest caz, prima literă se scrie, iar a doua se transformă într-un “X”.  În funcție de poziția relativă a celor două litere din tabelul cheie, există 3 cazuri:  1.Dacă cele două litere se află pe același rând, atunci, litera cod este cea cu o poziție la dreapta (în cazul în care litera este la capăt de rând, se începe de la capătul celălalt cu primul element al rândului). În cazul decriptării, litera inițială este cu o poziție la stânga.  2.Dacă cele două litere se află pe aceeași coloană, poziția literei cod se află cu o poziție mai jos decât litera inițială. Pentru decriptare, se urcă cu o poziție pe verticală.  A group of letters on a white background  AI-generated content may be incorrect.3.Dacă cele două litere sunt colțul unui patrulater, atunci litera cod va fi litera aflată la colțul opus (pe orizontală), acest caz se aplică și la decriptare la fel. |
| Hill | Este un algoritm care are un input/output de aceeași lungime. Din mesajul inițial se fac grupe de câte n, unde n = 2 sau n = 3 litere, în funcție de caz. Dacă avem o matrice de ordinul 2, vom lua grupe de câte două litere, în cazul în care ordinul matricei este 3, grupele vor fi formate din 3 litere. În cazul în care, o grupare de litere nu are lungimea egală cu n, se poate adăuga oricare litere din alfabet până când lungimea este egală cu n.  Cheia poate va fi formată din 4 sau 9 litere care vor fi așezate într-o matrice pătratică de ordinul 2 sau 3 cu elementele așezate pe rânduri. Fiecare literă îi fi atribuită o valoare numerică echivalentă astfel. Matricea rezultantă trebuie să fie inversabilă pentru a putea decripta mesajul, asta înseamnă că determinantul acestea trebuie să fie nenul.  A screenshot of a computer  AI-generated content may be incorrect.  Fiecare grup (din mesajul inițial) se transformă într-un vector de 3 sau 2 elemente (în funcție de numărul de elemente al unui grup), în care, valorile numerice ale literelor sunt așezate pe verticală.  Pentru a obține codul, se înmulțește fiecare vector cu matricea cheie și se aplică mod 26. În cazul decriptării fiecare grup format din mesajul digerat se înmulțește cu inversa matricei și se aplică mod 26.  Inversa matricei se obține astfel:  Unde, det(K) este determinantul matricei K, iar adj(K) este adjuncta matricei.  Adjuncta matricei se calculează astfel:  Kt este matricea transpusă a lui K unde elementele de pe linii devin coloane  Unde, kij reprezintă elementele matricei adjuncte formate din matricea transpusă și det(Δ) este determinantul matricei transpuse unde linia i și coloana j nu sunt introduse. |
| Enigma | Istoria acestei capodopere inginerești nemțești își are originea nu de mult după apariția primelor aparate cu rotoare, în particular, invenția celor doi ofițeri de marină olandezi Theo van Hengel și Rudolf Spengler. În anul 1920 Arthur Scherbius și Ernst Richard Ritter, au fondat compania Scherbius & Ritter iar 3 ani mai târziu vor pune pe piață primele aparate de criptare/decriptare Enigma. Acestea au fost folosită de către naziști în Cel De-al Doilea Război Mondial.  Enigma era utilizat, în general, de 2 persoane, o persoană introducea mesajul inițial prin taste, iar cealaltă persoană copia literele afișate prin becuri pe o foaie de hârtie separată. Aparatul putea fi folosit pentru criptare și decriptare. Securitatea sistemului era dependentă de configurația aparatului care era schimbată zilnic.  Bateria aparatului se află în colțul din dreapta sus de unde pornește curentul electric și circuitul.  Enigma este constituită din 3 părți importante: rotoarele, tastatura și plugboard-ul.  Rotoarele    - fiecare Enigma are câte 3 rotoare  - un rotor are 26 de segmente pentru literele din alfabet A-Z.  - in interior, pe partea stângă, privind transversal, există 26 de puncte de contact din metal, în cealaltă parte sunt 26 de bare din metal.  - când două rotoare sunt conectate bara de metal merge în punctele de contact de metal, curentul electric poate circula printre cele două rotoare.  - în interiorul fiecărui rotor există fire care sunt întortocheate între ele (de la bară la punctul de contact) (ex pentru intrarea numărului unu iese ieșirea numărului 4)  - de fiecare dată când electricitatea traversează printr-un rotor, aceasta schimba litera o singură dată, în total sunt 3 rotoare toate cu configurații ale firelor diferite.  - după cele 3 rotoare este o parte numită reflector (și acesta are 26 puncte de contact)care schimba imput-ul primit după ce a fost schimbat de cele 3 rotoare și îl schimbă din nou. După ce reflectorul îl schimbă, curentul electric trece din nou prin cele 3 rotoare. Astfel intrarea inițială suferă 7 modificări.  - roata de intrare, pe acolo merge input-ul. (capătul opus reflectorului).(26 puncte de contact din metal).  - fiecare rotor se poate roti 26 de ori după fiecare apăsare a unei taste de pe tastatură, ceea ce determină o nouă cărare pe care curentul electric îl va parcurge.  Imaginea unui reflector  - rotorul din partea dreaptă este rotorul care se va roti de cele mai multe ori, acesta, trebuie să se rotească de 26 de ori pentru a roti o singură dată rotorul din mijloc, respectiv, pentru a roti o singură dată rotorul din stânga, rotorul din mijloc trebuie să se fi rotit de 26 de ori.  - acest mecanism al rotoarelor are drept consecință următorul lucru, aceeași literă apăsată de mai multe ori, are un output diferit de fiecare dată.  - metoda prin care aceste rotoare se rotesc este pur mecanică, nu necesită curent.  - sub circuit, chiar cel mai jos element al aparatului se află o placă de metal numită dispozitiv de acționare, funcționează ca și un balansoar, atunci când o tijă împinge dispozitivul în jos în partea dinspre tastatură, în partea opusă, dispozitivul este împins în sus.  - în partea împinsă în sus este un mecanism cu clichet, roata dințată este împinsă de către un clichet ce are forma unui cârlig, după care vine din nou în jos.  - fiecare apăsare a tastei va împinge roata dințată cu o poziție  - de cele mai multe ori, clichetul nu va putea avea contact cu rotorul din mijloc și cu cel din partea stângă, doar după ce primul rotor, cel din partea dreaptă s-a rotit de 26 de ori, prin simplul fapt că există o crestătură la un singur loc din cele 26 care permite clichetului să aibă contact cu rotorul din mijloc. Același mecanism are loc și la rotorul din stânga.  - fiecare rotor are o pârghie cu o roată indice conectată la roata dințată pentru a asigura că rotorul se învârte doar cu o singură poziție, de asemenea, acestea garantează alinierea exactă dintre punctele de contact din metal și barele de metal care facilitează conectarea firelor prin care curentul electric va trece.  Plugboard    - este în partea din față a aparatului, jos. Acesta este un comutator care poate schimba o literă cu alta, asemănător unei prize.  - aici vine input-ul de la rotoare  - în interior, fiecare cablu merge deasupra fiecărei prize care reprezintă o literă, și dedesubt-ul fiecărei prize  - în cazul în care priza unei litere este necuplată, curentul electric merge deasupra prizei, printr-o bară de scurtcircuitare, si ajunge dedesubt, astfel, curentul electric nu își schimbă calea (litera nu se schimbă)  - în cazul în care priza unei litere este cuplată cu o altă literă, bara de scurtcircuitare este împinsă și astfel nu are contact cu curentul electric. În acest caz, curentul merge prin partea de sus a prizei, după care prin cablul care face legătura dintre cele două prize, urmând ca, ieșirea curentului electric să fie prin partea de jos a celeilalte prize.  - de menționat este faptul că nu toate literele pot fi cuplate.  Mecanismul tastaturii  - fiecare tastă este conectată la o tijă dedesubtul ei.  - fiecare tijă are arcuri ce vor împinge tastele înapoi după ce nu mai este exercitată forța degetului.  - rândul din capătul de sus, are arcurile în partea de sus a barelor ce aparțin tastelor, nu pe tije.  - sub tije se află 26 de comutatoare, fiecare formate din 3 plăcuțe de cupru, curentul electric intră prin partea posterioară fiecărui comutator, iar curentul este controlat în partea anterioară.  - cele 3 plăcuțe sunt așezate una dedesubtul celeilalte fără a face contact cu toate 3 simultan, în mod implicit cele două plăcuțe de sus au contact, în cazul în care o tastă este apăsată, plăcuța este împinsă în jos și are contact cu plăcuța de jos.  - plăcuțele din partea de sus sunt conectate la câte un bec din cele 26 de deasupra tastaturii prin intermediul unor cabluri care merg prin partea stângă a aparatului.  - plăcuțele din mijloc sunt conectate la plugboard (placa de priză) prin intermediul unor cabluri care merg prin partea dreaptă a mașinii și prin fiecare loc corespondent al literei din plugboard.  - un cablu de la baterie merge direct prin partea de jos a aparatului, făcând contact cu fiecare plăcuță de jos.  - în modul implicit, curentul este blocat, și becurile nu pot fi aprinse, doar în momentul în care o tastă este apăsată este realizat un circuit iar becul se aprinde.  Un circuit arată în felul următor în momentul în care o tastă este apăsată:  - curentul electric circulă de la baterie prin partea de jos a aparatului și are contact cu plăcuța de jos, după ce este apăsată tasta, se realizează contactul dintre plăcuța de jos și cea din mijloc, care îndrumă curentul electric spre plugboard. În cazul în care litera nu este conectată la o altă literă, aceasta nu își schimbă calea și rămâne aceeași.  - ulterior, curentul merge prin partea laterală a aparatului și ajunge la roata de intrare.  - curentul trece printre 3 rotoare, reflector și încă o dată 3 rotoare (schimbând litera la fiecare popas)  - după care curentul trece din nou prin plugboard, iar litera se schimbă dacă litera care reprezintă ieșirea este conectată la o altă literă  - curentul ajunge încă o dată la o plăcuță de mijloc dar la o altă tastă (care este conectată la plăcuța de sus)  - curentul traversează cablul plăcuței de sus și ajunge într-un final la becul literei corespunzătoare.  - pentru a completa circuitul, curentul traversează prin placa de metal a becurilor iar în colț se află un fir care este conectat la baterie.  - în momentul în care o tastă nu mai este apăsată, nu mai există legătura dintre plăcuța din mijloc și plăcuța de sus care face legătura cu becurile și astfel are loc un scurtcircuit.  În total o literă se poate schimba în jur de 7-9 ori.  Pentru a putea folosi mașina Enigma, trebuie configurată cu setările adecvate înainte de utilizare.  În primul rând este ordinea rotoarelor, fiecare aparat enigma vine la pachet cu 5 rotoare, dintre cele 5 se pot alege 3, și se alege ordinea în care sunt puse (care va fi rotorul din dreapta, mijloc și stânga), urmează modul în care fiecare inel este configurat (poziția de început a rotorului), fapt care modifică momentul în care rotoarele din partea stângă pot fi rotite. Se poate configura și poziția de început a rotoarelor, iar ultima setare este plugboard-ul.  Dacă emițătorul și receptorul au aceleași configurații, doar atunci poate fi decriptat mesajul. Chiar dacă o altă persoană are un aparat Enigma, dacă nu are aceiași configurație, atunci, mesajul nu va fi decriptat corect. |

## Structura proiectului

Proiectul este structurat pe mai multe niveluri.

În directorul cu numele Proiect-Criptografie există următoarea structură: bin (fișierele executabilul), directorul Pachete, Licența, fișierul main (cel care rulează programul). Directorul pachete este are diferite subdirectoare fiecare axându-se pe o parte din cod.

Algoritmi

* Secțiunea unde au fost implementați toți algoritmii.

Conector

* Secțiunea unde este implementată validarea pentru algoritmi

Interfață

* Secțiunea unde a fost realizată interfața grafică a programului cu ajutorul pachetului Tkinter.

Teste

* Implementarea unui test pentru cifrul lui Caesar

Utilități

* Secțiunea unde au fost introduse funcții care calculează determinați și minorul unei matrice.

# Bibliografie

## Wikipedia

* <https://en.wikipedia.org/wiki/Caesar_cipher>
* <https://en.wikipedia.org/wiki/Vigen%C3%A8re_cipher>
* <https://en.wikipedia.org/wiki/Polybius_square>
* <https://en.wikipedia.org/wiki/Bifid_cipher>
* <https://en.wikipedia.org/wiki/ADFGVX_cipher>
* <https://en.wikipedia.org/wiki/SHA-1>
* <https://en.wikipedia.org/wiki/SHA-2>
* <https://en.wikipedia.org/wiki/MD5>
* <https://en.wikipedia.org/wiki/Advanced_Encryption_Standard>
* <https://en.wikipedia.org/wiki/RC4>
* <https://en.wikipedia.org/wiki/Playfair_cipher>
* <https://en.wikipedia.org/wiki/Hill_cipher>
* <https://en.wikipedia.org/wiki/Enigma_machine>

## Youtube

* <https://www.youtube.com/watch?v=ybkkiGtJmkM> – Enigma
* <https://www.youtube.com/watch?v=O4xNJsjtN6E> – AES
* <https://www.youtube.com/watch?v=C4ATDMIz5wc> – AES
* <https://www.youtube.com/watch?v=-KjFbTK1IIw> – Playfair
* <https://www.youtube.com/watch?v=mNGj_skw7Ck> – Hill
* <https://www.youtube.com/watch?v=DMtFhACPnTY&t=95s> - SHA-1
* <https://www.youtube.com/watch?v=JKSkxW2lf8E&list=PLh6pbpXpUUNFG-CJPZj-IPdXSdCK1G813&index=7> – Determinanți

## Cryptii

* <https://cryptii.com/>

## Diverse

* <https://en.wikipedia.org/wiki/Arthur_Scherbius>
* <https://de-m-wikipedia-org.translate.goog/wiki/Scherbius_%26_Ritter?_x_tr_sl=de&_x_tr_tl=en&_x_tr_hl=en&_x_tr_pto=sc> – Scherbius & Ritter
* <https://de-m-wikipedia-org.translate.goog/wiki/Ernst_Richard_Ritter?_x_tr_sl=de&_x_tr_tl=en&_x_tr_hl=en&_x_tr_pto=sc> – Ernst Richard Ritter
* <https://liceunet.ro/ghid-matrice/inversa> - Inversa unei matrice