**Wyższa Szkoła Bankowa w Chorzowie**

**Bartosz Wróbel 110129**

|  |
| --- |
|  |

**SPRAWOZDANIE Z ĆWICZENIA NR 1**

***Przetwarzanie Informacji Multimedialnej***

***ROZDZIAŁ I***

* 1. ***Na podstawie wskazanych algorytmów użyć strony*** [***https://www.onlinegdb.com/online\_c++\_compiler***](https://www.onlinegdb.com/online_c++_compiler)***. Zmodyfikować każdy z programów, aby dla każdego testować 5 różnych tekstów z dowolnej strony internetowej (po angielsku dla prostoty kodowania ASCII). Dla każdego z tekstów ustalić wzorzec do wyszukania (wzorzec nie musi się znajdować w tekście).***

|  |  |  |
| --- | --- | --- |
| **Algorytm naiwny** | Tekst: She sells sea shells by the sea shore | Wzorzec: „sea shells” |
| #include <iostream>  using namespace std;  #include <iostream>  #include <string>  #include <cstdlib>  #include <time.h>  #include <string.h>  using namespace std;  int main( )  {  string s, p;  int i;    srand ( ( unsigned )time ( NULL ) );  // generujemy łańcuch  s = "**In She sells sea shells by the sea shore**";  //for( i = 0; i < 60; i++ ) s += char ( 65 + ( rand( ) % 3 ) );  // generujemy wzorzec  p = "**sea shells**";  //for( i = 0; i < 3; i++ ) p += char ( 65 + ( rand( ) % 3 ) );  // wypisujemy wzorzec  cout << p << endl;  // wypisujemy łańcuch  cout << s << endl;  // szukamy wzorca w łańcuchu  for( i = 0; i < s.length(); i++ )  cout << ( p == s.substr ( i, p.length() ) ? "^": " " );  cout << endl << endl;  return 0;  } | | |
| Wynik |  | |
| **Algorytm Morrisa-Prata** | Tekst: A quick brown fox jumped over a lazy dog | Wzorzec: „fox” |
| #include <iostream>  #include <string>  #include <cstdlib>  #include <time.h>  using namespace std;  int main()  {  string s, p;  // generujemy łańcuch s  s = " **A quick brown fox jumped over a lazy dog** ";  const int N = s.length();  // generujemy wzorzec  p = " **fox**";  const int M = p.length();  int i, b, pp;  int\* PI = new int[M + 1];  srand((unsigned)time(NULL));  // dla wzorca obliczamy tablicę PI [ ]  PI[0] = b = -1;  for (i = 1; i <= M; i++)  {  while ((b > -1) && (p[b] != p[i - 1]))  {  b = PI[b];  }  PI[i] = ++b;  }    // wypisujemy wzorzec  cout << p << endl;  // wypisujemy łańcuch s  cout << s;  // poszukujemy pozycji wzorca w łańcuchu  pp = b = 0;  for (i = 0; i < N; i++)  {  while ((b > -1) && (p[b] != s[i]))  {  b = PI[b];  }  if (++b == M)  {  while (pp < i - b + 1)  {  cout << " ";  pp++;  }  cout << "^";  pp++;  b = PI[b];  }  }  if (pp==0)  { cout << endl << "Brak frazy" << endl;  return 0;  }  else {  cout << endl << "Fraza odnaleziona na pozycji = " << pp << endl;  return 0;  }  } | | |
| Wynik |  | |

|  |  |  |
| --- | --- | --- |
| **Algorytm Knutha-Morrisa-Pratta** | Tekst: Yesterday all my troubles seemed so far away | Wzorzec: „away” |
| #include <iostream>  #include <string>  #include <cstdlib>  #include <time.h>  using namespace std;  const int N = 44; // długość łańcucha s  const int M = 4; // długość wzorca p  int main( )  {  string s, p;  int KMPNext [ M + 1 ], i, b, pp;  srand ( ( unsigned )time ( NULL ) );  // generujemy łańcuch s  s = "**Yesterday all my troubles seemed so far away**";  //for( i = 0; i < N; i++ ) s += 65 + rand( ) % 2;  // generujemy wzorzec  p = "**away**";  //for( i = 0; i < M; i++ ) p += 65 + rand( ) % 2;  // dla wzorca obliczamy tablicę Next [ ]  KMPNext [ 0 ] = b = -1;  for( i = 1; i <= M; i++ )  {  while( ( b > -1 ) && ( p [ b ] != p [ i - 1 ] ) ) b = KMPNext [ b ];  ++b;  if( ( i == M ) || ( p [ i ] != p [ b ] ) ) KMPNext [ i ] = b;  else KMPNext [ i ] = KMPNext [ b ];  }  // wypisujemy wzorzec  cout << p << endl;  // wypisujemy łańcuch s  cout << s << endl;  // poszukujemy pozycji wzorca w łańcuchu  pp = b = 0;  for( i = 0; i < N; i++ )  {  while( ( b > -1 ) && ( p [ b ] != s [ i ] ) ) b = KMPNext [ b ];  if( ++b == M )  {  while( pp < i - b + 1 )  {  cout << " "; pp++;  }  cout << "^"; pp++;  b = KMPNext [ b ];  }  }  cout << endl;  return 0;  } | | |
| Wynik |  | |

|  |  |  |
| --- | --- | --- |
| **Algorytm Boyera-Moora** | Tekst: Abra cabra cadabra | Wzorzec: „Cabra” |
| #include <iostream>  #include <string>  #include <cstdlib>  #include <time.h>  using namespace std;  const int N = 18; // długość łańcucha s  const int M = 5; // długość wzorca p  const int zp = 65; // kod pierwszego znaku alfabetu  const int zk = 65; // kod ostatniego znaku alfabetu  int main( )  {  string s, p;  int Last [ zk - zp + 1 ], BMNext [ M + 1 ], Pi [ M + 1 ], b, i, j, pp;  srand ( ( unsigned )time ( NULL ) );  // generujemy łańcuch s  s = "**Abra cabra cadabra**";  // generujemy wzorzec p  p = "**cabra**";    // wypisujemy wzorzec  cout << p << endl;  // wypisujemy łańcuch  cout << s << endl;  // dla wzorca obliczamy tablicę Last [ ]  for( i = 0; i <= zk - zp; i++ ) Last [ i ] = -1;  for( i = 0; i < M; i++ ) Last [ p [ i ] - zp ] = i;  // Etap I obliczania tablicy BMNext [ ]  for( i = 0; i <= M; i++ ) BMNext [ i ] = 0;  i = M; b = M + 1; Pi [ i ] = b;  while( i > 0 )  {  while( ( b <= M ) && ( p [ i - 1 ] != p [ b - 1 ] ) )  {  if( BMNext [ b ] == 0 ) BMNext [ b ] = b - i;  b = Pi [ b ];  }  Pi [ --i ] = --b;  }  // Etap II obliczania tablicy BMNext [ ]  b = Pi [ 0 ];  for( i = 0; i <= M; i++ )  {  if( BMNext [ i ] == 0 ) BMNext [ i ] = b;  if( i == b ) b = Pi [ b ];  }  // szukamy pozycji wzorca w łańcuchu  pp = i = 0;  while( i <= N - M )  {  j = M - 1;  while( ( j > -1 ) && ( p [ j ] == s [ i + j ] ) ) j--;  if( j == -1 )  {  while( pp < i )  {  cout << " "; pp++;  }  cout << "^"; pp++;  i += BMNext [ 0 ];  }  else i += max ( BMNext [ j+1 ], j-Last [ s [ i+j ] -zp ] );  }  cout << endl;  return 0;  } | | |
| Wynik |  | |

* 1. **Dla każdego z czterech algorytmów i wygenerowanych 5 par „tekst / wzorzec” wykonać tabelkę**

|  |  |  |  |
| --- | --- | --- | --- |
| Algorytm | Tekst | Wzorzec | Wynik |
| **Algorytm naiwny** | She sells sea shells by the sea shore | sea shells |  |
| This video hands-down wins the internet | video |  |
| share your beloved pet with the world | beloved pet |  |
| Here are the best movies to stream on Disney+ | netflix | (brak wyników ze względu na brak frazy „netflix”) |
| Kevin has accidentally been left home alone | Home alone | (brak wyników ze względu na różnicę liter „h” vs „H”) |

|  |  |  |  |
| --- | --- | --- | --- |
| Algorytm | Tekst | Wzorzec | Wynik |
| **Algorytm Morrisa-Prata** | She sells sea shells by the sea shore | sea shells |  |
| This video hands-down wins the internet | video |  |
| share your beloved pet with the world | beloved pet |  |
| Here are the best movies to stream on Disney+ | netflix | (brak wyników ze względu na brak frazy „netflix”) |
| Kevin has accidentally been left home alone | Home alone | (brak wyników ze względu na różnicę liter „h” vs „H”) |

|  |  |  |  |
| --- | --- | --- | --- |
| Algorytm | Tekst | Wzorzec | Wynik |
| **Algorytm Knutha-Morrisa-Pratta** | She sells sea shells by the sea shore | sea shells | (algorytm odnalazł dwie frazy „sea” choć cała fraza brzmiała „sea shells”) |
| This video hands-down wins the internet | video |  |
| share your beloved pet with the world | beloved pet |  |
| Here are the best movies to stream on Disney+ | netflix | (brak wyników ze względu na brak frazy „netflix”) |
| Kevin has accidentally been left home alone | Home alone | (brak wyników ze względu na różnicę liter „h” vs „H”) |

|  |  |  |  |
| --- | --- | --- | --- |
| Algorytm | Tekst | Wzorzec | Wynik |
| **Algorytm Boyera-Moora** | She sells sea shells by the sea shore | sea shells |  |
| This video hands down wins the internet | video |  |
| share your beloved pet with the world | beloved pet |  |
| Here are the best movies to stream on Disney+ | netflix | (brak wyników ze względu na brak frazy „netflix”) |
| Kevin has accidentally been left home alone | Home alone | (brak wyników ze względu na różnicę liter „h” vs „H”) |

**1.3. Przedstawić krótką dyskusję wybranego algorytmu z punktu widzenia złożoności obliczeniowej, pozytywnych oraz negatywnych cech.**

Algorytm Morrisa-Pratta jest wysoce wydajnym i prostym algorytmem, który dobrze nadaje się do zastosowań związanych z wyszukiwaniem tekstu na dużą skalę. Jednak nie jest on idealnym rozwiązaniem. Jego ograniczenia i wymagania dotyczące przetwarzania wstępnego oraz wykorzystania pamięci powinny być brane pod uwagę przy dobraniu go do wyznaczonego zadania.

Jedną z głównych zalet algorytmu jest to, że jego wydajność rośnie wykładniczo wraz z rozmiarem przeszukiwanego bloku tekstu. To pokazuje, że nadaje się idealnie do wykorzystania w programach przeszukujących frazę w dużych rozmiarach tekstu . Kolejną zaletą algorytmu jest prostota, pozwalająca na łatwą implementację przez programistów. Ograniczeniem algorytmu jest fakt, że został stworzony do odnajdywania pojedynczego wzorca w tekście i trudno go przystosować do znajdowania wielu wzorców bądź adaptacji do innych typów operacji wyszukiwania tekstu. Algorytm również jest zasobożerny przez wstępne przetwarzanie przez co nie będzie dobrym wyborem na urządzeniach o gorszej specyfikacji. Wstępne przetwarzanie jest potrzebne w celu określenia odległości pomiędzy wystąpieniami prefiksów wzorca, co może zwiększyć ogólną złożoność czasową algorytmu.

***ROZDZIAŁ II***

* 1. **Uruchomić każdy z algorytmów z użyciem strony** [**https://www.onlinegdb.com/online\_c++\_compiler**](https://www.onlinegdb.com/online_c++_compiler) **dla co najmniej dwóch różnych kombinacji danych wejściowych.**

|  |
| --- |
| **Szyfrowanie kodem Cezara** |
| // Szyfrowanie Kodem Cezara  // Data: 18.08.2008  // (C)2020 mgr Jerzy Wałaszek  //-----------------------------  #include <iostream>  #include <string>  using namespace std;  int main( )  {  string s;  int i;  // pobieramy i wypisujemy wiersz znaków  s = "Marnypopis";  cout << s << endl;    // zamieniamy małe litery na duże  // i kodujemy szyfrem cezara  for( i = 0; i < s.length( ); i++ )  {  s [ i ] = toupper ( s [ i ] );  if( ( s [ i ] >= 'A' ) && ( s [ i ] <= 'Z' ) ) s [ i ] = char ( 65 + ( s [ i ] - 62 ) % 26 );  }  // wypisujemy zaszyfrowany tekst  cout << s << endl << endl;  return 0;  } |
|  |

|  |
| --- |
| **Szyfrowanie z pseudolosowym odstępem** |
| // Szyfrowanie z pseudolosowym odstępem  // Data: 20.08.2008  // (C)2020 mgr Jerzy Wałaszek  //-----------------------------  #include <iostream>  #include <string>  using namespace std;  int main( )  {  string s;  int i;  unsigned long long X, a, m, c;  // definiujemy generator LCG  m = 3956280000ull;  a = 1978141ull;  c = 1309ull;  // pobieramy klucz i wiersz znaków  X = 113278;  s = "Marnypopis";    // wypisujemy wiersz znaków  cout << s << endl;  // szyfrujemy  for( i = 0; i < s.length( ); i++ )  {  // obliczamy kolejną liczbę pseudolosową  X = ( a \* X + c ) % m;  // szyfrujemy literkę  s [ i ] = toupper ( s [ i ] );  if( ( s [ i ] >= 'A' ) && ( s [ i ] <= 'Z' ) ) s [ i ] = 65 + ( s [ i ] - 65 + X % 26 ) % 26;  }  // wypisujemy zaszyfrowany tekst  cout << s << endl << endl;  return 0;  } |
|  |

|  |
| --- |
| **Szyfr przestawieniowy** |
| // Szyfr przestawieniowy  // Data: 11.02.2011  // (C)2020 mgr Jerzy Wałaszek  //-----------------------------  #include <iostream>  #include <string>  using namespace std;  int main( )  {  string s;  unsigned i;  // pobieramy tekst  s = "Obelix";  // wypisujemy tekst  cout << s << endl;  // zamieniamy miejscami litery  for( i = 0; i < s.length( ) - 1; i += 2 )  swap ( s [ i ], s [ i + 1 ] );  // wyświetlamy wynik  cout << s << endl;  return 0;  } |
|  |

|  |
| --- |
| **Szyfr enigmy** |
| // Symulator Enigmy  // Data: 22.08.2008  // (C)2020 mgr Jerzy Wałaszek  //-----------------------------  #include <iostream>  #include <string>  using namespace std;  // definicje elementów Enigmy  const string pierscien\_szyfr [ 5 ] = {"EKMFLGDQVZNTOWYHXUSPAIBRCJ",  "AJDKSIRUXBLHWTMCQGZNPYFVOE",  "BDFHJLCPRTXVZNYEIWGAKMUSQO",  "ESOVPZJAYQUIRHXLNFTGKDCMWB",  "VZBRGITYUPSDNHLXAWMJQOFECK"};  const string przeniesienie = "RFWKA";  const string beben\_odwr = "YRUHQSLDPXNGOKMIEBFZCWVJAT";  int main( )  {  int pierscien [ 3 ], i, j, k, n, c;  bool ruch;  string szyfr, s, lacznica;  // pobieramy i wypisujemy konfigurację pierścieni szyfrujących  n = 125;  cout << n << endl;  for( i = 2; i >= 0; i-- )  {  pierscien [ i ] = ( n % 10 ) - 1; // numer pierścienia na i-tej pozycji  n /= 10;  }  // pobieramy i wypisujemy położenia początkowe pierścieni  szyfr = "BDF";  cout << szyfr << endl;    for( i = 0; i < szyfr.length( ); i++ ) szyfr [ i ] = toupper ( szyfr [ i ] );  // pobieramy i wypisujemy stan łącznicy wtyczkowej  s = "RZ";  cout << s << endl;    for( i = 0; i < s.length( ); i++ ) s [ i ] = toupper ( s [ i ] );    lacznica = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";  for( i = 0; i < s.length( ) - 1; i += 2 )  {  lacznica [ s [ i ] - 65 ] = s [ i + 1 ];  lacznica [ s [ i + 1 ] - 65 ] = s [ i ];  }  // pobieramy i wypisujemy szyfrogram  s = "Antywirus";  cout << s << endl;    for( i = 0; i < s.length( ); i++ ) s [ i ] = toupper ( s [ i ] );  // szyfrujemy/rozszyfrowujemy szyfrogram  for( i = 0; i < s.length( ); i++ )  {  // najpierw ruch pierścieni szyfrujących  for( ruch = true, j = 2; ruch && ( j >= 0 ); j-- )  {  ruch = ( szyfr [ j ] == przeniesienie [ pierscien [ j ] ] );  szyfr [ j ] = 65 + ( szyfr [ j ] - 64 ) % 26;  }  // pobieramy znak szyfrogramu  c = s [ i ];  // przechodzimy przez łącznicę wtyczkową  c = lacznica [ c - 65 ];  // przechodzimy przez pierścienie w kierunku do bębna odwracającego  for( j = 2; j >= 0; j-- )  {  k = szyfr [ j ] - 65;  c = pierscien\_szyfr [ pierscien [ j ] ][ ( c - 65 + k ) % 26 ];  c = 65 + ( c - 39 - k ) % 26;  }  // przechodzimy przez bęben odwracający  c = beben\_odwr [ c - 65 ];  // wracamy ścieżką powrotną  for( j = 0; j < 3; j++ )  {  k = szyfr [ j ] - 65;  c = 65 + ( c - 65 + k ) % 26;  for( n = 0; pierscien\_szyfr [ pierscien [ j ] ][ n ] != c; n++ );  c = 65 + ( 26 + n - k ) % 26;  }  // przechodzimy przez łącznicę wtyczkową  c = lacznica [ c - 65 ];  // uaktualniamy szyfrogram  s [ i ] = c;  }  // wyświetlamy szyfrogram  cout << s << endl << endl;  return 0;  } |
|  |

|  |
| --- |
| **Szyfr RSA** |
| /\*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*\* Przykładowa aplikacja obrazująca sposób działania \*\*  \*\* asymetrycznego systemu kodowania informacji RSA. \*\*  \*\* ------------------------------------------------- \*\*  \*\* (C)2020 mgr Jerzy Wałaszek \*\*  \*\* I Liceum Ogólnokształcące \*\*  \*\* im. Kazimierza Brodzińskiego \*\*  \*\* w Tarnowie \*\*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*/  #include <iostream>  #include <iomanip>  #include <cstdlib>  #include <time.h>  using namespace std;  // Funkcja czeka na dowolny klawisz i czyści ekran  //------------------------------------------------  void czekaj ( void )  {  char c [ 1 ];  cout << "\nZapisz te dane\n\n";  cin.getline ( c, 1 );  cin.getline ( c, 1 );  for( int i = 1; i < 500; i++ ) cout << endl;  }  // Funkcja obliczająca NWD dla dwóch liczb  //----------------------------------------  int nwd ( int a, int b )  {  int t;  while( b != 0 )  {  t = b;  b = a % b;  a = t;  };  return a;  }  // Funkcja obliczania odwrotności modulo n  //----------------------------------------  int odwr\_mod ( int a, int n )  {  int a0, n0, p0, p1, q, r, t;  p0 = 0; p1 = 1; a0 = a; n0 = n;  q = n0 / a0;  r = n0 % a0;  while( r > 0 )  {  t = p0 - q \* p1;  if( t >= 0 )  t = t % n;  else  t = n - ( ( -t ) % n );  p0 = p1; p1 = t;  n0 = a0; a0 = r;  q = n0 / a0;  r = n0 % a0;  }  return p1;  }  // Procedura generowania kluczy RSA  //---------------------------------  void klucze\_RSA( )  {  const int tp [ 10 ] = {11, 13, 17, 19, 23, 29, 31, 37, 41, 43};  int p, q, phi, n, e, d;  cout << "Generowanie kluczy RSA\n"  "----------------------\n\n";  // generujemy dwie różne, losowe liczby pierwsze  do  {  p = tp [ rand( ) % 10 ];  q = tp [ rand( ) % 10 ];  } while( p == q );  phi = ( p - 1 ) \* ( q - 1 );  n = p \* q;  // wyznaczamy wykładniki e i d  for( e = 3; nwd ( e, phi ) != 1; e += 2 );  d = odwr\_mod ( e, phi );  // gotowe, wypisujemy klucze  cout << "KLUCZ PUBLICZNY\n"  "wykladnik e = " << e  << "\n modul n = " << n  << "\n\nKLUCZ PRYWATNY\n"  "wykladnik d = " << d << endl;  czekaj( );  }  // Funkcja oblicza modulo potęgę podanej liczby  //---------------------------------------------  int pot\_mod ( int a, int w, int n )  {  int pot, wyn, q;  // wykładnik w rozbieramy na sumę potęg 2  // przy pomocy algorytmu Hornera. Dla reszt  // niezerowych tworzymy iloczyn potęg a modulo n.  pot = a; wyn = 1;  for( q = w; q > 0; q /= 2 )  {  if( q % 2 ) wyn = ( wyn \* pot ) % n;  pot = ( pot \* pot ) % n; // kolejna potęga  }  return wyn;  }  // Procedura kodowania danych RSA  //-------------------------------  void kodowanie\_RSA( )  {  int e, n, t;  cout << "Kodowanie danych RSA\n"  "--------------------\n\n"  "Podaj wykladnik = "; cin >> e;  cout << " Podaj modul = "; cin >> n;  cout << "----------------------------------\n\n"  "Podaj kod RSA = "; cin >> t;  cout << "\nWynik kodowania = " << pot\_mod ( t, e, n ) << endl;  czekaj( );  }  // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  // \*\* Program główny \*\*  // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  int main( )  {  int w;  srand ( ( unsigned )time ( NULL ) );  do  {  cout << "System szyfrowania danych RSA\n"  "-----------------------------\n"  " (C)2012 mgr Jerzy Walaszek\n\n"  "MENU\n"  "====\n"  "[ 0 ] - Koniec pracy programu\n"  "[ 1 ] - Generowanie kluczy RSA\n"  "[ 2 ] - Kodowanie RSA\n\n"  "Jaki jest twoj wybor? ( 0, 1 lub 2 ) : ";  cin >> w;  cout << "\n\n\n";  switch( w )  {  case 1 : klucze\_RSA( ); break;  case 2 : kodowanie\_RSA( ); break;  }  cout << "\n\n\n";  } while( w != 0 );  return 0;  } |
|  |

* 1. **Podać wynik działania konsolki programu dla uruchomionych kombinacji danych wejściowych.**

|  |  |  |
| --- | --- | --- |
| **Szyfr Cezara** | **Wejście** | **Wynik programu** |
| **Marnypopis** |  |
| **Asterix** |  |

|  |  |  |
| --- | --- | --- |
| **Szyfrowanie z pseudolosowym dostępem** | **Wejście** | **Wynik programu** |
| **Marnypopis** |  |
| **Asterix** |  |

|  |  |  |
| --- | --- | --- |
| **Szyfr przestawieniowy** | **Wejście** | **Wynik programu** |
| **Obelix** |  |
| **Panoramix** |  |

|  |  |  |
| --- | --- | --- |
| **Szyfr enigmy** | **Wejście** | **Wynik programu** |
| **125**  **BDF**  **RZ**  **Antywirus** |  |
| **125**  **BDF**  **RZ**  **GajuszCePlus** |  |

|  |  |  |
| --- | --- | --- |
| **RSA Generowanie** | **Wygenerowane** | **Wynik programu** |
| **E=3**  **N=493**  **D=299** |  |
| **E=3**  **N=667**  **D=411** |  |

|  |  |  |
| --- | --- | --- |
| **RSA Kodowanie** | **Kodowanie** | **Wynik programu** |
| **Wykładnik: 3**  **Modul: 493**  **Kod RSA: 250** |  |
| **Wykładnik: 3**  **Modul: 667**  **Kod RSA: 125** |  |

* 1. **Przedstawić krótką dyskusję wybranego algorytmu z punktu widzenia złożoności obliczeniowej, pozytywnych oraz negatywnych cech.**

Szyfrowanie z pseudolosowym dostępem to sposób szyfrowania, w której zaszyfrowane informacje są ukryte w niezwiązanym ze sobą, szyfrowanym strumieniu danych. Zaletą metody jest to, że umożliwia ona dostęp do każdej sekcji zaszyfrowanych danych bez konieczności odczytywania całej jej zawartości. To pozwala na szybki dostęp do potrzebnych informacji, jednocześnie zapewniając ochronę przed nieuprawnionym dostępem do pozostałej treści. Wadą tej metody jest wymóg posiadania dużej ilości pamięci i sensownego procesora w celu ukrycia danych w strumieniu. Ponadto, jeśli klucz szyfrujący jest źle zabezpieczony, może zostać złamany w krótkim czasie , co grozi dostępem do danych przez nieupoważnione osoby.

Szyfrowanie z pseudolosowym dostępem jest użyteczne, gdy potrzebna jest szybkość dostępu do danych i ochrona przed nieuprawnionym dostępem, w zamian za ilość wykorzystania zasobów systemowych oraz dbałość o odpowiednie zabezpieczenie klucza szyfrującego.