Aplikacja Maple do kryptograficznej ochrony folderów, utworzonych na pamięciach chmurowych, dyskach twardych i na nośnikach wymiennych

Maple implementation of cryptographic protection of folders created in cloud storage, hard disks and portable memory devices

Czesław Kościelny[1]

Treść: Opisano sposób realizacji aplikacji do szyfrowania i deszyfrowania folderów. Aplikację zrealizowano stosując program Maple 2016. Jest to program wyjątkowo łatwy w obsłudze, ponieważ użytkownik stosuje wyłącznie lewy klawisz myszy, wybierając folder, który będzie przetwarzany oraz jedną z operacji, czyli szyfrowanie lub deszyfrowanie folderu. Dlatego też aplikację można polecić specjalistom średnio biegłym w informatyce, jak filolodzy, ekonomiści, prawnicy, osoby duchowne, itp., którzy w swoich komputerach posiadają foldery z poufnymi dokumentami. Program posiada prosty graficzny interfejs użytkownika i przeznaczony jest głównie do kryptograficznej ochrony plików przechowywanych w folderach tworzonych w pamięciach chmurowych, (`cloud storage`) oraz na dyskach twardych i na nośnikach wymiennych. Aplikacja niezawodnie chroni pliki przed nieupoważnionym dostępem.

Słowa kluczowe: szyfr strumieniowy o strukturze bajtowej, szyfrowanie symetryczne plików.

Abstract. It has been shown how to implement a Maple worksheet which performs cryptographic protection of folders created in cloud storage, hard disks and portable memory devices by means of byte-oriented stream-cipher using the addition in GF(256).

Keywords: Maple byte-oriented stream-cipher, symmetric file encryption.

1. Wstęp

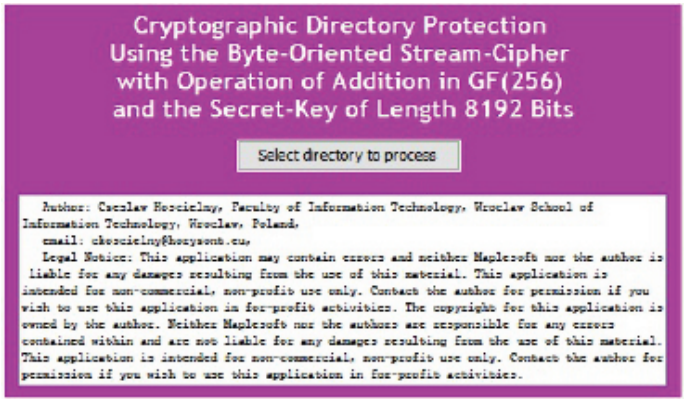
W niniejszym artykule opisano oryginalną aplikację w postaci programu typu `worksheet` o nazwie `a256k8192.mw` utworzonego w środowisku Maple, którego zadaniem jest ochrona kryptograficzna folderów tworzonych na pamięciach chmurowych, dyskach twardych i na nośnikach wymiennych. Jest oczywiste, że przechowywanie niezaszyfrowanych plików w pamięciach chmurowych jest wyjątkowo nierozsądne, dlatego też użytkownicy np. usługi `Microsoft OneDrive` powinni koniecznie stosować opisywany tu program `a256k8192.mw` lub jakiś inny program szyfrujący.
W aplikacji zastosowano narzędzie kryptograficzne w postaci szyfru strumieniowego o strukturze bajtowej, stosującego operację dodawania w 256-elementowym ciele Galois GF(256). Podstawowymi elementami klucza dla tego szyfru jest 1024-elementowa lista o nazwie `K`, zawierająca pseudolosową sekwencję bajtów o wartościach w zakresie 0 .. 255 oraz zmienna `seed` typu `posint`.
W zastosowanym szyfrze strumieniowym długość klucza szyfrującego musi być równa rozmiarowi pliku który jest przetwarzany, zatem sekwencja zawarta w liście `K` jest powtarzana odpowiednią liczbę razy. Dzięki takiej konstrukcji szyfru każdy zaszyfrowany plik ma postać pseudolosowego ciągu bajtów o wartościach od 0 do 255.
Można więc stwierdzić, że wartość tajnego klucza w zastosowanym szyfrze strumieniowym wynosi 8192 bity.
Oczywiście liczbę znaków w liście `K` można dowolnie zmieniać i tworzyć inne aplikacje, mniej lub bardziej odporne na rozszyfrowanie.
Użytkownik łatwo zauważy, że szyfrowana jest nie tylko zawartość pliku jawnego, ale też nazwa tego pliku.
Zaszyfrowana nazwa pliku jawnego nie ma żadnego rozszerzenia i składa się z wybranego zestawu dużych i małych liter alfabetu łacińskiego. W ten sposób ewentualny oponent nie ma żadnych wiadomości na temat struktury plików, zapamiętanych w zaszyfrowanym folderze.
Warto pamiętać, że tajny klucz jest zawarty w aplikacji, która powinna być zapamiętana na pendrajwie, pieczołowicie chronionym i przechowywanym w bezpiecznym miejscu.
Jeśli oponent chciałby podjąc próbę zdeszyfrowania zaszyfrowanego folderu, musiałby znać wartość zmiennej `seed` oraz wartości 1024 elementów listy `K`.
Można obliczyć, że dysponując najszybszym procesorem, wypróbowanie wszystkich wartości `K` oraz `seed` trwa-

1.Wydział Informatyki, Wrocławska Wyższa Szkoła Informatyki Stosowanej, ul. Wejherowska 28, 54-239 Wrocław, ckoscielny@horyzont.eu

łoby dłużej niż wiek ziemi (4 467 000 000 lat).
Zatem prezentowaną aplikację można uważać za bardzo efektywny i skuteczny sposób kryptograficznej ochrony plików przed nieupoważnionym dostępem.

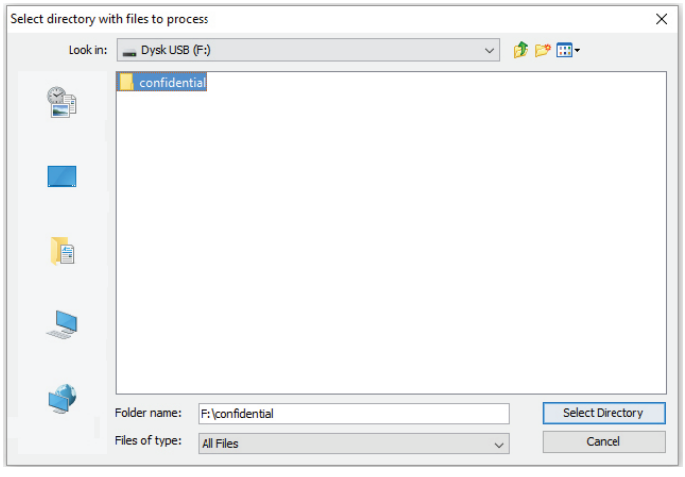
2. Graficzny interfejs użytkownika

Graficzny interfejs użytkownika pokazano na Rys. 1. Interfejs składa się z jednego elementu typu `table` w którym unieszczono elementy `Button0`, `TextArea0` oraz niewidoczny po otwarciu programu w sesji Maple element `ComboBox0`. Aby wybrać folder który ma być szyfrowany lub deszyfrowany użytkownik powinien kliknąć na nagłówku elementu `Button0` (`Select directory to process`).



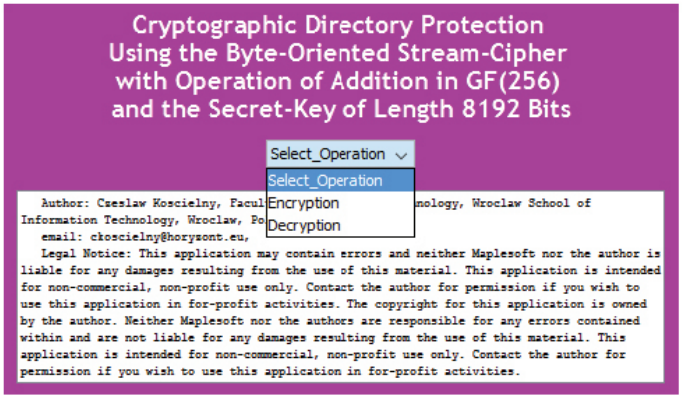
Rys. 1. Graficzny interfejs użytkownika

Po wykonaniu tej czynności pojawi się okno, umożliwiające wybór folderu do zaszyfrowania lub do odszyfrowania.



Rys. 2. Okno do wyboru folderu – wybrano folder `confidential` na pendrawie `F`

Okno pozwala wybrać dowolny folder dostępny w komputerze użytkownika. Po kliknięciu na przycisku `Select Directory` folder zostaje wybrany, element `Button0` staje się niewidoczny, a na interfejsie ukazuje się element `ComboBox0` z napisem `Select_Operation`. Teraz możliwy jest wybór jednej z dwóch opcji: `Encryption` lub `Decryption`. Warto pamiętać, że wybrany folder można szyfrować/deszyfrować dowolną liczbę razy.



Rys. 3. Opcje elementu `ComboBox0`

3. Oprogramowanie aplikacji

W obszarze `Edit Click Action` elementu `Button0` umieszczono instrukcje:

```
cd := dirsel();
use DocumentTools in
   SetProperty(ComboBox0,visible,true);
   SetProperty(Button0,visible,false)
end use;
```

Z kolei obszar kodu startowego aplikacji zawiera pięć procedur i siedem instrukcji:

```
#Generowanie tablicy dodawania w GF(256) i listy K
AEDK := proc(sd::posint)
local f, i, k;
global EcDc, K;
      randomize(sd);
      EcDc := Array(0 .. 255,0 .. 255);
      f := GF(2,8);
      for i from 0 to 255 do
        for k from 0 to 255 do
             EcDc[i,k] := f:-output(f:-`+`(f:-
             input(i),f:-input(k)))
        end do
      end do;
      K := [seq(rand(256)(),i = 1 .. 1024)]
end proc;

#Wybór folderu
dirsel := proc()
local d, i, db, dd;
      dd := Maplets:-Elements:-Maplet(Maplets:-
Elements:-FileDialog['DiDia'](('approvecap-
tion') = „Select Directory",fileselectionmode
= directoriesonly,('title') = „Select di-
rectory with files to process",('directory')
= „C:/",('onapprove') = Maplets:-Elements:-
Shutdown(['DiDia']),('oncancel') = Maplets:-
Elements:-Shutdown()));
    d := Maplets[Display](dd)[1];
      db := convert(d,bytes);
      for i to nops(db) do
```

```
        if db[i] = 92 then
          db[i] := 47
        end if
      end do;
      db := convert(db,bytes)
end proc;

#Szyfrowanie nazwy pliku
fne := proc(fn::string, sd::posint)
local a, l, ib2ob;
      randomize(sd);
      a := combinat:-randcomb([seq(i+65,i = 0..
      25), seq(i+97,i = 0 .. 25)],26);
      ib2ob := table([seq(i-1 = a[i],i = 1 ..
      26)]);
      l := convert(fn,bytes);
      l := convert(l,base,128,26);
      l := [seq(ib2ob[l[i]],i = 1 .. nops(l))];
      convert(l,bytes)
end proc;

#Deszyfrowanie zaszyfrowanej nazwy
pliku
fnd := proc(fn::string, sd::posint)
local a, l;
global ob2ib;
      randomize(sd);
      a := combinat:-randcomb([seq(i+65,i = 0..
      25), seq(i+97,i = 0 .. 25)],26);
      ob2ib := table([seq(a[i] = i-1,i = 1 ..
      26)]);
      l := convert(fn,bytes);
      l := [seq(ob2ib[l[i]],i = 1 .. nops(l))];
      convert(convert(l,base,26,128),bytes)
end proc;

#Usuwanie niepotrzebnego pliku
fr := proc(fn::string)
      if FileTools:-Exists(fn) then
      FileTools:-Remove(fn)
      end if
end proc;

#Ustalanie wartości elementów `TextArea0`,
`Button0`, `ComboBox0` use DocumentTools in
  SetProperty(TextArea0,value," Author: Czeslaw
Koscielny, Faculty of Information Technology,
Wroclaw School of Information Technology, Wroc-
law, Poland, \n  email: ckoscielny@horyzont.eu,
\n  Legal Notice: This application may contain
errors and neither Maplesoft nor the author is
liable for any damages resulting from the use of
this material. This application is intended for
non-commercial, non-profit use only. Contact the
author for permission if you wish to use this
application in for-profit activities. The copyri-
ght for this application is owned by the author.
Neither Maplesoft nor the authors are responsi-
ble for any errors contained within and are not
liable for any damages resulting from the use of
this material. This application is intended for
non-commercial, non-profit use only. Contact the
author for permission if you wish to use this ap-
plication in for-profit activities.");
  SetProperty(Button0,visible,true);
  SetProperty(Button0,caption,"Select directory
to process");
      SetProperty(ComboBox0,value,Select_Opera-
tion);
  SetProperty(ComboBox0,visible,false)
end use;

#Ustalanie wartości zmiennej `seed` i wywołanie
procedury `AEDK` z wybraną przez #użytkownika
wartością parametru `seed`
seed := 1234567654321;
AEDK(seed);
```

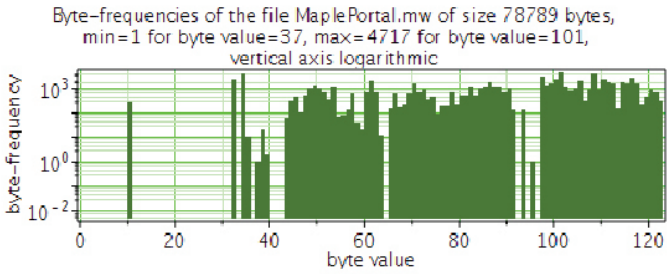
Pozostałą część kodu aplikacji umieszczono w obszarze `ComboBox0 Select Action`.

```
ed := DocumentTools:-GetProperty(ComboBox0,va-
lue);
fl := FileTools:-ListDirectory(cd);
nfl := nops(fl);
ss := {}:
for i to nfl do
   ss := ss union {FileTools:-
Size(cat(cd,"/",fl[i]))}
end do;
fsm := ss[nops(ss)];
K := [seq(K[1+`mod`(s-1,1024)],s=1 .. fsm)]:
t := time[real]();
tfs := 0;
if ed = „Encryption" then
  DocumentTools:-Do(%TextArea0 = „Encrypting .
. ."",refresh = true);
  for i to nfl do
    m := readbytes(cat(cd,"/",fl[i]),infinity);
    fs := nops(m);
    c := [seq(EcDc[m[n],K[n]],n = 1 .. fs)];
    efn := cat(cd,"/",fne(fl[i],seed));
    writebytes(efn,c);
    fclose(efn);
    tfs := tfs+fs;
  end do;
  t := time[real]()-t;
  mes := cat(convert(nfl,string)," files of total
size „,convert(tfs,string)," Bytes have been
encrypted in „,convert(t,string)," seconds. En-
cryption rate: „,convert(round(tfs/t),string),"
Bps. The input files have been removed. To use
the program once again click on the icon `Re-
start Maple server`. To finish select `File -
Exit` or press Alt and F4.");
  DocumentTools:-Do(%TextArea0 = mes,refresh =
true)
elif ed = „Decryption" then
  DocumentTools:-Do(%TextArea0 = „Decrypting .
. ."",refresh = true);
  for i to nfl do
    m := readbytes(cat(cd,"/",fl[i]),infinity);
    fs := nops(m);
    c := [seq(EcDc[m[n],K[n]],n = 1 .. fs)];
    efn := cat(cd,"/",fnd(fl[i],seed));
    writebytes(efn,c);
    fclose(efn);
    tfs := tfs+fs;
  end do;
  efl := FileTools:-ListDirectory(cd);
  t := time[real]()-t;
  mes := cat(convert(nfl,string)," files of to-
tal size „,convert(tfs,string)," have been de-
crypted in „,convert(t,string)," seconds. De-
cryption rate: „,convert(round(tfs/t),string),"
Bps. The input files have been removed. To use
the program once again click on the icon `Re-
start Maple server`. To finish select `File -
Exit` or press Alt and F4.");
  DocumentTools:-Do(%TextArea0 = mes,refresh =
```

```
true)
end if;
+for i to nfl do
    fr(cat(cd,"/",fl[i]))
end do:
DocumentTools:-SetProperty(ComboBox0,visible,f
alse);
```

4. Przykład

Zakładając, że w folderze pokazanym na Rys. 2 znajduje się plik `MaplePortal.mw` dostępny w instalacji programu Maple pod adresem `C:/Program Files/Maple 2016/examples`



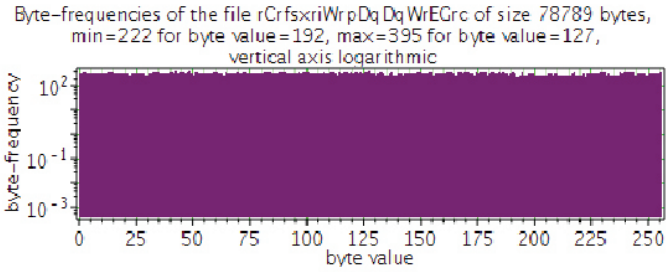
Rys. 4. Częstotliwości występowania bajtów w pliku niezaszyfrowanym o strukturze bajtowej pokazanej na Rys. 4 (strukturę bajtową pliku można obliczyć za pomocą aplikacji [8]).

W pliku `MaplePortal.mw` występują 83 wartości bajtowe podane niżej w postaci par

(wartość_bajtu liczba_wystąpień)

```
Byte-frequencies in the file `MaplePortal.mw`
(10 289),(32 2522),(34 3912),(35 11),(37 1),(38
21),(39 2),(43 63),(44 304),(45 494),(46
110),(47 535),(48 1100),(49 1444),(50 1071),(51
712),(52 342),(53 1163),(54 71),(55 86),(56
136),(57 596),(58 39),(59 21),(60 704),(61
1964),(62 704),(63 13),(65 150),(66 625),(67
174),(68 252),(69 623),(70 1588),(71 700),(72
911),(73 339),(74 483),(75 99),(76 218),(77
216),(78 684),(79 254),(80 561),(81 549),(82
1187),(83 573),(84 1027),(85 1258),(86
1906),(87 1295),(88 1267),(89 694),(90
1098),(91 142),(93 142),(95 1),(97 3203),(98
1316),(99 1616),(100 2438),(101 4717),(102
1224),(103 802),(104 1040),(105 2148),(106
315),(107 869),(108 4127),(109 983),(110
2352),(111 1815),(112 1636),(113 152),(114
1905),(115 1665),(116 2791),(117 1516),(118
255),(119 458),(120 989),(121 677),(122 334),
min. freq. = 1 for byte value = 37,
max. freq. = 4717 for byte value = 101.
```

Po zaszyfrowaniu plik zmienia nazwę na `rCrfsxriWrpDqDqWrEGrc`.
Rozmiar zaszyfrowanego pliku jest taki sam jak rozmiar pliku niezaszyfrowanego, ale w tym pliku występuje teraz 256 wartości bajtowych z zakresu 0 .. 255. Częstotliwość występowania wartości bajtowych jest dosyć równomierna, co ilustruje Rys. 5



Rys. 5. Częstotliwości występowania bajtów w pliku zaszyfrowanym

oraz zestaw par (wartość_bajtu liczba wystąpień).

```
Byte-frequencies in the file`rCrfsxriWrpDqDqWr-
EGrc`
(0 300),(1 321),(2 344),(3 314),(4 314),(5
294),(6 313),(7 286),(8 291),(9 298),(10
306),(11 300),(12 306),(13 336),(14 334),(15
292),(16 302),(17 298),(18 294),(19 289),(20
297),(21 310),(22 266),(23 300),(24 310),(25
304),(26 289),(27 316),(28 297),(29 267),(30
291),(31 309),(32 343),(33 355),(34 335),(35
294),(36 289),(37 308),(38 284),(39 318),(40
325),(41 317),(42 370),(43 309),(44 369),(45
335),(46 287),(47 324),(48 377),(49 302),(50
273),(51 359),(52 308),(53 283),(54 285),(55
313),(56 300),(57 311),(58 316),(59 353),(60
367),(61 322),(62 292),(63 246),(64 350),(65
293),(66 288),(67 309),(68 319),(69 342),(70
298),(71 300),(72 327),(73 312),(74 316),(75
302),(76 289),(77 304),(78 298),(79 294),(80
296),(81 339),(82 328),(83 328),(84 317),(85
307),(86 295),(87 312),(88 332),(89 344),(90
328),(91 303),(92 286),(93 303),(94 304),(95
312),(96 332),(97 297),(98 349),(99 346),(100
360),(101 300),(102 322),(103 314),(104
311),(105 299),(106 283),(107 337),(108
304),(109 315),(110 333),(111 363),(112
316),(113 262),(114 350),(115 284),(116
328),(117 355),(118 299),(119 354),(120
296),(121 364),(122 316),(123 327),(124
308),(125 328),(126 359),(127 395),(128
315),(129 293),(130 268),(131 296),(132
324),(133 315),(134 258),(135 274),(136
299),(137 325),(138 276),(139 268),(140
304),(141 298),(142 271),(143 309),(144
305),(145 303),(146 290),(147 305),(148
341),(149 339),(150 312),(151 312),(152
318),(153 301),(154 333),(155 268),(156
293),(157 318),(158 318),(159 318),(160
322),(161 339),(162 357),(163 334),(164
306),(165 278),(166 353),(167 347),(168
282),(169 325),(170 327),(171 309),(172
316),(173 267),(174 351),(175 312),(176
333),(177 335),(178 342),(179 279),(180
314),(181 343),(182 300),(183 319),(184
353),(185 316),(186 357),(187 314),(188
268),(189 314),(190 365),(191 319),(192
222),(193 272),(194 333),(195 252),(196
278),(197 301),(198 292),(199 302),(200
266),(201 275),(202 270),(203 279),(204
277),(205 276),(206 297),(207 285),(208
280),(209 284),(210 260),(211 271),(212
266),(213 260),(214 262),(215 269),(216
299),(217 346),(218 274),(219 250),(220
278),(221 287),(222 233),(223 294),(224
312),(225 284),(226 318),(227 291),(228
290),(229 332),(230 332),(231 279),(232
305),(233 258),(234 272),(235 277),(236
```

```
281),(237 332),(238 307),(239 335),(240
326),(241 329),(242 348),(243 333),(244
267),(245 311),(246 284),(247 283),(248
284),(249 272),(250 312),(251 303),(252
306),(253 286),(254 305),(255 344),
min. freq. = 222 for byte value = 192,
max. freq. = 395 for byte value = 127.
```

Inaczej mówiąc, zaszyfrowany plik ma postać pseudolosowej sekwencji bajtów z zakresu 0 .. 255. Podobną strukturę ma każdy plik zaszyfrowany.

5. Podsumowanie i wnioski

W artykule przedstawiono sposób tworzenia bardzo szybkiej, efektywnej aplikacji do kryptograficznej ochrony folderów przed nieupoważnionym dostępem. Aplikacja działa poprawnie pod warunkiem, że wybrany do przetwarzania folder nie posiada podfolderów, tzn. że w wybranym folderze zapisane są wyłącznie pliki i że w folderze zapamiętany jest co najmniej jeden plik.
Zaawansowany użytkownik Maple bez problemu utworzy plik `**a256k8192.mw**`, ale początkujący użytkownicy Maple mogą mieć pewne problemy z wykonaniem tego zadania. W takich przypadkach wystarczy pobrać plik `**a256k8192.mw**`, zapamiętany pod adresem http://wdawnictwo.horyzont.eu/podstrony/publikacje.html.

6. Literatura

[1] Czesław Kościelny, Mirosław Kurkowski, Marian Srebrny – Modern Cryptography Primer, Springer Verlag, 2013
[2] Czesław Kościelny - Program Maple do szyfrowania i deszyfrowania plików, zapisanych na dyskach i na nośnikach wymiennych, Biuletyn Naukowy Wrocławskiej Wyższej Szkoły Informatyki Stosowanej, Informatyka 2013.
[3] https://en.wikipedia.org/wiki/Stream_cipher.
[4] https://en.wikipedia.org/wiki/Cloud_storage
[5] How to Encrypt Files and Folders in Windows 10, http://windowsreport.com/encrypt-files-folders-windows-10/
[6] Czesław Kościelny, Cryptographic Protection of Definite PC Directory Using the AES Algorithm, http://fr.maplesoft.com/applications/view.aspx?SID=7086, 2009
[7] Dengguo Feng, Xiutao Feng, Wentao Zhang, Chuankun Wu, 1995, A Byte-Oriented Stream Cipher, https://www.researchgate.net/publication/220776034_Loiss_A_Byte-Oriented_Stream_Cipher
[8] Czesław Kościelny, Byte Frequency Analyzer, 2015, http://fr.maplesoft.com/applications/view.aspx?SID=153920