Program Maple do szyfrowania i deszyfrowania plików, zapisanych na dyskach i na nośnikach wymiennych

Maple implementation of an interactive application for cryptographic protection of files stored on hard disk and portable memory devices

Czesław Kościelny[1]

Treść: Opisano przykład interaktywnej aplikacji typu `workhsheet` uruchamianej w środowisku Maple 2015.1 i realizującej zadania „bezkluczowego" szyfrowania i deszyfrowania plików. Aplikacja jest prosta w obsłudze, ponieważ użytkownik nie wprowadza żadnych danych tylko używa myszy. Program posiada prosty graficzny interfejs użytkownika i przeznaczony jest głównie do kryptograficznej ochrony plików przechowywanych na dyskach i na nośnikach wymiennych. Aplikacja wyjątkowo skutecznie i niezawodnie chroni pliki przed nieupoważnionym dostępem.

Słowa kluczowe: funkcja biblioteczna Maple convert/base, szyfrowanie symetryczne plików.

Abstract: An example of an interactive implementation Maple worksheet application which performs the „keyless" file encryption or decryption by means of the symmetric cipher has been presented. The application has simple graphical user interface and may be used mainly for cryptographic protection of files stored in disks and in portable memory devices. The application very effectively protect files against unauthorized access.

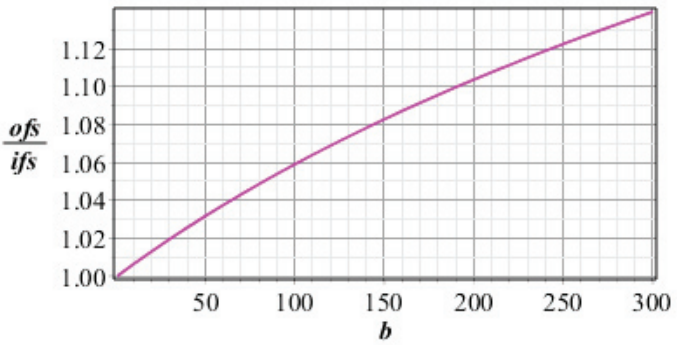
Keywords: Maple convert/base built-in function, symmetric file encryption

1. Wstęp

W pracy [1] przedstawiono program umożliwiający ochronę załączników poczty elektronicznej przed nieupoważnionym dostępem. W niniejszym artykule opisano oryginalną aplikację w postaci programu Maple typu worksheet o nazwie **sdpdnw.mw**, dostępnego w witrynie WWSIS pod adresem
http://www.wydawnictwo.horyzont.eu/publikacje/Informatyka/,
którego zadaniem jest kryptograficzna ochrona plików przechowywanych na dyskach twardych i na nośnikach wymiennych. Aplikację zrealizowano stosując najnowszą wersję tego matematycznego narzędzia, Maple 2015.1.

Rys. 1. Logo Maple 2015.1

Oryginalność prezentowanego rozwiązania polega na zastosowaniu funkcji bibliotecznej programu Maple o nazwie **convert/base** jako przekształcenia kryptograficznego. W procedurach realizujących zadanie szyfrowania i deszyfrowania tę funkcję biblioteczną wywołuje się z parametrem zawierającym zmienną typu **posint** o nazwie **b**. Od wartości tej zmiennej zależy rozmiar zaszyfrowanego pliku oraz długość



Ryc. 2. Zależność stosunku **ofs/ifs** od zmiennej **b** dla **ifs** = 36496 bajtów,
ofs – rozmiar pliku zaszyfrowanego w bajtach , **ifs** – rozmiar pliku niezaszyfrowanego w bajtach.

tajnego klucza. Jeśli **b** = 1, to **ofs/ifs** = 1 a aplikacja realizuje szyfrowanie jedynie nazwy pliku, zapisując pod tą nazwą plik wybrany do zaszyfrowania i usuwając z dysku lub nośnika plik wejściowy. Przy **b** większym lub równym 2 aplikacja szyfruje zarówno nazwę jak i zawartość pliku. W przypadku gdy **b** = 2 a ifs = 36496 stosunek **ofs/ifs** = 1,00071. Na Rys. 2. pokazano zależność tego stosunku od zmiennej **b** w zakresie od 1 do 300 jeśli rozmiar szyfrowanego pliku wynosi 36496 bajtów. Teoretycznie **b**

1. Wrocławska Wyższa Szkoła Informatyki Stosowanej, ul. Wejherowska 28, 54-239 Wrocław

może mieć wartość dowolnie dużej liczby naturalnej akceptowaną przez system Maple. W tabeli 1. zamieszczono wartość **ofs/ifs** dla ośmiu wartości **b** w zakresie od 500 do 200 000.

Tabela 1. Zależność stosunku **ofs/ifs** od zmiennej **b** o wartościach w zakresie od 500 do 200000 przy ifs = 36496

b	$5\cdot10^2$	$1\cdot10^3$	$2\cdot10^3$	$5\cdot10^3$	$1\cdot10^4$	$2\cdot10^4$	$5\cdot10^4$	$1\cdot10^5$	$2\cdot10^5$
$\frac{ofs}{ifs}$	1, 19	1, 27	1, 38	1, 55	1, 66	1, 78	1, 95	2, 08	2, 2

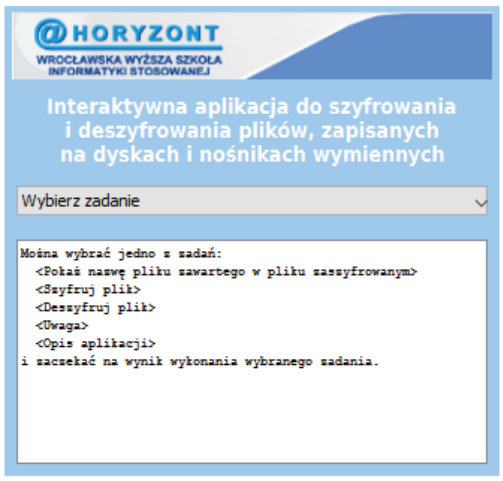
Procedura **fne** szyfruje a procedura **fnd** deszyfruje nazwę pliku, natomiast procedury **af2sf** i **sf2af** wykonują szyfrowanie/deszyfrowanie zawartości pliku. Dlatego też tajny klucz, „zainstalowany" w aplikacji, ma dwie składowe: **kn** - klucz dla procedur **fne** i **fnd**, i **kf** - klucz dla procedur **af2sf** i **sf2af**. Długość tych kluczy w bitach można obliczyć za pomocą instrukcji

```
>vf:=(b-1)*nops(convert(convert(fn,bytes),base,128,6))!;
  kn:=nops(convert(vf,base,2));
  kf:= nops(convert((b-1)*ifs!,base,2));
```

gdzie **fn** oznacza nazwę pliku wybranego do zaszyfrowania a **ifs** rozmiar tego pliku w bajtach. Największe znaczenie dla kryptograficznej ochrony pliku ma klucz **kf**, dlatego nie warto stosować dużych wartości zmiennej **b**, ponieważ wzrost tej zmiennej powoduje zarówno wzrost rozmiaru pliku zaszyfrowanego jak i nieznaczny wzrost bitów klucza, zaś liczba bitów klucza **kf** wzrasta bardzo szybko ze wzrostem **ifs**.

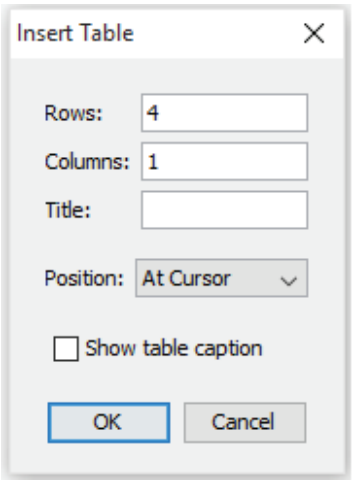
2. Graficzny interfejs użytkownika aplikacji

Po otwarciu programu **sdpdnw.mw** w sesji Maple zostaje wyświetlony graficzny interfejs użytkownika pokazany na Ryc. 3. Taki interfejs można z łatwością skonstruować przy pomocy palety **`Components`**. Należy uruchomić program Maple i najpierw kliknąć na belce narzędzi **File/ New/Document Mode** i do wyświetlonego szablonu pokazanego na Ryc. 4



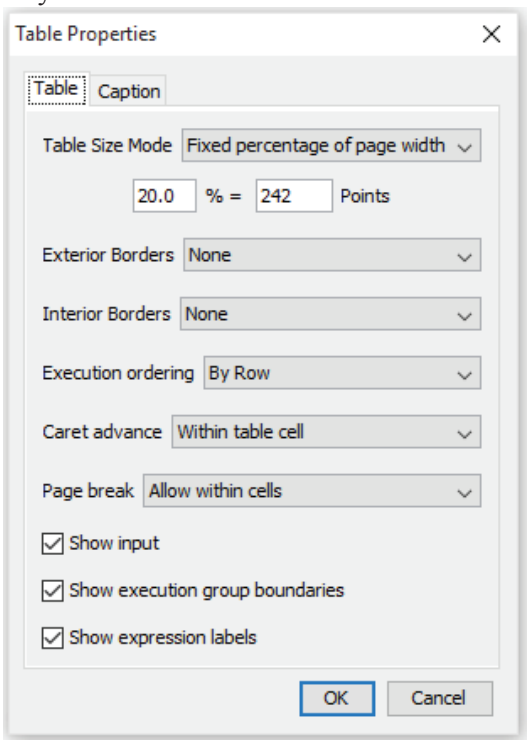
Ryc. 3. Graficzny interfejs użytkownika programu sdpdnw.mw.

należy wpisać ile elementów ma zawierać tworzona tablica.



Ryc. 4. Szablon ustalania liczby wierszy i liczby kolumn tablicy graficznego interfejsu.

Tablicę trzeba zwymiarować, używając szablonu pokazanego na Ryc. 5.



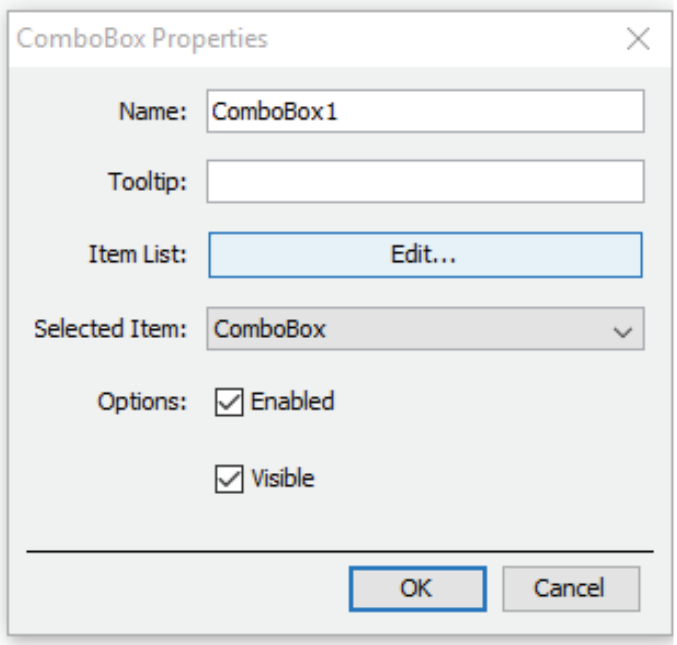
Ryc. 5. Szablon wymiarowania rozmiarów tablicy tworzącej graficzny interfejs.

Do pierwszego wiersza tablicy można wstawić dowolny „obrazek" a w drugim wierszu należy wpisać tytuł aplikacji natomiast do trzeciego wiersza trzeba ściągnąć z palety **`Components`** element o nazwie **`Combo Box`**. Do czwartego wiersza z palety ściąga się element **`Text Area`** wyznaczając liczbę znaków w wierszu i liczbę wierszy:

`Visible Character Width:` 60
`Visible Rows:` 12

Ostatecznie ustala się własności elementu `Combo Box` edytując listę zadań:

`Pokaż nazwę pliku zawartego w pliku zaszyfrowanym`
`Szyfruj plik`
`Deszyfruj plik`
`Uwaga`
`Opis aplikacji`



Ryc. 6. Szablon edycji listy zadań elementu `Combo Box`.

przy pomocy szablonu pokazanego na Ryc. 6.

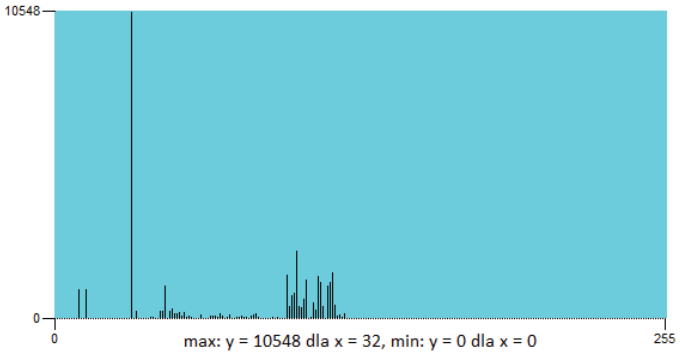
3. Kod źródłowy aplikacji

Kod źródłowy aplikacji umieszczony jest w obszarze kodu startowego i w obszarze wyboru zadań elementu **`Combo Box`**. W obszarze kodu startowego są instrukcje przypisania wartości zmiennym **sfs**, **sfn** i **b** oraz instrukcje procedur **sf2ed**, **fr**, **af2sf**, **sf2af**, **fne** i **fnd**. Pierwsza procedura umożliwia wybranie pliku do szyfrowania lub deszyfrowania, druga pozwala usunąć niepotrzebny plik, trzecia szyfruje wybrany plik, czwarta wybrany plik deszyfruje, piąta szyfruje nazwę pliku wybranego do zaszyfrowania i ostatnia nazwę zaszyfrowanego pliku deszyfruje. Nazwą zaszyfrowanego pliku jest zaszyfrowana nazwa pliku wejściowego procedury szyfrowania. Dzięki temu nie wiadomo co zawiera plik zaszyfrowany. Strukturę algorytmów służących do szyfrowania/deszyfrowania można łatwo prześledzić przeglądając kod procedur **af2sf**, **sf2af**, **fne** i **fnd**. W procedurach **fne** i **fnd** zastosowano funkcję biblioteczną **convert/base** jako przekształcenie kryptograficzne. Przekształceniami kryptograficznymi w procedurze **af2sf** są funkcja **convert/base** i dodawanie liczb naturalnych. W procedurze **sf2af** zastosowano również funkcję **convert/base** jako przekształcenie kryptograficzne a drugim przekształceniem kryptograficznym jest tu odejmowanie liczb naturalnych. Zaszyfrowany plik jest wyjątkowo skutecznie chroniony przed nieupoważnionym dostępem, ponieważ przestrzeń tajnego klucza jest ogromna i zależy od rozmiaru pliku wejściowego. Plik zaszyfrowany jest zapamiętywany w tym samym folderze co plik niezaszyfrowany.

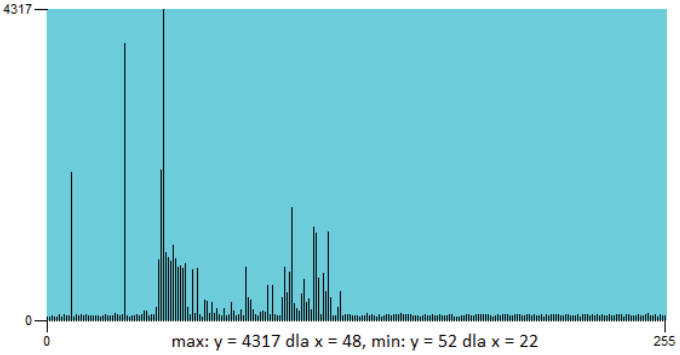
Graficzny interfejs użytkownika umożliwia wybranie trzech podstawowych zadań: **`Szyfrowanie pliku`**, **`Deszyfrowanie pliku`** i **`Deszyfrowanie nazwy pliku zawartego w pliku zaszyfrowanym`**. Po wykonaniu wybranego zadania w obszarze tekstowym interfejsu użytkownik otrzymuje wyczerpującą informację o wykonanym zadaniu, a plik wejściowy zostaje usunięty. Należy pamiętać, że tajny klucz jest zawarty w kodzie aplikacji i każdy użytkownik może na wiele sposobów zamontować swój własny klucz. Najprościej może on na przykład zmienić wartości zmiennych **b**, **sfn** i **sfs**. Poza tym aplikacja musi mieć prawo do zapisywania i usuwania przetwarzanych plików.
Drugą część kodu zawiera element **`Combo Box`**. Znajdują się tam przede wszystkim instrukcje warunkowe uzależnione od interakcji użytkownika, który wybiera określone zadanie do wykonania. Program jest dosyć skomplikowany i po oszczędnym wylistowaniu zajmuje cztery strony instrukcji języka Maple. Mimo to aplikację mogą używać nie tylko biegli znawcy programu Maple, ale też początkujący użytkownicy tego programu. Ze względów bezpieczeństwa aplikacja powinna być zapisana na pendrajwie, który należy skutecznie pilnować.

4. Przykład

Do zilustrowania działania aplikacji wybrano dwa pliki[2] dostępne w internecie: **plik rfc4648.txt** o rozmiarze 36496 bajtów i plik **rfc4648.pdf**, którego rozmiar wynosi 56198 bajtów.



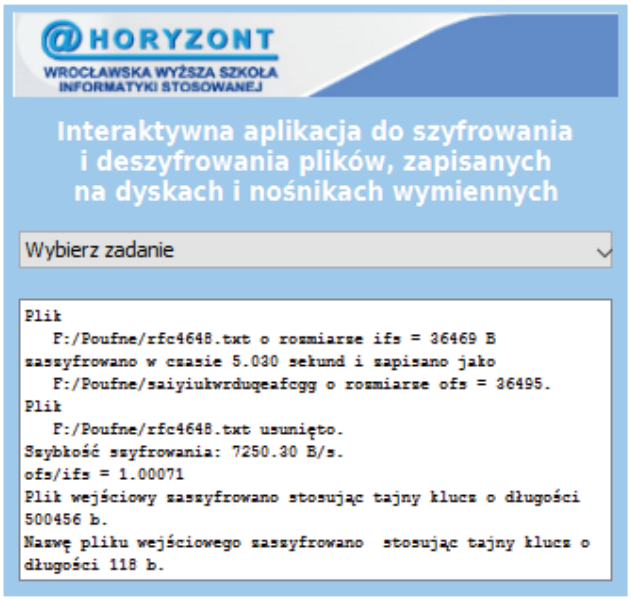
Ryc. 7. Histogram częstotliwości występowania znaków w pliku **rfc4648.txt**.



Ryc. 8. Histogram częstotliwości występowania znaków w pliku **rfc4648.pdf**.

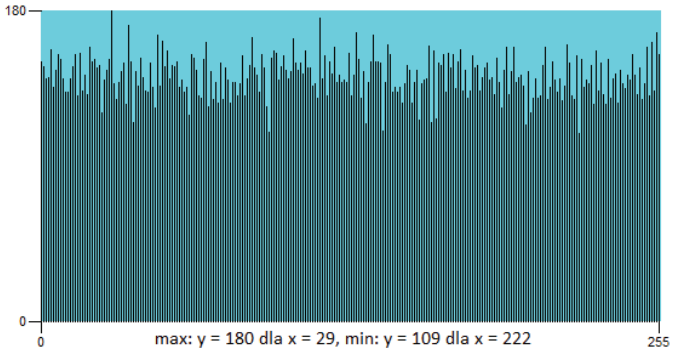
2. S. Josefsson, The Base 16, Base 32, and Base 64 Data Encodings, http://www.o-nline.com/nettools/rfc/rfcs/rfc4648.shtml.

Pierwszy plik zawiera tylko znaki 7-bitowe, w pliku drugim występują wszystkie znaki 8-bitowe. Są to więc dwa pliki różniące się znacznie strukturą znakową.
Najpierw zaszyfrowano za pomocą prezentowanej aplikacji plik tekstowy. Po wykonaniu zadania w polu tekstowym interfejsu pojawia się szczegółowy opis procesu szyfrowania. Podana jest m. in. wielkość tajnego klucza, posiadającego długość 500 456 bitów, przy pomocy którego plik został zaszyfrowany.



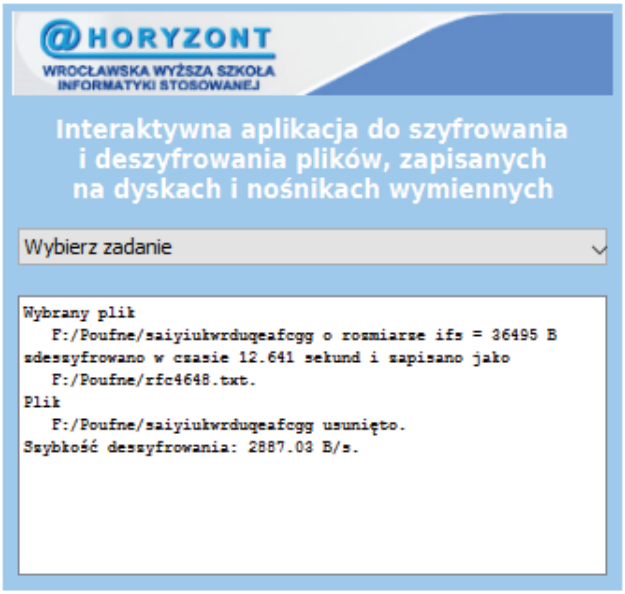
Ryc. 9. Graficzny interfejs użytkownika po wykonaniu zadania szyfrowania pliku rfc4648.txt.

Strukturę znakową zaszyfrowanego pliku ilustruje Ryc. 10. Można zauważyć, że histogramy częstotliwości występowania znaków dla pliku niezaszyfrowanego i zaszyfrowanego znacznie się różnią. Plik zaszyfrowany zawiera dosyć równomierny pseudolosowy ciąg znaków o wartościach bajtowych w zakresie



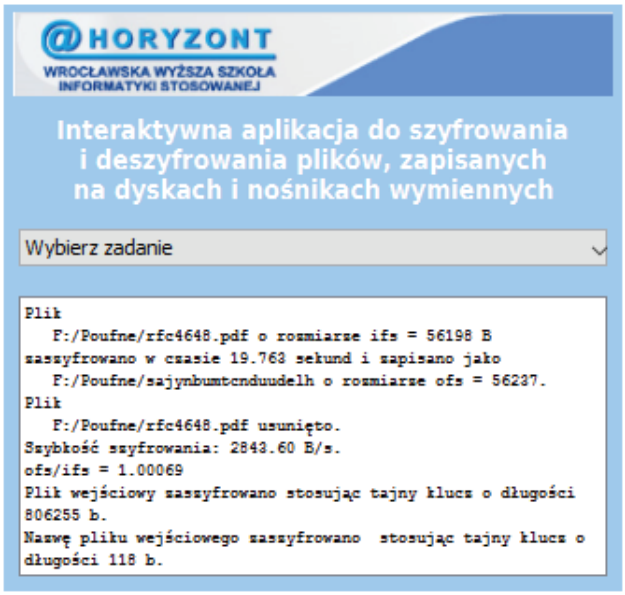
Ryc. 10. Histogram częstotliwości występowania znaków w pliku zaszyfrowanym saiyiukwrduqeafcgg.

0 .. 255. Podobnie po wykonaniu procesu deszyfrowania w polu tekstowym interfejsu można zobaczyć jaki jest rozmiar zaszyfrowanego pliku, jaką nazwę ma plik zdeszyfrowany i jaka jest szybkość deszyfrowania w bajtach/s (Ryc. 11.).

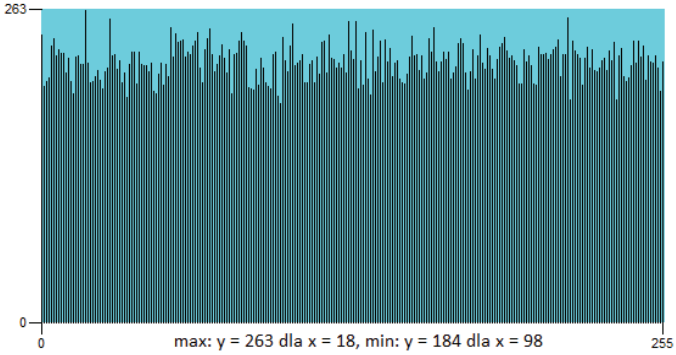


Ryc. 11. Graficzny interfejs użytkownika po wykonaniu zadania deszyfrowania pliku saiyiukwrduqeafcgg.

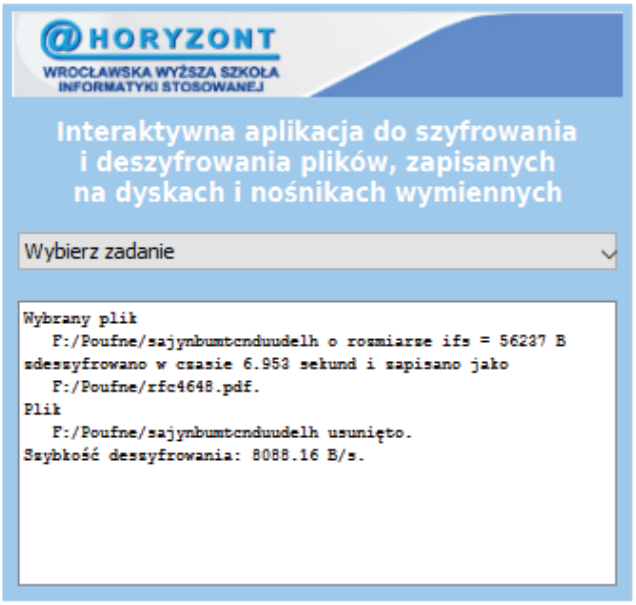
Proces szyfrowania i deszyfrowania drugiego pliku można prześledzić na Ryc. od 12 do 14.



Ryc. 12. Graficzny interfejs użytkownika po wykonaniu zadania szyfrowania pliku **rfc4648.pdf**.



Ryc. 13. Histogram częstotliwości występowania znaków w pliku zaszyfrowanym pliku **sajynbumtcnduudelh**.



Ryc. 14. Graficzny interfejs użytkownika po wykonaniu zadania deszyfrowania pliku **sajynbumtcnduudelh**.

Mimo że wejściowe pliki tekstowe mają bardzo różną strukturę, ich kryptogramy są bardzo podobne. Fakt ten świadczy o dużej mocy szyfru, zastosowanego w procedurach **af2sf** i **sf2af**.

4. Podsumowanie i wnioski

Opisano realizację stosunkowo prostych algorytmów kryptograficznych w których pozycyjny zapis liczb naturalnych o dowolnej wartości bazy jest przekształceniem kryptograficznym. Inaczej mówiąc, zrealizowano w środowisku Maple aplikację stosującą funkcję biblioteczną **convert/base** w procedurach szyfrujących i deszyfrujących, która „potrafi" zaszyfrować każdy plik przy zastosowaniu tajnego klucza o ogromnej długości, dzięki czemu plik jest praktycznie stuprocentowo chroniony przed nieupoważnionym dostępem. Poza tym aplikacja jest wyjątkowo „życzliwa" dla użytkownika, który posługuje się tylko myszą i nie wprowadza żadnych danych, ponieważ tajne klucze kryptograficzne są zawarte w kodzie aplikacji. Takie podejście może być źródłem bardzo wielu podobnych rozwiązań[3, 4, 5]. W pracy nie podano kodu programu typu worksheet o nazwie sdpdnw.mw, aby nie zwiększać objętości artykułu.

Literatura

[1] C. Kościelny, Realizacja szyfru „bezkluczowego" c80k395 do kryptograficznej ochrony załączników poczty elektronicznej w środowisku Maple, Biuletyn Naukowy Wrocławskiej Wyższej Szkoły Informatyki Stosowanej. Informatyka, 2013 3.

3. Maple Implementation of Transport Encryption Scheme Using the Secret Key of Length 479 Bits, http://www.maplesoft.com/applications/Author.aspx?mid=1638.
4. Base 64 "Keyless" File Encryption, http://www.maplesoft.com/aplications/Author.aspx?mid=16738.
5. Maple `Keyless` Base b Encryption Scheme,_http://www.maplesoft.com/applications/Author.aspx?mid=16738.