STUDIA INFORMATICA 2000
Volume 21 Number 1 (39)

Jarosław FRANCIK, Katarzyna TRYBICKA-FRANCIK
Politechnika Śląska, Instytut Informatyki

ZASTOSOWANIE WIZUALIZACJI ALGORYTMÓW W KRYPTOLOGII

Streszczenie. Analiza i projektowanie algorytmów stanowi jeden z ważnych obszarów zastosowań wizualizacji algorytmów. Niniejszy artykuł przedstawia proces implementacji nieskomplikowanego szyfratora i deszyfratora korzystającego z prostego szyfru podstawieniowego, a następnie stworzenia oprogramowania skutecznie łamiącego ten szyfr. To nieskomplikowane zadanie posłużyło za przykład zastosowania wizualizacji algorytmów, którą wykorzystano w procesie doboru najwłaściwszej metody statystycznej analizy danych. Wizualizacji poddano też bardziej zaawansowane algorytmy kryptograficzne.

APPLYING ALGORITHM VISUALIZATION IN CRYPTOLOGY

Summary. Algorithm analysis and design is one of the most important areas of application of algorithm visualization. This paper presents implementation of a crypting and decrypting software, which uses a simple substitution cipher, and then creating a tool for cryptanalysis of this cipher. This not very complicated task was a case study of using software visualization in the area. It was applied to choose the best method of statistic analysis of linguistic data. Visualization has been also used to render more advanced algorithms.

1. Wstęp

Kilkanaście ostatnich lat rozwoju informatyki było okresem bezprecedensowego rozwoju rozwiązań graficznych w zakresie pośrednictwa między człowiekiem i komputerem. Wizualizacja oprogramowania [1, 2, 3] jest wyrazem tej tendencji w zakresie konstrukcji oprogramowania i dydaktyki informatyki. Umożliwia ona wgląd w sposób działania i zachowania się algorytmów, czy szerzej – oprogramowania, używając języka najbardziej zrozumiałego dla człowieka – języka obrazów. Pozwala dostrzec wiele aspektów działania algorytmów trudno za-

uważalnych w inny sposób, czy wręcz niemal całkiem nieuchwytnych. W ciągu minionych lat wizualizacja algorytmów i programów przyciągała sporo uwagi. Stworzono wiele systemów dostarczających środki do jej realizacji. Przedstawione w niniejszym artykule wizualizacje algorytmów kryptograficznych wykonano za pomocą systemu animacji algorytmów *Daphnis* [3, 4, 5], stworzonego przez pierwszego z autorów w ramach pracy doktorskiej.

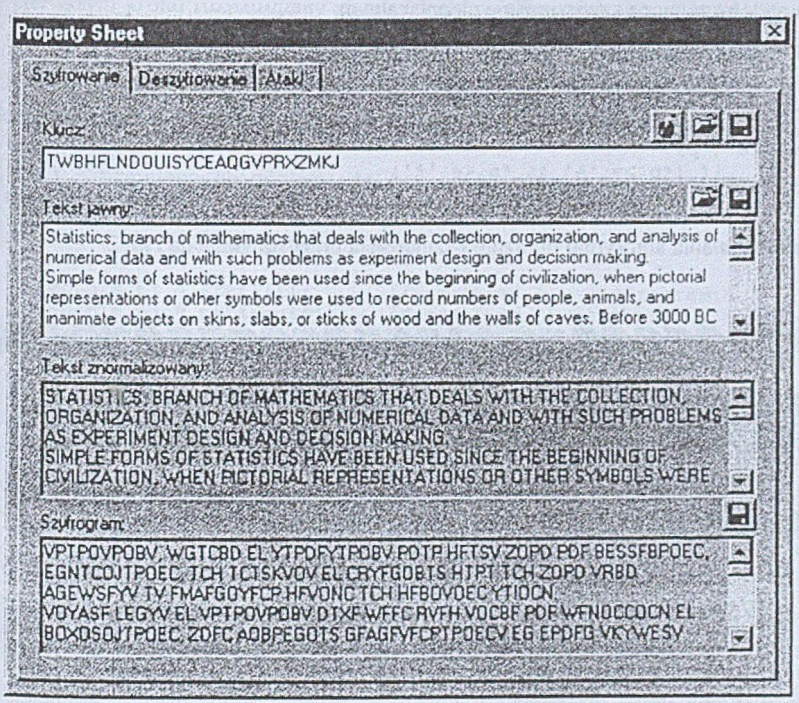
W ramach prac związanych z kryptografią drugi ze współautorów tego artykułu od niedawna stosuje techniki wizualizacyjne. W niniejszym artykule prezentujemy wprawkę o charakterze *case study*, przedstawiającą proces implementacji nieskomplikowanego szyfratora i deszyfratora korzystającego z prostego szyfru podstawieniowego, a następnie stworzenia narzędzia skutecznie łamiącego ten szyfr. Wizualizacja algorytmów została zastosowana nie tylko dla zilustrowania opisywanych zagadnień. W procesie doboru najwłaściwszej metody statystycznej analizy danych, zastosowanej do złamania szyfru, czytelne przedstawienie graficzne wyników analizy pozwoliło dostrzec zależności bardzo trudne do wychwycenia bez tego typu narzędzi. Wizualizacja algorytmów pomogła w tym przypadku zweryfikować różne techniki, jakie były brane pod uwagę w trakcie implementacji, a także ocenić wpływ takich czynników, jak jakość wzorcowej statystyki języka naturalnego.

2. Szyfry podstawieniowe

Szyfry podstawieniowe należą do klasyki w kryptografii: dość wspomnieć sławny szyfr Cezara, tzw. szyfr cmentarny, który od czasu, gdy wygrawerowano go w 1794 roku na płycie grobowca na cmentarzu Świętej Trójcy w Nowym Jorku przez ponad 100 lat opierał się próbom złamania, czy wreszcie szyfr Enigmy. Mimo to nie są one trudne do złamania. Żaden z szyfrów tej klasy nie stanowi prawdziwego wyzwania dla współczesnego kryptoanalityka [6, 7]. Opisywany w niniejszym opracowaniu szyfr również jest łatwy do złamania, stanowi jednak idealny – niezbyt skomplikowany – pretekst do tego, by zastosować technikę wizualizacji algorytmów w kryptografii i kryptoanalizie. W tym celu stworzone zostało oprogramowanie szyfrujące, deszyfrujące i łamiące szyfr podstawieniowy, z możliwością wizualizacji zastosowanych algorytmów (rys. 1). Wszystkie wizualizacje prezentowane w niniejszym opracowaniu przygotowano z pomocą systemu wizualizacji algorytmów Daphnis [5].

Zaimplementowany szyfr jest szyfrem monoalfabetycznym, w którym każdemu znakowi alfabetu przyporządkowano szyfrujący go znak. Ponieważ przyporządkowanie to może mieć w zasadzie charakter dowolnej permutacji znaków alfabetu, szyfr ten jest znacznie silniejszy od szyfru Cezara. Podobnie, jak w przypadku wspomnianego już szyfru cmentarnego, wymaga on łamania z użyciem analizy statystycznej częstości występowania znaków. Ta klasyczna

metoda kryptoanalizy (bez znajomości tekstu jawnego) znalazła swój opis w opowiadaniu Edgara Allana Poe *Złoty żuk* [8].



Rys. 1. Prosta aplikacja do szyfrowania, deszyfrowania i łamania szyfru
Fig. 1. Simple application for crypting, decrypting and cryptanalysis

3. Algorytm szyfrowania i deszyfrowania

Przyjęto upraszczające założenie, że szyfrowaniu (i deszyfrowaniu) podlegają jedynie litery. Wszystkie pozostałe znaki są kopiowane do szyfrogramu bez zmian. Dodatkowo, wszystkie litery tekstu są przed szyfrowaniem zamieniane na wielkie. Ogranicza to rozmiar alfabetu do 26 liter alfabetu angielskiego (litery charakterystyczne dla języka polskiego są zamieniane na najbliższe im odpowiedniki łacińskie). Ponieważ dopuszczono możliwość dowolnego permutowania podstawień liter alfabetu jawnego, kluczem szyfru jest ciąg 26 znaków (liter) $k_1..k_{26}$, definiujących podstawienia dla poszczególnych liter alfabetu. Każdej literze tek-

stu jawnego odpowiada w szyfrogramie litera wyznaczona przez element klucza k_i, gdzie i jest numerem porządkowym litery (A = 1, B = 2 itd.). Liczba możliwych kluczy $k_1..k_{26}$ wynosi:

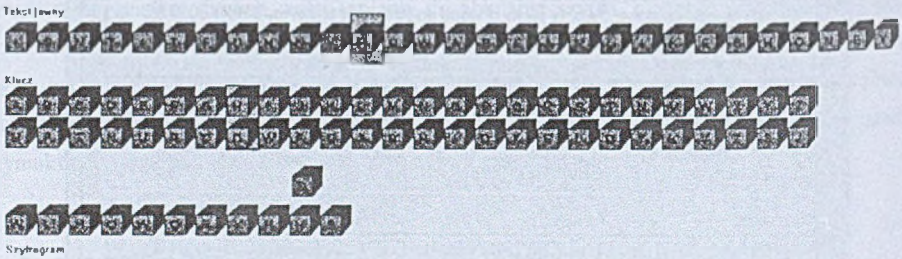
$$N = 26! \approx 4 \cdot 10^{26}$$

co czyni atak brutalny na szyfr zupełnie nieopłacalnym.

Algorytm szyfrujący można w języku C zapisać następująco:

```
// char *p - tekst poddawany szyfrowaniu
// char *k - klucz 26-literowy
while (*p)
      if (*p >= 'A' && *p <= 'Z')
            *p = k[*p - 'A'];
```

Zasadę działania algorytmu obrazuje wizualizacja przedstawiona na rys. 2.



Rys. 2. Kadr z projekcji algorytmu szyfrowania: szyfrowanie litery „H"
Fig. 2. Scene from a crypting algorithm animation: crypting letter "H"

Deszyfrowanie może przebiegać według tego samego algorytmu, po zamianie klucza $k_1..k_{26}$ na odpowiadający mu klucz odwrotny $k'_1..k'_{26}$, w którym znak k'_i jest literą o numerze porządkowym równym indeksowi litery o numerze porządkowym i w kluczu $k_1..k_{26}$.

4. Algorytm łamania szyfru

Algorytm łamania szyfru oparty jest na analizie porównawczej tzw. statystyk, to znaczy zbiorów informujących o względnej częstotliwości występowania danego znaku lub zbitki znaków w tekście. Pod uwagę bierze się:

- Statystykę wzorcową, obliczoną dla typowego i odpowiednio długiego tekstu zapisanego najlepiej w tym samym języku, w którym prawdopodobnie sformułowany jest poszukiwany tekst.
- Statystykę szyfrogramu.

Zakłada się, że częstotliwość występowania znaku w statystyce wzorcowej jest zbliżona do częstotliwości występowania odpowiadającego jej zaszyfrowanego znaku w statystyce

szyfrogramu. Fakt, że różne teksty napisane w tym samym języku mają zbliżone, ale jednak na ogół nieco odmienne statystyki, stanowi podstawową trudność algorytmu. Należy więc tak dobrać pary znak wzorca – znak szyfrogramu, by zgodność częstotliwości była maksymalna.

W pierwszym kroku rozpatrujemy statystyki dla pojedynczych liter. Dla każdej pary liter (x, y) rozważamy prawdopodobieństwo zdarzenia, że znak x w szyfrogramie odpowiada znakowi y w tekście jawnym (i wzorcowym). Wartości będące miarą prawdopodobieństw – współczynniki korelacji częstości – zachowujemy w tablicy dwuwymiarowej:

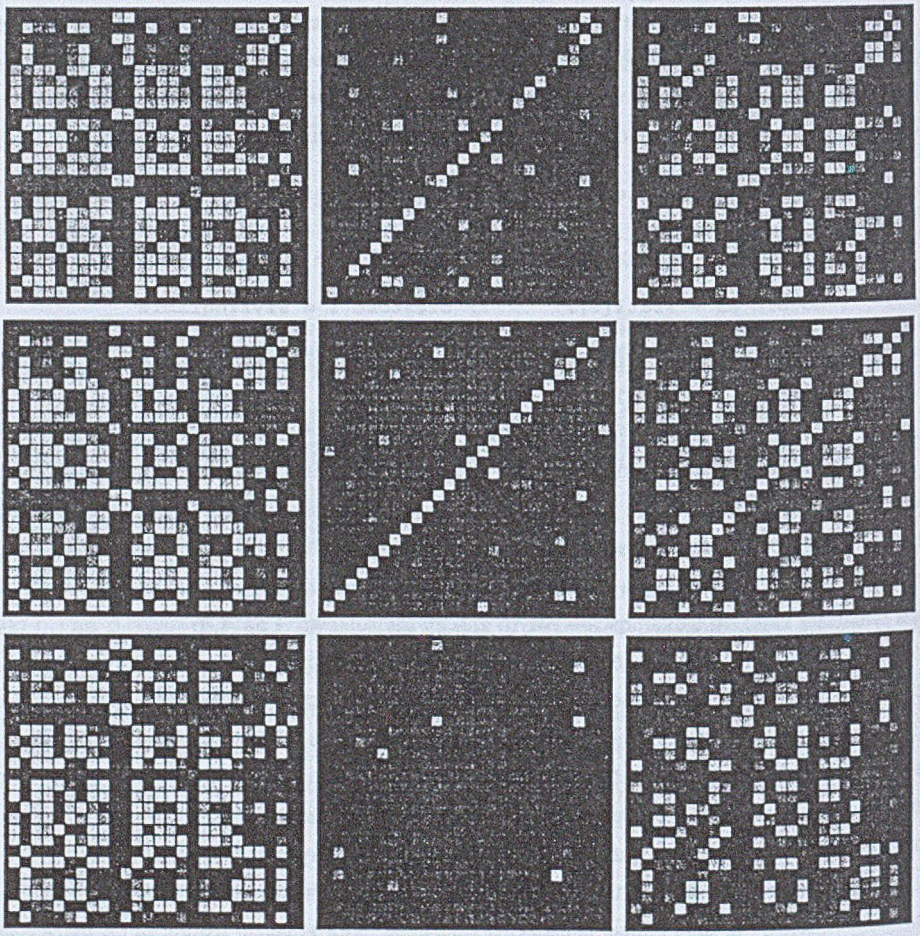
```
// M[i][j] - prawdopodobieństwo, że i w szyfrogramie to j w tekście jawnym
double M[26][26];
  for (i = 0; i < 26; i++)
     for (j = 0; j < 26; j++)
     {
        double pi = stat1[i];        // częstość i w szyfrogramie
        double pj = pattern1[j];     // częstość j we wzorcu
        ASSERT(pj != 0);             // statystyka wzorcowa nie powinna
                                     // zawierać pozycji o zerowym częst.
        M[i][j] = pi > pj ? pj / pi : pi / pj;
     }
```

Powyższa tablica stała się interesującym przedmiotem wizualizacji (rys. 3, kolumna po lewej). Wartości poszczególnych elementów tablicy są kodowane barwnie – ciepłe kolory (na rys. 3 – jaśniejsze) oznaczają wysokie prawdopodobieństwo, że pozycja tablicy odpowiada rzeczywistej parze znak wzorca – znak szyfrogramu. Tablica w górnej części rys. 3 przedstawia sytuację z kluczem tożsamościowym, to znaczy szyfrującym A na A, B na B, C na C i tak dalej. Właściwe pary grupują się wówczas na przekątnej tablicy – zauważalna jest czerwona, skośna pręga (na rysunku widoczna jako jaśniejsza). Na tablicy w środkowej części rysunku (pionowo) pręga jest szczególnie widoczna. W tym przypadku tekst wzorcowy i tekst jawny szyfrogramu były identyczne, zatem wszystkie częstości występowania odpowiadających sobie znaków były identyczne – intensywnie czerwone. W dolnej części widnieje przypadek z nietożsamościowym kluczem – właściwe pary rozrzucone są chaotycznie w obrębie tablicy.

Zastosowanie – na etapie uruchamiania algorytmu – wizualizacji wyników analizy dla klucza tożsamościowego daje okazję zobaczenia zależności, które inaczej trudne są do uchwycenia. Przede wszystkim efektywność (pewność) rozwiązania przekłada się wprost na obecność i wyrazistość czerwonej (jasnej) pręgi, biegnącej po przekątnej obrazu tablicy.

Wyniki wizualizacji utwierdzają w przekonaniu, że potrzebny jest dalszy krok algorytmu, pozwalający na skuteczniejsze wyodrębnienie właściwych par z tła. Krokiem tym jest analiza porównawcza statystyk wyodrębnionych dla digramów – dwuliterowych ciągów literowych. Każda z liter może wystąpić w 52 różnych digramach. Dla każdej z 26·26 możliwych par należy porównać dwa zestawy 52 częstości występowania digramów. Tablice przedstawione na rys. 3 w środkowej kolumnie obrazują otrzymany w ten sposób rozkład prawdopodobieństwa.

Tablice znajdujące się w prawej kolumnie przedstawiają kombinację rozkładów prawdopodobieństwa obliczonych na podstawie pojedynczych liter oraz digramów.



Rys. 3. Wizualizacja rozkładów prawdopodobieństwa dla możliwych wszystkich par znak wzorca – znak szyfrogramu. Od góry w dół: szyfr z kluczem tożsamościowym; przypadek, w którym tekst wzorcowy i szyfrowany są identyczne; przypadek ogólny (z kluczem nietożsamościowym). Od lewej do prawej: wyniki analizy częstości pojedynczych znaków, digramów i kombinacja obydwu

Fig. 3. Visualization of probability distribution for each possible pair character of pattern – character of cipher. From top to down: cipher with an identity key; case of identical pattern text and crypted text; general case (with non-identity key). From left to right: results of analysis for single characters, pairs of characters (digrams) and a combination of both

W trakcie pracy ze zwizualizowanym obrazem tablic prawdopodobieństw udało się dobrać właściwy sposób porównania zestawów 52 częstości występowania digramów. Oprócz średniej arytmetycznej i wariancji rozpatruje się też współczynniki korelacji częstotliwości występowania digramów, traktując obydwa 52-elementowe zestawy jak posortowane wektory. Inne sposoby porównywania, oparte na znajdowaniu ekstremów lub obliczaniu średniej odległości między elementami zbioru, nie zostały zaakceptowane, gdyż nie dawały zadowalającego obrazu w trakcie wizualizacji.

Ostatnim krokiem algorytmu był wybór z tablicy rozkładu prawdopodobieństw (prawa kolumna na rys. 3) par o maksymalnym obliczonym prawdopodobieństwie.

5. Wyniki

Stwierdzono wysoki stopień korelacji pomiędzy przewidywaniami opartymi na wnioskach z wizualizacji a ich faktyczną weryfikacją, otrzymaną poprzez deszyfrowanie szyfrogramu kluczem uzyskanym z łamania. Tylko wtedy, gdy na obrazie widoczna była wyraźna, skośna, czerwona pręga biegnąca po przekątnej tablicy, wyniki deszyfracji były czytelne. Pozytywna weryfikacja algorytmu zależała od dwóch czynników. Pierwszym był właściwy dobór algorytmu. Algorytm nie analizujący częstości występowania digramów lub analizujący je niewłaściwymi metodami nie dawał czytelnych wyników. Ogromne znaczenie miała też jakość i wielkość zastosowanej statystyki wzorcowej: zmiana wzorca z pliku o długości 10 kb na plik o długości 65 kb znacząco ulepszyła niezawodność programu.

W testach praktycznych metoda uzyskiwała stuprocentową pewność w przypadku łamania szyfrogramu będącego jednocześnie tekstem wzorcowym dla statystyki (jest to przypadek przedstawiony na rys. 3 w środkowym rzędzie). W typowych przypadkach, z nietożsamościowym kluczem i przypadkowym tekstem, algorytm pozwalał na trafne odgadnięcie 15-20 liter (w 26-literowym alfabecie). Pozwalało to uzyskać całkiem czytelny zdeszyfrowany tekst, gdyż pozostałe, nietrafne dopasowania dotyczyły przede wszystkim liter stosunkowo rzadziej występujących. Generalnie, nieco lepsze wyniki uzyskuje się „łamiąc" trywialny szyfr z tożsamościowym kluczem, choć efekt ten trudno jest wyjaśnić.

A oto przykład: szyfrowano hasło z anglojęzycznej encyklopedii multimedialnej *Encarta*. Tak wyglądał początkowy fragment tekstu jawnego:

> *Statistics, branch of mathematics that deals with the collection, organization, and analysis of numerical data and with such problems as experiment design and decision making.*

Poniżej przytaczamy szyfrogram:

MVYVIMVIJM, CUYOJH FP KYVHBKYVIJM VHYV QBYLM AIVH VHB JFLLBJVIFO, FUDYOISYVIFO, YOQ YOYLGMIM FP OWKBUIJYL QYVY YOQ AIVH MWJH RUFCLBKM YM BXRBUIKBOV QBMIDO YOQ QBJIMIFO KYZIOD.

A oto odtworzony tekst:

STATISTICS, PRANCH OF MATHEMATICS THAT DEALS WITH THE COLLECTION, ORGANIXATION, AND ANALBSIS OF NYMERICAL DATA AND WITH SYCH UROPLEMS AS EKUERIMENT DESIGN AND DECISION MAJING.

Przytoczymy jeszcze klucze, w górnej linijce oryginalnie zastosowany do szyfrowania, w dolnej – odtworzony na drodze łamania szyfru:

YCJQBPDHINZLKOFRTUMVWEAXGS
YGJQBPDHIZXLKOFCNUMVREASWT

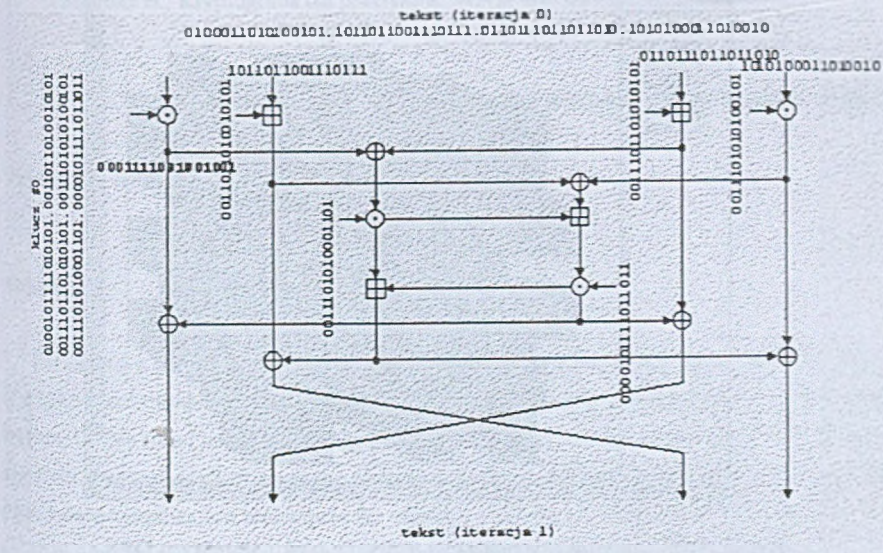
6. Wizualizacja algorytmów DES i IDEA

Długofalowym celem naszych prac jest wykorzystanie wizualizacji algorytmów do oceny jakości algorytmów kryptograficznych i ich implementacji. Oczywiście chodzi tu o bardziej zaawansowane, „poważne" algorytmy szyfrujące, jak DES (potrójny DES) czy IDEA.

Pierwszym krokiem jest zbadanie konkretnej implementacji celem stwierdzenia, czy i w jakim stopniu jest ona zgodna z oczekiwaniami. Techniki wizualizacyjne mogą tu grać rolę „wizualnego debuggera". Przykładem wizualizacji tego typu jest projekcja działania algorytmu IDEA [9] (rys. 4). Pozwala ona na drobiazgowe prześledzenie działania algorytmu i wartości wyników pośrednich. Poza walorem badawczym i inżynieryjnym przedstawiana animacja może mieć zastosowanie edukacyjne.

Znacznie ciekawsze mogą się okazać projekcje przedstawiające dane w wysokim stopniu przetworzone – w wizualizacji oprogramowania mówimy o *wysokim poziomie abstrakcji* [2, 3]. Pierwszą naszą próbą w tym zakresie była trójwymiarowa wizualizacja s-bloków algorytmu DES (rys. 5). Są to struktury o krytycznym znaczeniu w przypadku ataku metodą kryptoanalizy różnicowej lub liniowej [6, 7]. Wprawdzie algorytm DES został już złamany metodą brutalną, jednak takie podejście nie jest – na razie – wystarczające dla potrójnego DES, a zatem analiza s-bloków jest w dalszym ciągu potencjalnie obiecującym kierunkiem badań. W praktyce jednak s-bloki są opracowane tak dobrze, że prawdopodobnie jeszcze długo będą się opierać wysiłkom kryptoanalityków. Nasza wizualizacja zdołała jedynie potwierdzić ten fakt – oglądając z różnych stron trójwymiarowe obiekty będące modelami kolejnych s-bloków można ujrzeć na własne oczy fascynujące, matematyczne piękno tych struktur, wprost doskonale

chaotycznych, piękno, które normalnie kryje się pośród szeregów zer i jedynek w podręcznikach kryptografii.

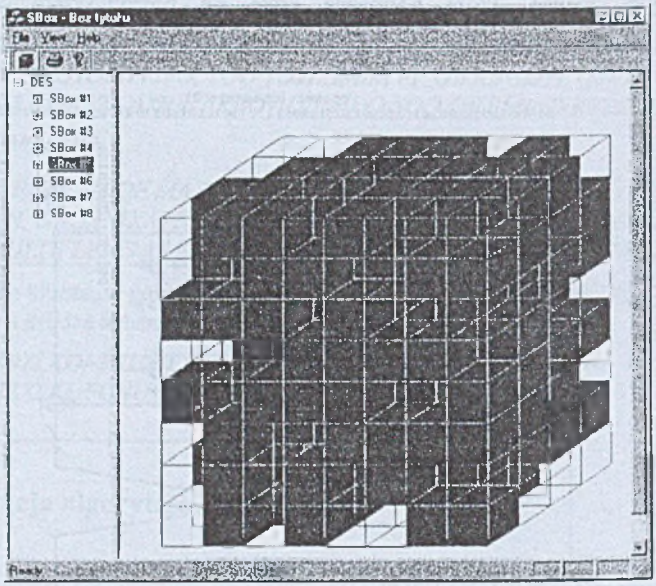


Rys. 4. Wizualizacja algorytmu szyfrującego IDEA
Fig. 4. Visualization of the IDEA crypting algorithm

7. Wnioski i kierunki przyszłych badań

Przedstawiony w opracowaniu algorytm łamania szyfru podstawieniowego nie ma, być może, zbyt dużego znaczenia praktycznego, jednak wsparcie, jakie uzyskano w trakcie jego implementacji ze strony narzędzi wizualizacji algorytmów, obiecuje sporo na przyszłość. Wykrywanie normalnie trudno zauważalnych niuansów w analizowanych strukturach staje się łatwe i intuicyjnie wyczuwalne podczas projekcji algorytmu. Skłoniło nas to do stworzenia projekcji bardziej zaawansowanych algorytmów (DES i IDEA).

W najbliższej przyszłości zastosujemy projekcje o wyższym poziomie abstrakcji – przedstawiające dane analityczne otrzymane z badanych algorytmów. Dane te, z natury rzeczy złożone, po przetworzeniu i przedstawieniu za pomocą środków wizualnych mogą odsłonić wiele właściwości trudnych do wychwycenia na drodze klasycznej analizy.



Rys. 5. Trójwymiarowa wizualizacja piątego s-bloku algorytmu DES
Fig. 5. Three-dimensional visualization of the fifth s-box of DES algorithm

LITERATURA

1. Stasko J. T., Domingue J.B., Brown M. H., Price B. A. (red.) Software Visualization. Programming as a Multimedia Experience. The MIT Press, Cambridge, MA, USA, 1999. ISBN 0-262-19395-7.
2. Price B. A., Baecker R. M., Small I. S., A Principled Taxonomy of Software Visualization. Journal of Visual Languages and Computing, Vol. 4, 1993, str. 211-266, http://www-cs.open.ac.uk/~doc/jvlc/JVLC-Body.html
3. Francik J.: Śledzenie przepływu danych dla potrzeb animacji algorytmów. Rozprawa doktorska. Politechnika Śląska, Gliwice, Université de Lille, 1999, http://sun.iinf.polsl.gliwice.pl/~jfrancik/publ/dissertation.htm.
4. Francik J.: Surveillance du flux des données dans l'animation des algorithmes. Resumé d'une thèse en co-tutelle. Université de Lille, 1999, http://sun.iinf.polsl.gliwice.pl/~jfrancik/publ/dissertation.htm (also in English).
5. Algorithm Animation Pages. Witryna www. http://sun.iinf.polsl.gliwice.pl/~jfrancik/aa.
6. Grzywak A. (red.), Trybicka-Francik K. (red.), Bezpieczeństwo sieci komputerowych. Skrypt Politechniki Śląskiej nr 2156, Gliwice 1999.

7. Robling Denning D.E., Kryptografia i ochrona danych. WNT, Warszawa 1992.
8. Poe E. A., Opowiadania. Tłumaczył: S. Wyrzykowski. Czytelnik, Warszawa 1986.
9. Schneier B., Kryptografia dla praktyków. WNT, Warszawa 1995.

Recenzent: Dr inż. Ryszard Winiarczyk

Wpłynęło do Redakcji 25 kwietnia 2000 r.

Abstract

Ten or more last years were in computer industry a period of unprecedented development of graphical solutions in human-computer interface. Software visualization [1, 2, 3] is a manifestation of this trend in software engineering and computer science education. One of its most important areas of application is the algorithm analysis and design. The paper presents applying software visualization techniques for creating a simple crypting and decrypting tool (fig. 1) and then attacking it to break the cipher. The software visualization tool that we used was an algorithm animation system *Daphnis* [3, 4, 5]. The crypting method applied in this case study was not very complicated: it was just a simple substitution cipher [6, 7]. The cryptanalysis method was based on statistic analysis of linguistic data [6, 7]. A simple visualization has been prepared to show the idea of the substitution cipher (fig. 2). A more sophisticated projection helped us to verify various approaches taken into consideration during solving the problem and to choose the best method of statistic analysis of linguistic data. On the visualization results presented on fig. 3 the distribution of probability for each possible pair: character of pattern – character of cipher is shown. Red squares (in fig. 3: lighter) represent high probability of correct identification of the character in the cipher key being searched, while blue squares (in fig. 3: darker) represent low probability. In a general case the results are not so good as in special test conditions, but they are still satisfactory: ca 15-20 of 26 characters were recognized correctly. The 6-11 characters that were not recognized are characters relatively rarely used. The finally decrypted (broken) text, although not identical with the source text, is quite readable (see examples in the very end of section 5).

Visualization has been also used to render more advanced algorithms: IDEA [9] (fig. 4) and so called s-boxes in the DES algorithm [6, 7, 9] (fig. 5). The latter is a step towards creating projections with high level of abstraction, that could render properties of highly processed data, which are very difficult to analyze in a traditional way.