Funkcja jednokierunkowa i szyfr strumieniowy VMPC

Bartosz Żółtak

Na CD:

Na płycie CD zamieściliśmy pełną dokumentację dotyczącą funkcji i szyfru VMPC.

Algorytmy szyfrowania podzielone są na dwie główne kategorie: szyfry blokowe i szyfry strumieniowe. Te pierwsze przekształcają bloki bajtów (najczęściej 16), drugie zaś operują na pojedynczych bitach lub bajtach i generują strumień pseudolosowych wartości, który jest dodawany (lub poddawany operacji XOR) z kolejnymi bajtami wiadomości. Dobry szyfr strumieniowy powinien generować ciąg wartości nieodróżnialny od ciągu idealnie losowego. Szyfry blokowe, do jakich zaliczamy poczciwego DES-a, jego następcę AES-a (Rijndael) czy na przykład popularne algorytmy IDEA lub Blowfish, są chronologicznie wcześniejsze. Przykłady szyfrów strumieniowych to bardzo popularny RC4 czy też nowsze algorytmy jak SNOW, Helix czy SOBER oraz właśnie VMPC.

Szyfry strumieniowe w implementacjach programowych są zwykle o wiele szybsze od blokowych i najczęściej charakteryzują się prostszą konstrukcją. Są jednak dużo młodsze od szyfrów blokowych (DES powstał w latach 70, podczas gdy RC4, którego można uznać za ojca szyfrów strumieniowych, dopiero pod koniec lat 80) i dlatego są mniej dogłębnie przebadane. W ostatnich latach zauważyć można jednak znaczny wzrost zainteresowania szyframi strumieniowymi.

Funkcja jednokierunkowa VMPC

U teoretycznych podstaw algorytmów szyfrowania leżą funkcje jednokierunkowe – każdy algorytm, jeśli ma być w praktyce niemożliwy do złamania, musi zawierać przekształcenia jednokierunkowe – takie, których obliczenie jest łatwe, ale których odwrócenie – znalezienie argumentów na podstawie wartości (jak znalezienie wiadomości lub klucza na podstawie szyfrogramu) – jest obliczeniowo niewykonalne.

Bartosz Żółtak jest odkrywcą funkcji jednokierunkowej VMPC oraz autorem opartego na niej szyfru strumieniowego. W lutym 2004 roku zaprezentował VMPC na prestiżowej międzynarodowej konferencji kryptograficznej FSE'04 w Delhi w Indiach. Kontakt z autorem: *bzoltak@vmpcfunction.com*

Funkcja VMPC w Polsce

Prace nad oryginalną, jak się niedawno okazało, funkcją jednokierunkową od kilku lat prowadzone były w Polsce przez Bartosz Żółtaka, absolwenta Politechniki Wrocławskiej. Będąc studentem, w 1998 roku, zaczął zajmować się pewnym przekształceniem matematycznym własnego autorstwa, które, jak się później okazało, nie było nigdzie w literaturze opisane. Przekształcenie było niezwykle proste, a jego *magiczną* cechą, która autora zaintrygowała, było właśnie to, że nie umiał on go odwrócić. Badaniu tego przekształcenia, nazwanego funkcją jednokierunkową VMPC (*Variably Modified Permutation Composition*, zmiennie modyfikowane złożenie permutacji), autor poświęcił ponad pięć lat.

Prace okazały się nadzwyczaj owocne, gdyż autor został w tym roku zaproszony do zaprezentowania VMPC na międzynarodowej konferencji kryptograficznej organizowanej przez Międzynarodowe Stowarzyszenie Badań Kryptologicznych (IACR, *www.iacr.org*), *Fast Software Encryption* 2004 (FSE'04, *www.isical.ac.in/~fse2004*), która odbyła się w Delhi w Indiach w dniach 5-7 lutego 2004. Konferencja ta należy do najbardziej prestiżowych światowych wydarzeń naukowych w kryptografii.

Referat o VMPC został później wygłoszony przez autora także w Polsce – na seminarium kryptologicznym w Instytucie Matematycznym Polskiej Akademii Nauk (*www.impan.gov.pl*) w Warszawie 18 marca 2004 oraz na VIII Krajowej Konferencji Zastosowań Kryptografii Enigma 2004, która odbyła się w dniach 10-13 maja 2004 w Warszawie (*http://www.enigma.com.pl/konferencje/viii_kkzk*).

Funkcja VMPC – jak to działa

Cecha jednokierunkowości funkcji VMPC jest niemal paradoksalna – przekształcenie jednego bajtu informacji wymaga jedynie trzech podstawowych instrukcji `MOV` procesora, zajmujących jeden cykl zegarowy każda, natomiast odwrócenie tak prostego przekształcenia wymaga wykonania średnio około 2260 operacji, co jest liczbą niewyobrażalnie wielką. Uznany dziś za całkowicie wystarczający poziom bezpieczeństwa kryptograficznego to 2128, a więc mniej niż pierwiastek z 2260.

Z matematycznego punktu widzenia funkcja VMPC jest połączeniem podstawowych opera-

cji na permutacjach z podstawowymi operacjami arytmetycznymi na liczbach całkowitych, na przykład z operacją dodawania. Permutacja to nic innego jak ciąg kolejnych liczb, ale *potasowany*, na przykład P=(1, 3, 0, 4, 2) to przykładowa permutacja 5-elementowa ($P[0]=1$, $P[1]=3$, $P[2]=0$, $P[3]=4$, $P[4]=2$).

Podstawową operacją na permutacjach jest ich składanie, które realizowane jest w następujący prosty sposób: w celu uzyskania na przykład zerowego elementu permutacji *złożonego* P, a więc $P[P[0]]$, sprawdzamy jaką wartość ma permutacja P dla indeksu 0 (jest to 1, bo $P[0]=1$), a następnie *przenosimy* tę wartość na indeks i sprawdzamy jaka liczba *stoi* pod indeksem 1 – jest to 3 ($P[1]=3$). Uzyskaliśmy w ten sposób $P[P[0]]=3$. Gdybyśmy dodatkowo wykonali jeszcze jedno przeniesienie – a więc sprawdzili, co stoi pod indeksem 3 (jest to 4, bo $P[3]=4$), uzyskalibyśmy zerowy element potrójnego złożenia permutacji P: $P[P[P[0]]]=4$.

Po wykonaniu analogicznych operacji na wszystkich 5 elementach P otrzymamy wynik podwójnego złożenia permutacji P, permutację $Q2$: $Q2[x]=P[P[x]]=(3, 4, 1, 2, 0)$, natomiast wynikiem potrójnego złożenia P będzie permutacja $Q3$: $Q3[x]=P[P[P[x]]]=(4, 2, 3, 0, 1)$.

Okazuje się, że odwrócenie podwójnego czy potrójnego złożenia permutacji (a więc odnalezienie permutacji P na podstawie $Q2$ czy $Q3$) jest zadaniem bardzo łatwym. Jest tak dlatego, że proste złożenie zachowuje strukturę cykli permutacji P (relacji między wartościami a indeksami), gdyż jedyną operacją, jaką wykonujemy, jest *przenoszenie* wartości na indeks w ramach danej permutacji P.

Co by się jednak stało, gdybyśmy zakłócili tę regularność wprowadzając dodatkową operację? Gdyby zamiast przenosić wartość bezpośrednio na indeks – najpierw ją zmodyfikować? Zmieńmy nasz przykład i obliczmy zerowy element permutacji $Q3'$ dla naszego przykładowego P=(1, 3, 0, 4, 2), ale stosując taką właśnie dodatkową operację na przykład dodanie liczby 1. Pierwszy krok: $P[0]=1$. Drugi krok: wartość 1 przechodzi na indeks i sprawdzamy, jaką wartość ma $P[1]$ – jest to 3 ($P[1]=3$). Dla prostego potrójnego złożenia odczytalibyśmy teraz $P[3]=4$. Jednak zamiast tego dodajmy do liczby 3 jedynkę: 3+1=4 i odczytajmy wartość $P[4]$, gdzie $P[4]=2$. Uzyskaliśmy w ten sposób $Q3'[0]=P[P[P[0]]+1]=2$.

Pozornie nic istotnego nie zmieniliśmy, dołożyliśmy łatwą do odwrócenia operację dodania stałej (+1), więc intuicja mogłaby podpowiadać, że niczego szczególnego po operacji tej spodziewać się nie należy i takie *zmodyfikowane* złożenie permutacji będzie tak samo łatwe do odwrócenia, jak proste złożenie.

Tak kryptologowie mogli sądzić jednak do czasu odkrycia funkcji VMPC. Powyżej ($Q3'$) obliczyliśmy bowiem nic innego, jak zerowy element właśnie funkcji VMPC permutacji P. W tym momencie jesteśmy już w stanie obliczyć wartość funkcji VMPC dla całej permutacji P=(1, 3, 0, 4, 2): Q=VMPC(P): $Q[x]=P[P[P[x]]+1]=(2, 1, 0, 4, 3)$. Oczywiście operacja +1 jest dodawaniem modulo rozmiar permutacji (tutaj modulo 5), a więc P[P[1]]+1 jest równe nie 4+1=5, ale 0, co przy P[0]=1 daje wynik P[P[P[1]]+1]=1.

Na tej niepozornej dodatkowej operacji arytmetycznej, jaką jest dodanie jedynki do wartości elementu permutacji P podczas jej składania, opiera się potężna siła kryptograficzna funkcji VMPC. Okazuje się bowiem, że nie możemy – tak, jak było to możliwe w przypadku prostego złożenia – rozpisać struktury cykli elementów P na podstawie Q, a potem uzyskane pary czy trójki elementów P *poskładać* i odtworzyć postać permutacji P.

Odwrócenia funkcji VMPC jest problemem całkowicie odmiennym od zadania odwrócenia prostego złożenia permutacji i do jego rozwiązania, a więc do odnalezienia P na podstawie Q, gdzie $Q[x]=P[P[P[x]]+1]$, potrzebny jest skomplikowany algorytm założeniowo-wnioskujący, który część elementów permutacji P musi po prostu zgadnąć i dopiero na ich podstawie wnioskować wartości pozostałych elementów P z postaci permutacji Q. Proces założeń i wnioskowania powtarza się do skutku – aż zakończy się albo powodzeniem i odkryciem wszystkich elementów P, co kończy zadanie, albo porażką, gdy okaże się, że po serii założeń i wnioskowań dochodzimy do sprzeczności (na przykład wywnioskowaliśmy, że $P[3]$ musi być jednocześnie równe 1 i 2). W przypadku porażki musimy cofnąć się, zmienić wartość któregoś z założonych elementów P i spróbować ponownie wnioskować – tak długo, aż ujawnimy wszystkie elementy P przy braku sprzeczności. Okazuje się, że zadanie to jest niezwykle złożone obliczeniowo i średnio dopiero po założeniu 34 elementów P (dla 256-elementowych permutacji) możliwe jest wywnioskowanie całej pozostałej części permutacji P. Zanim to nastąpi – algorytm musi przeszukać średnio połowę z tych 34 możliwych wartości założonych elementów P. Proces ten zajmuje średnio właśnie 2260 operacji.

Zarysowany tu algorytm jest najefektywniejszym znanym aktualnie sposobem na odwrócenie funkcji VMPC. Dokładny opis algorytmu można znaleźć w pracy konferencyjnej FSE'04 *VMPC One-Way Function and Stream Cipher*, która jest dostępna na stronie internetowej poświęconej VMPC pod adresem *www.vmpcfunction.com/vmpc.pdf*. Algorytm jest także opisany w formacie HTML w sekcji *VMPC Function* strony głównej VMPC – *www.vmpcfunction.com*.

Prace Bartosza Żółtaka

Na załączonym do magazynu CD znajduje się praca konferencyjna *VMPC One-Way Function and Stream Cipher*, zaprezentowana na konferencji FSE'04 w Indiach, a także dwie nowe prace autora – *Tail-MAC: An Efficient Message Authentication Scheme for Stream Ciphers*, opisująca algorytm obliczania kodów uwierzytelniających wiadomość dla szyfru VMPC (tzw. MAC – *Message Authentication Code*), które pozwalają zweryfikować, czy wiadomość została poprawnie zdeszyfrowana oraz praca *One-Way IND-CNA Key Setup – a Step Towards Provably Secure Symmetric Encryption*, opisująca szereg założeń, po spełnieniu których możliwe jest skonstruowanie dowodu bezpieczeństwa szyfru VMPC. Prace dostępne są także na stronie internetowej VMPC.

Listing 1. *KSA VMPC*

```
Zmienne:
     c : ustalona długość klucza w bajtach, 16   c   64
     K : c-elementowa tablica zawierająca klucz
     z : ustalona długość wektora inicjującego w
                    bajtach, 16   z   64
     V : z-elementowa tablica zawierająca wektor
                    inicjujący
     s : 8-bitowa zmienna
Inicjowanie permutacji P i zmiennej s:
+  oznacza dodawanie modulo 256
    1. s = 0
    2. Dla n od 0 do 255: P[n]=n
    3. Dla m od 0 do 767: wykonaj kroki 4-6:
       4. n = m modulo 256
       5. s = P[ s + P[n] + K[m modulo c] ]
       6. x    = P[n]
          P[n] = P[s]
          P[s] = x
    7. Dla m od 0 do 767: wykonaj kroki 8-10:
        8. n = m modulo 256
        9. s = P[ s + P[n] + V[m modulo z] ]
       10. x    = P[n]
           P[n] = P[s]
           P[s] = x
```

Prostotę funkcji można dodatkowo zilustrować jej implementacją w asemblerze – obliczenie jednego elementu permutacji Q=VMPC(P) można zrealizować wykonując trzy podstawowe instrukcje `MOV` procesora (zajmujące jeden cykl zegarowy na procesorze 486 i Pentium) – patrz ramka.

Zastosowanie funkcji VMPC w kryptografii

Obszarem, na którym prostota funkcji VMPC oraz jej siła kryptograficzna mogą być wykorzystane jest projektowanie algorytmów szyfrujących. Dysponujemy bowiem funkcją jednokierunkową o ogromnej mocy, ale o niezwykle małych wymaganiach – równoważnych wykonaniu trzech jednocyklowych instrukcji MOV na bajt.

Pozwoliło to zbudować algorytm szyfrowania o bardzo prostej konstrukcji i bardzo wysokiej wydajności, który moc kryptograficzną czerpie z funkcji VMPC. Algorytm ten, nazwany przez autora jako szyfr strumieniowy VMPC, został zaprezentowany na konferencji FSE'04 w Indiach jako praktyczne zastosowanie funkcji VMPC w kryptografii.

Konkurencja – szyfr RC4, jego wady i zalety

Najpopularniejszym szyfrem strumieniowym jest opracowany w roku 1987 przez legendarnego Rona Rivesta algorytm RC4. Znalazł on zastosowanie w wielu komercyjnych aplikacjach, takich jak Lotus Notes, Netscape Navigator, SSL, WEP, Microsoft XBOX, Microsoft Office, Adobe Acrobat czy Oracle SQL. Głównymi zaletami RC4 są prostota konstrukcji i szybkość działania. Mimo że RC4 pozostaje niezłamany, to nie jest już uważany za algorytm zapewniający bardzo wysoki poziom bezpieczeństwa. W ostatnich latach odkryto wiele wad statystycznych strumienia generowanego przez RC4 (ataki typu *distinguisher*, a więc sposoby na odróżnienie generowanego strumienia wartości od strumienia idealnie losowego). Nawet ostatnia konferencja FSE była owocna w tej dziedzinie – w swojej pracy Bart Preneel, ze swoim doktorantem, opublikowali kolejną wadę statystyczną RC4.

Wiele wad znaleziono także w procedurze inicjowania klucza wewnętrznego (tzw. *Key Scheduling Algorithm*, KSA) oraz procedurze stosowania wektora inicjującego algorytmu RC4, co było obszarem, na jakim Scott Fluhrer, Itsik Mantin i Adi Shamir znaleźli skuteczny atak na protokół WEP. Jaką rolę w szyfrowaniu odgrywa KSA oraz wektor inicjujący, zobaczymy podczas omawiania konstrukcji szyfru VMPC.

Implementacja funkcji VMPC w asemblerze

```
MOV AL, [P] + EAX (Umieść (EAX=AL)-ty element P w AL)
MOV AL, [P] + EAX (Umieść (EAX=AL)-ty element P w AL)
MOV AL, [P] + EAX+1 (Umieść ((EAX=AL)+1)-ty element P w AL)
```

Instrukcje te umieszczają AL-ty (EAX-ty) element permutacji Q, gdzie Q[AL]=P[P[P[AL]]+1], w tym samym rejestrze AL (i EAX). Zakładamy, że P[256]=P[0].

VMPC w praktyce

Załóżmy, że w `L`-elementowej tablicy `Wiadomość` znajdują się dane do zaszyfrowania i że mamy przygotowaną `L`-elementową pustą tablicę `Szyfrogram`. Załóżmy też, że w 16-bajtowej (`c`=16) tablicy `K` zapisany jest klucz oraz w 16-bajtowej (`z`=16) tablicy `V` zapisany jest wektor inicjujący. `V` możemy wygenerować używając na przykład funkcji `Random`, `K` zaś powinno mieć postać możliwie losową i trudną do odgadnięcia. Teraz wykonujemy procedurę KSA (Listing 1), która na wejściu otrzymuje `K`, `V`, `c` oraz `z`, a na wyjściu zwraca wartość zainicjowanej permutacji `P` oraz zmiennej `s`. Następnie wykonujemy algorytm szyfru strumieniowego VMPC (Listing 2), który na wejściu otrzymuje `P`, `s` i `Wiadomość`, a na wyjściu zwraca `Szyfrogram`. Wektor inicjujący `V` dopisujemy na końcu `Szyfrogramu` i przesyłamy [`Szyfrogram+V`] odbiorcy. Odbiorca, aby zdeszyfrować `Szyfrogram`, musi znać wartość klucza `K`, odczytać wartość wektora inicjującego `V` z końca `Szyfrogramu`, wykonać KSA (Listing 1), a następnie zapisać `Szyfrogram` w tablicy `Wiadomość`, wykonać algorytm szyfru VMPC (Listing 2), który w tablicy `Szyfrogram` (!) zwróci zdeszyfrowaną wiadomość. Krytycznie ważne jest, aby każda wiadomość szyfrowana tym samym kluczem `K` miała INNĄ wartość wektora inicjującego `V`!

Listing 2. *Szyfr strumieniowy VMPC*

```
L : Długość szyfrowanej wiadomości w bajtach
+ oznacza dodawanie modulo 256
1. n = 0
2. Dla m od 0 do L - 1: wykonaj kroki 3-6:
   3. s = P[ s + P[n] ]
   4. Szyfrogram[m] = Wiadomość[m] XOR P[P[P[s]]+1]
   5. x    = P[n]
      P[n] = P[s]
      P[s] = x
   6. n = n + 1
```

Szyfr strumieniowy VMPC

Szyfr strumieniowy VMPC generuje strumień pseudolosowych bajtów na podstawie 256-elementowej permutacji *P*. W celu przekształcenia klucza (na przykład 128-bitowego) w permutację *P* wykorzystuje się procedurę inicjowania klucza wewnętrznego, *Key Scheduling Algorithm* (KSA). Posiada ona, oprócz klucza, jeden ważny parametr – wektor inicjujący (*Initialization Vector*, IV). Jest to rodzaj dodatkowego klucza, który ma inną wartość dla każdej szyfrowanej wiadomości i może być przesłany w sposób jawny, wraz z szyfrogramem. Dzięki zastosowaniu wektora inicjującego możemy używać tego samego klucza do szyfrowania wielu wiadomości i nie martwić się tym, że do zaszyfrowania ich użyty zostanie ten sam strumień wartości. Tajny klucz zapewnia bowiem tajność wygenerowanej permutacji *P*, natomiast wektor inicjujący sprawia, że permutacja *P* będzie miała inną postać dla każdej szyfrowanej wiadomości. Procedura inicjowania klucza wewnętrznego (KSA) szyfru strumieniowego VMPC zdefiniowana jest na Listingu 1. Stworzenie i optymalizacja implementacji algorytmu w dowolnym języku na podstawie listingu nie powinny sprawić problemów. W sprawdzeniu poprawności kodu pomóc mogą przykładowe dane wyjściowe algorytmu zamieszczone na stronie *www.vmpcfunction.com/cipher.htm#3*.

Ważną właściwością, jaka powinna charakteryzować bezpieczną procedurę KSA, jest efekt dyfuzji. Oznacza on, że oczekiwalibyśmy, iż – dla tego samego klucza *K* oraz dla dwóch wektorów inicjujących *V1* i *V2*, *prawie* takich samych (różniących się jednym tylko bitem lub bajtem) – powstałe permutacje *P* są *całkiem* różne, a więc relacje między nimi są takie same, jakich spodziewalibyśmy się od relacji między dwoma losowymi permutacjami. KSA algorytmu RC4 nie zapewnia prawidłowego efektu dyfuzji, co było podstawą opisanych przez Grosula i Wallacha, a także przez wspomnianych już Fluhrera, Mantina i Shamira, ataków typu related-key (kluczami pokrewnymi) oraz, pośrednio, ataku na protokół WEP.

Zgodnie z przeprowadzonymi analizami procedurę KSA szyfru VMPC charakteryzuje prawidłowy efekt dyfuzji. Niweluje on tego typu ataki, gdyż z definicji znalezienie regularności między permutacjami wygenerowanymi z *podobnych* (pokrewnych) kluczy lub wektorów inicjujących jest niemożliwe. Przeprowadzone testy statystyczne na algorytmie KSA VMPC są dokładniej opisane w pracy konferencyjnej FSE'04, a także w dziale *Research* strony internetowej VMPC – *www.vmpcfunction.com/research.htm.*

Wiedząc już, w jaki sposób permutacja *P* jest inicjowana kluczem i wektorem inicjującym, możemy przyjrzeć się, jak zbudowany jest sam szyfr VMPC, wykorzystujący postać wygenerowanej przez KSA VMPC permutacji *P* oraz zmiennej *s* – patrz Listing 2.

Zauważmy, że szyfr VMPC jest algorytmem niezwykle prostym. Jednak mimo to przeprowadzone testy statystyczne na strumieniach wartości wygenerowanych przez szyfr, wliczając w to wszystkie testy, w których wykazywane zostały słabości algorytmu RC4, nie ujawniły żadnej słabości statystycznej w generowanym przez VMPC strumieniu. Innymi słowy strumień bajtów wygenerowany przez szyfr VMPC, zgodnie z dzisiejszym stanem wiedzy, jest nieodróżnialny od strumienia wartości, jaki zostałby wygenerowany przez hipotetyczny generator liczb idealnie losowych. Nie istnieją także żadne inne znane sposoby złamania czy znalezienia luki w bezpieczeństwie VMPC. Jeśli dodamy omówione wcześniej znaczne różnice w bezpieczeństwie procedury inicjowania klucza (KSA) VMPC i RC4, to szyfr strumieniowy VMPC możemy uznać – zgodnie z dotychczasowym stanem wiedzy – nie tylko za bezpieczny i bardzo efektywny algorytm szyfrowania danych, ale także za postęp w kierunku zapewnienia wyższego poziomu bezpieczeństwa w stosunku do popularnego i szeroko stosowanego szyfru RC4.

Prostota konstrukcji szyfru VMPC pociąga za sobą także bardzo dużą wydajność w implementacjach programowych – na procesorze Intel Pentium 4; 2,66 GHz przy implementacji VMPC w języku asemblera, szybkość szyfrowania danych wynosi około 210 megabajtów na sekundę.

Dokładniejszy opis testów i analiz przeprowadzonych dla szyfru VMPC można znaleźć w pracy konferencyjnej FSE'04 oraz w dziale *Research* na stronie *www.vmpcfunction.com*. ■

Status prawny VMPC

Szyfr strumieniowy VMPC nie jest chroniony patentem i zastosowanie go w dowolnego rodzaju aplikacjach wymagających szyfrowania danych nie jest obarczone koniecznością wnoszenia opłat licencyjnych. Jeśli wymagana jest gwarancja zgodności zastosowanej implementacji szyfru z oryginałem, możliwe jest skorzystanie z oryginalnej implementacji VMPC, której zasady udostępniania opisane są na stronie *www.vmpcfunction.com*, w dziale *Policy of Using VMPC*.

Aplikacja wykorzystująca VMPC

W październiku 2004 roku planowane jest wprowadzenie na rynek aplikacji umożliwiającej szyfrowanie danych na dyskach i w sieci lokalnej oraz wiadomości tekstowych (na przykład przesyłanych pocztą elektroniczną) z wykorzystaniem technologii VMPC. Bieżące informacje na temat aplikacji będą udostępniane na stronie *www.vmpcfunction.com*.