

Kryptoanaliza funkcji skrótu

Praca przedstawia kryptoanalizę funkcji skrótu. W pierwszej części przedstawiono definicję używanych pojęć, oraz dokonano syntezy wyników kryptoanalizy znanych funkcji skrótu. W drugiej części przedstawiono autorski atak na funkcje skrótu z rodziny Petra, pokazując algorytm generowania pseudokolizji w poszczególnych rundach, oraz podważając jednokierunkowość rund.

1. Wprowadzenie

W 1990 roku Rivest przedstawił funkcje skrótu MD4. Stała się ona podstawą do tworzenia innych dedykowanych funkcji takich jak MD5, SHA, RIVEST czy HAVAL. Po wielu latach znikomego zainteresowania kryptoanalizą funkcji skrótu, nadszedł czas wielu ciekawych prac w tej dziedzinie. Większość funkcji z rodziny MD/SHA została skompromitowana, głównie za sprawą Chińczyków pod przewodnictwem profesor Wang. Sukcesy w kryptoanalizie funkcji skrótu opartych na filozofii zaczerpniętej z funkcji MD4, stały się bodźcem do przeanalizowania przez mnie innych funkcji skrótu. Swoją kryptoanalizę rozpocząłem od dedykowanej funkcji skrótu – Petra.

2. Funkcje skrótu

Pod pojęciem funkcji skrótu h rozumie się łatwe do wyznaczenia przekształcenie odwzorowujące wiadomość m o dowolnej, skończonej długości, w ciąg bitów o określonej, stałej długości – n .

$$h : \{0,1\}^* \rightarrow \{0,1\}^n,$$

$$\text{gdzie } \{0,1\}^* = \bigcup_{i \in \mathbb{N}} \{0,1\}^i$$

Od kryptograficznych funkcji skrótu wymaga się spełnienia dodatkowych własności:

(*) *Preimage resistance (non-invertibility)*:

Dany jest skrót $h(m)$, wiadomość m jest nieznana.

Znalezienie wiadomości m jest obliczeniowo trudne.

(**) *2nd preimage resistance (weak collision resistance)*:

Dany jest skrót $h(m)$ i odpowiadająca mu wiadomość m .

Znalezienie wiadomości $m' \neq m$ takiej, że $h(m) = h(m')$ jest obliczeniowo trudne.

(***) *Collision resistance (strong collision resistance)*:

Obliczeniowo trudne jest znalezienie dowolnej pary różnych wiadomości m i m' takich, że $h(m) = h(m')$

W literaturze szeroko stosowane są również następujące oznaczenia funkcji skrótu:

- jednokierunkowa funkcja skrótu (*OWHF – One Way Hash Function*)
– spełnienie własności * i **.
- bezkolizyjna funkcja skrótu (*CRHF – Collision Resistance Hash Function*)
– spełnienie własności ***.

Funkcje skrótu mają szerokie zastosowanie praktyczne. Stosuje się je w schematach podpisu cyfrowego, do przechowywania haseł systemów operacyjnych czy też baz danych, do wykrywania zmian w wiadomościach jako kody integralności (*MDC - Modification Detection Code*), oraz kody uwierzytelniające (*MAC - Message Authentication Code*). Używa się ich na szeroką skalę w celu badania integralności programów, różnego rodzaju łań i uaktualnień, czy też sygnatur wirusów. Funkcje skrótu znalazły także szerokie zastosowanie w różnych protokołach, m.in. SSL, SSH, IPSec. Stosowane są również w generatorach ciągów pseudolosowych.

3. Metody kryptoanalizy funkcji skrótu

Wśród metod łamania funkcji skrótu należy wyróżnić klasę ataków niewykorzystujących słabości wewnętrznej struktury analizowanej funkcji skrótu (brutalny, urodzinowy, wieloblokowy), oraz ataki znajdujące słabości przekształcające zastosowanych wewnątrz funkcji (różnicowy, liniowy).

3.1. Atak brutalny

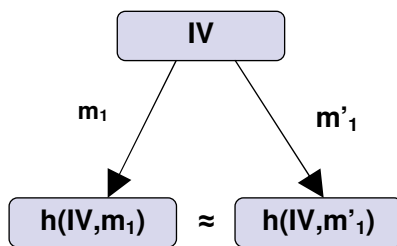
Celem ataku brutalnego jest znalezienie dowolnej wiadomości $m' \neq m$ która po skróceniu da zadany skrót $h(m') = h(m)$. Atak ten polega na przeszukiwaniu losowego zbioru wiadomości i porównywaniu z oczekiwaną wartością skrótu. Jest najwolniejszym z ataków, a jego złożoność zarówno obliczeniowa jak i pamięciowa wynoszą $O(2^n)$.

3.2. Atak urodzinowy

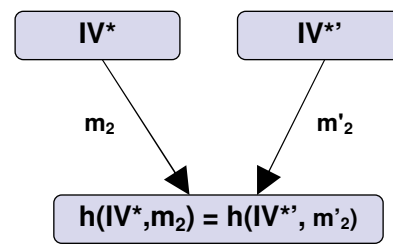
Celem ataku urodzinowego jest znalezienie dowolnej pary wiadomości m i m' , takich że $h(m) = h(m')$. Atak ten oparty jest na paradoksie dnia urodzin, głoszącym że prawdopodobieństwo tego, że spośród 23 osób, dwie mają urodziny w tym samym dniu wynosi więcej niż $1/2$. Atak ten polega na wygenerowaniu i zapamiętaniu $2^{n/2}$ wiadomości i tego rzędu jest złożoność ataku urodzinowego.

3.3. Atak wieloblokowy

Celem tego ataku jest znalezienie dowolnej pary złożonych wiadomości $m_1 || m_2 || \dots || m_k$ i $m'_1 || m'_2 || \dots || m'_k$, takich że $h(m_1 || m_2 || \dots || m_k) = h(m'_1 || m'_2 || \dots || m'_k)$. Atak ten stosowany jest przeciwko strukturze Merkle-Damgarda, którą wykorzystuje większość współczesnych funkcji. W ataku tym jako wyniki cząstkowe wykorzystuje się zarówno pseudokolizje (*pseudo-collision*) jak i prawiekolizje (*near-collision*).



Rys.1. Prawiekolizja



Rys.2. Pseudokolizja

Konkatenacja obu metod, a więc połączenie prawie i pseudokolizji może dawać w wyniku pełne kolizje dla wieloblokowych wiadomości:

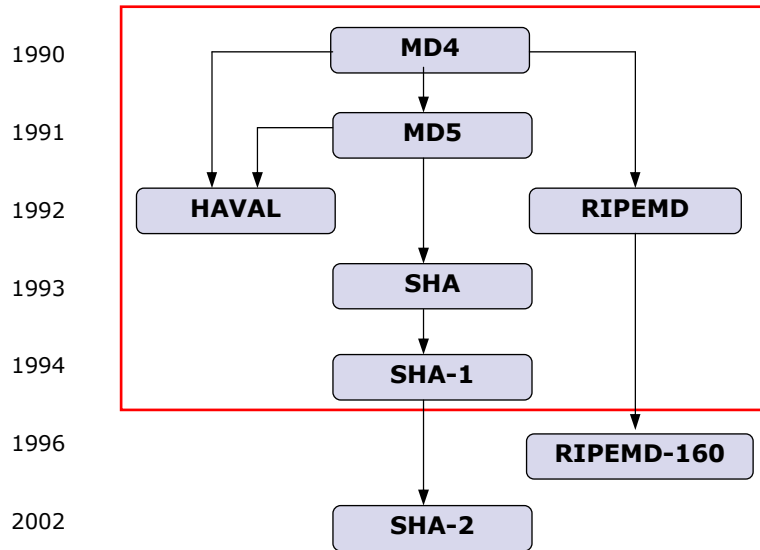
$$h(h(IV, m_1) || m_2) = h(h(IV, m'_1) || m'_2) \Rightarrow h(m_1 || m_2) = h(m'_1 || m'_2)$$

3.4. Atak różnicowy

Celem tego ataku jest znalezienie dwóch wiadomości dających ten sam skrót korzystając z niedoskonałości zastosowanej funkcji kompresyjnej. Różnica jest zazwyczaj definiowana jako funkcja logiczna *xor*, a filozofia ataku wykorzystuje fakt, iż poprzez zmianę kilku bitów w wiadomości prawdopodobne jest zniwelowanie różnicy wewnątrz funkcji kompresyjnej po kilku rundach

4. Aktualny stan kryptoanalizy rodziny MD/SHA

Rodzina funkcji skrótu wywodzących się od MD4 przedstawiona została poniżej. W ramce znajdują się funkcje skrótu które zostały skompromitowane.



Rys.3. Rodzina funkcji skrótu MD/SHA

4.1. MD

Funkcja skrótu MD4, która dała początek całej rodzinie MD/SHA została stworzona w roku 1990 przez Rona Rivest'a [16]. Rok 1992 przynosi atak na dwie ostatnie rundy algorytmu autorstwa pary: den Boer i Bosselaers [1]. Hans Dobbertin w 1997 roku najpierw wykazuje że dwie pierwsze rundy algorytmu nie są jednokierunkowe, a następnie przedstawia algorytm znajdowania kolizji z prawdopodobieństwem 2^{-22} [5]. Rok 2004 to kres funkcji MD4, Chńczycy pod przewodnictwem Wanga łamią przedstawiając wzór na generowanie kolizji [18]:

$$MD4(M) = MD4(M + \Delta M)$$

$$\Delta M = (0, 2^{31}, -2^{28} + 2^{31}, 0, 0, 0, 0, 0, 0, 0, 0, -2^{16}, 0, 0, 0)$$

Funkcja MD5 jest bezpośrednią następczynią MD4, autorstwa Rivesta, zaprezentowana w 1991 roku po sugestiach Dobbertina odnośnie słabości poprzedniczki. Jednak już w roku 1993 Boer i Bosselaers [3] znajdują pseudokolizje dla tej funkcji, a już trzy lata później Dobbertin [6] znajduje kolizje dla funkcji kompresującej algorytmu MD5. Rok 2004 to także obniżenie słabości tej funkcji. Wang przedstawia przepis na znajdowanie kolizji dla dwublokowych wiadomości. Znalezienie kolizji na komputerze klasy PC zajmuje około godziny.

W marcu 2006 roku Vlastimil Klima publikuje algorytm [12] znajdujący kolizje w czasie do 1 minuty wykorzystując metodę zwaną tunelowaniem.

4.2. HAVAL

Funkcja HAVAL została przedstawiona w roku 1992 przez trójkę Zheng, Pieprzyk i Seberry. W latach 2000-2003 zostaje przedstawionych kilka ataków na zredukowaną wersję algorytmu [9,11,13], natomiast Rompay [17] znajduje kolizję dla pełnej wersji algorytmu 3-Pass HAVAL przeprowadzając atak o złożoności 2^{29} . Wang w 2004 [18] przedstawił atak na 3-Pass HAVAL o złożoności 2^6 , a dwa lata później na 4-Pass HAVAL o złożoności 2^{36} . Natomiast inna grupa

Chńczyków [23] w 2006 roku przedstawia atak na 4-Pass HAVAL znajdujący kolizję w kilka godzin na komputerze klasy PC.

4.3. RIPEMD

Funkcja skrótu RIPEMD powstała w ramach projektu Unii Europejskiej o nazwie RIPE (RACE Integrity Primitives Evaluation) realizowanego w latach 1988-1992. Kilka lat później powstaje wzmocniona wersja algorytmu o nazwie RIPEM-160, której projektantami są Dobbertin, Bosselaers oraz Preneel [7]. Już w roku 1995 Dobbertin [8] udowadnia że dwie rundy funkcji kompresującej RIPEMD nie są odporne na kolizje. Następnie Wang łamie ją „na kartce papieru” przedstawiając wzór na znajdowanie kolizji [18]:

$$RIPEMD(M_1) = RIPEMD(M_1 + \Delta M)$$

$$\Delta M = (0,0,0,2^{20},0,0,0,0,0,0,2^{18} + 2^{31},0,0,0,0,2^{31})$$

Funkcja RIPEMD-160 do dnia dzisiejszego nie poddała się skutecznie kryptoanalizom.

4.4. SHA

Początek rodziny funkcji SHA (Secure Hash Algorithm) datuje się na 1993 rok. Wówczas NSA (National Security Agency) poprzez NIST (National Institute of Standards and Technology) publikuje pierwszą funkcję z tej rodziny nazywaną często SHA-0. Dwa lata później opublikowany zostaje algorytm SHA-1, który zastępuje swojego poprzednika ze względu na nieujawnione oficjalnie wady. W 2001 roku NIST publikuje ulepszoną wersję funkcji SHA dając jej roboczą nazwę SHA-2 i czekając na komentarze. W jej skład wchodzi trzy funkcje: SHA-256, SHA-384 i SHA-512. Rok później rodzina ta staje się standardem opublikowanym jako FIPS PUB 180-2

Historia kryptoanalizy funkcji SHA-0 przedstawia się następująco:

- Chabaud i Joux [4] w roku 1998 przedstawiają atak znajdujący kolizje złożoności 2^{61}
- Biham i Chen [1] w roku 2004 znajdują prawiekolizje (18 bitów różnicy), o raz pełną kolizję dla zredukowanej do 62 rund (z 80) wersji algorytmu
- 12 sierpnia 2004 Joux, Carribault, Lemuet i Jalby przedstawiają algorytm znajdujący kolizję o złożoności 2^{51}
- Trzy dni później – 17 sierpnia 2004 na konferencji Crypto 2004 Chinczycy (Wang) anonsują atak o złożoności 2^{40} , którego szczegółów jednak nie podali [18]
- X. Wang, Yin i Yu [20] w lutym 2005 przedstawiają atak o złożoności 2^{39}
- Pełna kolizja została przedstawiona wcześniej przez Joux [10]

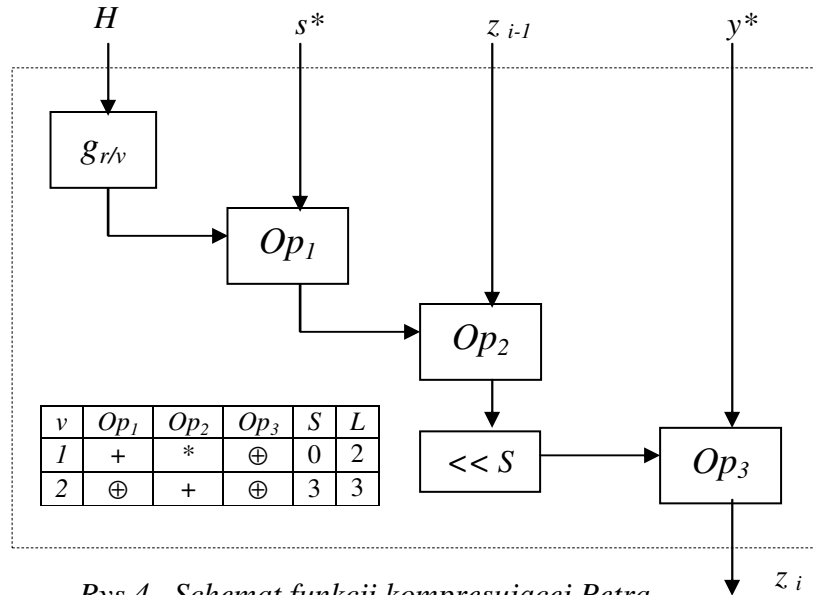
Następczyni algorytmu, czyli funkcja SHA-1 także okazała się podatna na ataki:

- W roku 2004 dwa niezależne ataki na zredukowaną do 53 rund (z 80) wersję algorytmu przeprowadzone przez pary: Biham, Chen oraz Rijmen, Oswald
- W lutym 2005 roku X. Wang, Yin i Yu przedstawiają atak o złożoności 2^{69} [21], po czym w sierpniu ulepszają go do złożoności 2^{63} [22].

Funkcje z rodziny SHA-2 na dzień dzisiejszy okazały się odporne na znane ataki.

5. Funkcje skrótu z rodziny Petra

Najjar w swojej rozprawie doktorskiej przedstawił całą rodzinę funkcji Petra składającą się z 4 funkcji. Każda funkcja należąca do rodziny jest oznaczana jako Petra- r/v , gdzie $r \in \{192, 256\}$ jest długością skrótu w bitach, natomiast $v \in \{1, 2\}$ oznacza jedną z dwóch wersji funkcji kompresyjnej. Ogólny schemat struktury bloku kompresyjnego został przedstawiony na schemacie (Rys.4)



Rys.4. Schemat funkcji kompresyjnej Petra

Wszystkie funkcje z rodziny Petra bazują na tych samych zmiennych inicjujących:

IV[0]=\$DAA4361D IV[1]=\$B16EC431 IV[2]=\$615BE562 IV[3]=\$98256F2E
 IV[4]=\$E565B322 IV[5]=\$98D8EB18 IV[6]=\$53A0EF98 IV[7]=\$6EA9D519

oraz tych samych stałych y^* :

y[0]=\$4CC59886 y[1]=\$33166219 y[2]=\$F31941D9 y[3]=\$E9053E34
 y[4]=\$155DD64D y[5]=\$9809045A y[6]=\$44E8ECDD y[7]=\$B310C998
 y[8]=\$CA0ECF98 y[9]=\$29F1A748 y[10]=\$EEB268AA y[11]=\$4822D4C0
 y[12]=\$4766EA27 y[13]=\$864CC598 y[14]=\$767CC650 y[15]=\$8D3A414F

Dodatkowo wykorzystywane są następujące funkcje:

$$g_{1/192} = H[0] \wedge H[2] \oplus H[0] \wedge H[5] \oplus H[1] \wedge H[3] \oplus H[1] \wedge H[5] \oplus H[4] \wedge H[5]$$

$$g_{2/192} = H[0] \wedge H[1] \oplus H[0] \wedge H[4] \oplus H[2] \wedge H[3] \oplus H[4] \wedge H[5]$$

$$g_{3/192} = H[0] \wedge H[3] \oplus H[0] \wedge H[5] \oplus H[1] \wedge H[4] \oplus H[2] \wedge H[4] \oplus H[2] \wedge H[5] \oplus H[3] \wedge H[4]$$

$$g_{1/256} = H[0] \wedge H[1] \oplus H[0] \wedge H[2] \oplus H[0] \wedge H[3] \oplus H[0] \wedge H[5] \oplus H[0] \wedge H[6] \oplus H[0] \wedge H[7] \oplus H[1] \wedge H[2] \oplus H[1] \wedge H[3] \oplus H[1] \wedge H[4] \oplus H[1] \wedge H[5] \oplus H[1] \wedge H[6] \oplus H[2] \wedge H[5] \oplus H[3] \wedge H[7] \oplus H[4] \wedge H[5] \oplus H[4] \wedge H[7] \oplus H[5] \wedge H[6] \oplus H[5] \wedge H[7] \oplus H[6] \wedge H[7]$$

$$g_{2/256} = H[0] \wedge H[1] \oplus H[0] \wedge H[3] \oplus H[0] \wedge H[4] \oplus H[0] \wedge H[5] \oplus H[0] \wedge H[6] \oplus H[1] \wedge H[2] \oplus H[1] \wedge H[4] \oplus H[1] \wedge H[5] \oplus H[1] \wedge H[6] \oplus H[2] \wedge H[3] \oplus H[2] \wedge H[6] \oplus H[3] \wedge H[4] \oplus H[3] \wedge H[7] \oplus H[4] \wedge H[7]$$

$$g_{3/256} = H[0] \wedge H[7] \oplus H[1] \wedge H[2] \oplus H[1] \wedge H[3] \oplus H[1] \wedge H[7] \oplus H[2] \wedge H[4] \oplus H[3] \wedge H[5] \oplus H[4] \wedge H[5] \oplus H[4] \wedge H[6] \oplus H[6] \wedge H[7]$$

Przed skróceniem wiadomości m jest uzupełniania do długości odpowiadającej wielokrotności 512 bitów, klasycznym algorytmem polegającym najpierw na dodaniu do wiadomości jednego bitu o wartości 1, następnie taką liczbą bitów 0, tak aby ostatnie 64 bity zawierały w sobie długość oryginalnej wiadomości. Następnie przeprowadzana jest operacja permutacji danych wejściowych oraz zmiennej y^* . Jednak ten etap w niniejszej pracy pomijam, gdyż dalsza analiza będzie bazować na pierwotnym uporządkowaniu wiadomości m oraz zmiennej y^* . Jest to pewne uproszczenie algorytmu, ale nie wpływa na złożoność znajdowania pseudokolizji i nieznacznie wpływa na odwracalność rundy.

Algorytm funkcji Petra-r/v (bez wstępnej permutacji)

WEJŚCIE: $y^*, m, g_{k/r}, IV$
 WYJŚCIE: H

METODA:

```

H = IV;
II= IV[0];
for i=1 to L do begin // liczba rund L ∈ {2,3}
  m = s*[0] || s*[1] || ... || s*[15];
  for j=1 to 16 // 16 kroków w rundzie
    temp = gi/r(H);
    temp = temp Op1 s*[j]; // Op1 ∈ {+, ⊕}
    temp = temp Op2 II; // Op2 ∈ {*, +}
    temp = temp SHL S; // S ∈ {0,3}
    temp = temp Op3 y*[j]; // Op3 ∈ {⊕}
    II = temp;

    if (r = 192) and (v = 1) then H[0]=H[5]; H[5]=H[2]; H[2]=H[3]; H[3]=H[4]; H[4]=H[1]; H[1]=II;
    if (r = 192) and (v = 2) then H[0]=H[5]; H[5]=H[2]; H[2]=H[3]; H[3]=H[1]; H[1]=H[4]; H[4]=II;
    if (r = 256) and (v = 1) then
      H[0]=H[7]; H[7]=H[3]; H[3]=H[1]; H[1]=H[5]; H[5]=H[2]; H[2]=H[6]; H[6]=H[4]; H[4]=II;
    if (r = 256) and (v = 2) then
      H[0]=H[1]; H[1]=H[4]; H[4]=H[2]; H[2]=H[6]; H[6]=H[7]; H[7]=H[3]; H[3]=H[5]; H[5]=II

  end
H = H ⊕ IV;
IV = H;
end
    
```

Legenda:

⊕ - xor logiczny
 ∧ - and logiczny
 + - dodawanie mod 2^{32}
 * - mnożenie mod 2^{32}
 S - wielkość przesunięcia bitowego
 L - liczba rund

6. Kryptoanaliza funkcji skrótu z rodziny Petra

Ponieważ autor funkcji Petra zaproponował całą ich rodzinę funkcji, jednak na potrzeby niniejszej pracy i większej przejrzystości skupię się na jednej z nich a mianowicie na funkcji Petra-192/2.

6.1. Jednokierunkowość rundy?

Prześledzę trzecią rundę funkcji kompresyjnej, zaznaczając że poniższa analiza może być przeprowadzona dla każdej rundy. Rozważania poniższe będą przeprowadzone dla uproszczonej wersji funkcji Petra, bez wstępnej permutacji zmiennej y , oraz bez końcowej operacji xor ($H=H \oplus IV$) występującej po każdej rundzie. Załóżmy że znamy wyjście funkcji kompresyjnej w postaci zmiennych $H_{16}[0], H_{16}[1], H_{16}[2], H_{16}[3], H_{16}[4], H_{16}[5]$. Indeksami będą oznaczał numer kroku w rundzie.

Z budowy rundy trzeciej algorytmu Petra-192/2 wynika że (*):

$$\forall_{i \in \{0,15\}} H_{i+1}[4] = shl_3 \left[(g_{3/192}(H_i[0], H_{i+1}[3], H_{i+1}[5], H_{i+1}[2], H_{i+1}[4], H_{i+1}[0]) \oplus s[i]) + H_{i+1}[1] \right] \oplus y[i]$$

$$H_0 = IV$$

z powyższego równania dla $i=15$ nie znamy $H_{15}[0]$ oraz $s[15]$, jednak analizując funkcję $g_{3/192}$ widzimy że dla pewnych wartości zmiennych $H[1], H[2], H[3], H[4], H[5]$ wartość funkcji $g_{3/192}$ jest niezależna od zmiennej $H[0]$. Kombinacje tych zmiennych przedstawia poniższa tabela.

H[0]	H[1]	H[2]	H[3]	H[4]	H[5]	$g_{3/192}$
x	0	0	0	0	0	0
x	0	0	0	1	0	0
x	0	1	0	0	0	0
x	0	1	0	1	0	1
x	1	0	0	0	0	0

x	1	0	0	1	0	1
x	1	1	0	0	0	0
x	1	1	0	1	0	0
x	0	0	1	0	1	0
x	0	0	1	1	1	1
x	0	1	1	0	1	1
x	0	1	1	1	1	1
x	1	0	1	0	1	0
x	1	0	1	1	1	0
x	1	1	1	0	1	1
x	1	1	1	1	1	0

Zależność ta występuje dokładnie dla połowy kombinacji zmiennych, z czego wynika iż przy znajomości wartości H_{16} uda się średnio odtworzyć połowę bitów zmiennej $s[15]$ z prawdopodobieństwem równym 1, przy założeniu braku występniej permutacji zmiennej y .

Celem bliższego przedstawienia idei posłużyć się spreparowanym przykładem. Niech wynikiem trzeciej rundy dla algorytmu Petra-192/2 będzie:

$$H_{16}[0] = \$DA5E522A \quad H_{16}[1] = \$5D77A7F7 \quad H_{16}[2] = \$DA5E522A$$

$$H_{16}[3] = \$18C8176A \quad H_{16}[4] = \$998D888A \quad H_{16}[5] = \$359D5C21$$

Podstawiając powyższe dane do równania (*)

$$H_{16}[4] = shl_3[(g_{3/192}(H_{15}[0], H_{16}[3], H_{16}[5], H_{16}[2], H_{16}[4], H_{16}[0]) \oplus s[15]) + H_{16}[1]] \oplus y[15]$$

widzimy iż wynik funkcji $g_{3/192}$ nie zależy od zmiennej $H_{15}[0]$ i otrzymujemy kolejno

$$\begin{aligned} \$998D888A &= shl_3 ((\$451F5141 \oplus s[15]) + \$5D77A7F7) \oplus \$8D3A414F \\ \$14B7C9C5 &= shl_3 ((\$451F5141 \oplus s[15]) + \$5D77A7F7) \\ \$A296F938 &= (\$451F5141 \oplus s[15]) + \$5D77A7F7 \\ \$451F5141 &= \$451F5141 \oplus s[15] \\ \$00000000 &= s[15] \end{aligned}$$

A więc został odtworzony ostatni wyraz skracanej wiadomości z prawdopodobieństwem równym 1. Powyższą czynność możemy kontynuować dalej, jednak w kolejnej iteracji zmiennymi niewiadomymi będą już $H_{16}[0]$, $H_{16}[1]$ oraz $s[14]$, a więc prawdopodobieństwo odtworzenia zmiennej $s[14]$ będzie znacznie mniejsze.

6.2. Pseudokolizje

Kryptoanaliza struktury wewnętrznej funkcji kompresującej używanej w rodzinie Petra pozwoliła dojść do algorytmu generowania pseudokolizji, czyli znajdowania takiej wiadomości m , która dla dwóch różnych wartości wektora inicjującego IV i IV' daje w wyniku ten sam skrót.

$$h(IV, m) = h(IV', m)$$

Podstawą do znajdowania pseudokolizji było zauważenie faktu, iż dla pewnych kombinacji danych wejściowych, wyście funkcji g jest niezależne od jednej z danych wejściowych. Przy odpowiednim wyborze, zmiana jednego bitu wektora inicjującego nie spowoduje zmiany wyniku działania funkcji g . Rozpatrując przykładowo funkcje $g_{1/192}$ widzimy jej niezależność od zmiennej $H[5]$ przy pozostałych bitach o wartości 0:

H[0]	H[1]	H[2]	H[3]	H[4]	H[5]	$g_{1/192}$
0	0	0	0	0	x	0

Wystarczy teraz odpowiednio zmodyfikować wektor inicjujący IV który jest postaci:

IV[0]	DAA4361D	11011010101001000011011000011101
IV[1]	B16EC431	10110001011011101100010000110001
IV[2]	615BE562	01100001010110111110010101100010
IV[3]	98256F2E	10011000001001010110111100101110
IV[4]	E565B322	11100101011001011011001100100010
IV[5]	98D8EB18	10011000110110001110101100011000

na następujący:

IV' [0] DAA4361D	11011010101001000011011000011101
IV' [1] B16EC431	10110001011011101100010000110001
IV' [2] 615BE562	01100001010110111110010101100010
IV' [3] 98256F2E	10011000001001010110111100101110
IV' [4] E565B322	11100101011001011011001100100010
IV' [5] 98D8EB98	10011000110110001110101110011000

i otrzymujemy identyczny wynik działania funkcji $g_{1/192}$. W następnym kroku algorytmu Petra zmienna $H[5]$ jest przepisywana w miejsce $H[0]$ i w drugim kroku także będzie różnica jednego bitu na wejściu funkcji $g_{1/192}$. Funkcja ta dla połowy liczby kombinacji zmiennych wejściowych jest niezależna od zmiennej $H[0]$, a więc z prawdopodobieństwem $\frac{1}{2}$ uzyskamy pseudokolizję pierwszej rundy. Różnice wewnątrz rundy zniwelują się już po dwóch krokach. Jeżeli rozważania są prowadzone dla uproszczonej wersji algorytmu (bez końcowej operacji xor po każdej rundzie) to automatycznie uzyskujemy pseudokolizję dla całej funkcji. Jeśli natomiast obliczenia zostaną przeprowadzone na pełnej wersji funkcji Petra (wraz z operacją xor) wówczas jesteśmy w stanie znaleźć prawiekolizję (wynik funkcji laszującej różni się jedynie tylko na jednym bicie – tym samym który zmieniliśmy w wektorze inicjującym) w czasie kilku sekund na komputerze klasy PC.

7. Wnioski

Ogromny wzrost zainteresowania kryptoanalizą funkcji skrótu w ostatnich latach, oraz wynikające z tego ataki (głównie Chńczykóów) stawiają pod znakiem zapytania bezpieczeństwo stosowania najpopularniejszych funkcji skrótu. Aktualnie najszerzej używane w zastosowaniach komercyjnych funkcje MD5 i SHA-1 zostały skompromitowane. NIST ogłosił, że do 2010 zaprzestanie stosować SHA-1 na rzecz funkcji SHA-2. Bezpieczeństwo wielu funkcji skrótu opartych na filozofii rodziny MD (w tym przykładowo funkcji Petra) stoi pod znakiem zapytania. Wydaje się iż nadszedł najwyższy czas na zaprojektowanie i przebadanie nowych dedykowanych funkcji skrótu. Jedną ze strategii może być filozofia zaczerpnięta z funkcji RIPEMD-160, gdzie wykonuje się przekształcenia w dwóch niezależnych gałęziach. Podejście takie zostało wykorzystane przez Koreńczyków w zaproponowanej przez nich funkcji FORK-256 (FSE 2006). Wykorzystali oni aż cztery gałęzie niezależnych obliczeń uzyskując przy tym funkcję wydajniejszą niż SHA-256. Czy podejście to okaże się skuteczne? Potrzeba czasu i zainteresowania wśród kryptoanalityków aby odpowiedzieć na to pytanie. Inne podejście to zaczerpnięcie z filozofii tworzenia szyfrów blokowych, a więc zastosowanie S-boxów, rotacje o zmienną liczbę bitów, szybki efekt lawinowy itp. Przykładami są tutaj funkcje Tiger czy Whirlpool. A może czas na zupełnie nowe podejście?

8. Literatura

1. E. Biham, R. Chen, *Near-collisions of SHA-0*, Cryptology ePrint Archive, <http://eprint.iacr.org/2004/146>, 2004
2. B. den Boer, A. Bosselaers, *An attack on the last two rounds of MD4*, Advances in Cryptology, Crypto'91, LNCS 576, Springer-Verlag, 1992
3. B. den Boer, A. Bosselaers, *Collisions for the Compression Function of MD5*, Advances in Cryptology, Proc. Eurocrypt'93, LNCS 756, Springer-Verlag, 1993
4. F. Chabaud, A. Joux, *Differential collisions in SHA-0*, Advances in Cryptology, Crypto'98, LNCS 1462, Springer-Verlag, 1998
5. H. Dobbertin, *Cryptanalysis of MD4*, Advances in Cryptology, FSE, LNCS 1039, Springer-Verlag, 1996
6. H. Dobbertin, *Cryptanalysis of MD5*, Rump Session Eurocrypt'06, 1996
7. H. Dobbertin, A. Bosselaers, B. Preneel, *"RIPMEMD-160: A Strengthened Version of RIPMMD"*, Advances in Cryptology, FSE'96, LNCS 1039, Springer-Verlag, 1996
8. H. Dobbertin, *RIPEMD with Two-round Compress Function is Not Collision-Free*, Journal of Cryptology 10(1), 1997
9. Y. S. Her, K. Sakurai, S. H. Kim, *Attacks for finding collision in reduced versions of 3-pass and 4-pass HAVAL*, International Conference on Computers, Communications and Systems, CE-15, 2003
10. A. Joux, *Collisions in SHA-0*, CRYPTO 2004 Rump Session, 2004
11. P. Kasselmann, W. Penzhorn, *Cryptanalysis of reduced version of HAVAL*, Vol. 36, No. 1, Electronic Letters, 2000
12. V. Klima, *Tunnels in Hash Functions: MD5 Collisions Within a Minute*, Cryptology ePrint Archive, <http://eprint.iacr.org/2006/105>, 2006
13. Park, S. H. Sung, S. Chee, J. Lim, *On the security of reduced versions of 3-pass HAVAL*, ACISP 2002, LNCS 2384, 2002
14. S. Landau, *Find Me a Hash*, Notices of the American Mathematical Society, 03.2006
15. M. Najjar, *Petra Family of Cryptographic Hash Functions*, Rozprawa doktorska, Politechnika Poznańska, Poznań, 2002
16. R. Rivest, *The MD4 message-digest algorithm*, Advances in Cryptology, Proc. Crypto'90, LNCS 597, Springer-Verlag, 1991
17. B. Van Rompay, A. Biryukov, B. Preneel, J. Vandewalle. *Cryptanalysis of 3-pass HAVAL*. *Advances in Cryptology*, Asiacrypto'03, LNCS 2894, Springer-Verlag, 2003
18. X. Wang, D. Feng, X. Lai, H. Yu, *Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD*, Cryptology ePrint Archive, <http://eprint.iacr.org/2004/199>, 2004
19. X. Wang, X. J. Lai, D. Feng, H. Chen, X. Yu, *Cryptanalysis of the Hash Function MD4 and RIPEMD*, Advances in Cryptology, Eurocrypt'05, LNCS 3494, Springer-Verlag, 2005
20. X. Wang, H. Yu, Y. Yin, *Efficient Collision Search Attacks on SHA-0*, Advances in Cryptology, Crypto'05, LNCS 3621, Springer-Verlag, 2005
21. X. Wang, A. L. Yin, H. Yu, *Finding Collisions in the Full SHA-1*, Advances in Cryptology, Crypto'05, LNCS 3621, Springer-Verlag, 2005
22. X. Wang, A. Yao, F. Yao, *New Collision search for SHA-1*, Rump Session Crypto'05, 2005
23. Z. Wang, H. Zhang, Z. Qin, Q. Meng, *Cryptanalysis of 4-Pass HAVAL*, Cryptology ePrint Archive, <http://eprint.iacr.org/2006/161>, 2006
24. Y. Zheng, J. Pieprzyk, J. Seberry, *HAVAL--A One-way Hashing Algorithm with Variable Length of Output*, Advances in Cryptology, Auscrypto'92, LNCS 718, Springer-Verlag, 1993