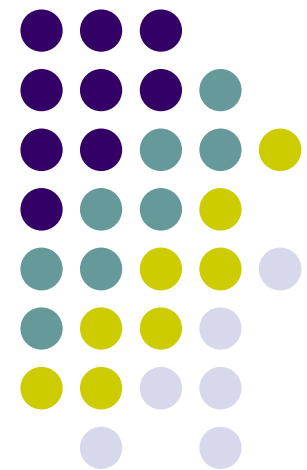


*Wojskowy Instytut Łączności*

# ***Kryptoanaliza funkcji skrótu z rodziny Petra***

*IV Warsztaty Kryptografii i Ochrony Informacji  
Katolicki Uniwersytet Lubelski  
19 - 20 maja 2006*



*Przemysław Rodwald*

# *Plan wystąpienia*



- Dlaczego funkcje skrótu?
- *Petra family* – co to za rodzina?
- Co złamano do tej pory?
- Co z tego wszystkiego wynika?



## *Kim jestem?*

- absolwent Wydziału Cybernetyki  
Wojskowej Akademii Technicznej
- pracownik Zakładu Kryptografii  
Wojskowego Instytutu Łączności



# *Co zrobiliśmy, nad czym pracujemy?*



- Grupowe urządzenie utajniające
- Inteligentny bank danych kryptograficznych
- Sprzętowy generator ciągów prawdziwie losowych
- Urządzenie utajniające ciągiem jednorazowym (szyfr Vernama)
- Kryptograficzna ochrona informacji w sieciach radiowych
- Cyfrowy terminal abonencki ISDN
- Szyfrator dla sieci IP
- Stacje generacji i dystrybucji kluczy





## ***Dlaczego funkcje skrótów?***

- Zaniedbana dziedzina kryptoanalizy
- Wzrost popularności za sprawą ostatnich ataków
- „Moda” na kryptoanalizę funkcji skrótów
- Potrzeba nowego podejścia

# Konstruowanie funkcji skrótu

- **Funkcje skrótu oparte na szyfrach blokowych**

- wykorzystują strukturę szyfrów blokowych
- gotowe rozwiązania, redukcja kosztów, bezpieczeństwo
- popularne w latach 80-tych
- zbyt wolne

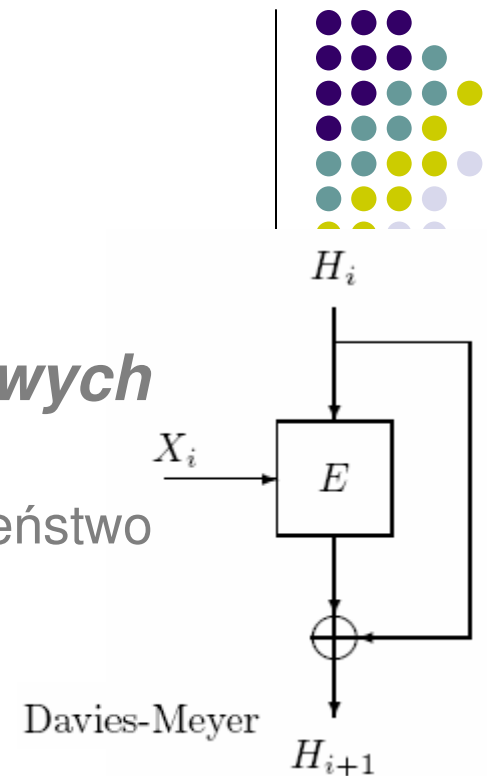
- **Funkcje oparte na arytmetyce modularnej**

- najczęściej wolne
- łatwo skalowalne

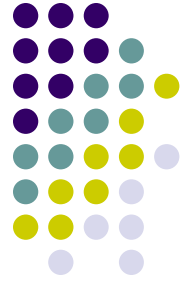
$$H_{i+1} = (((((H_i \oplus \tilde{X}_i) \vee A)^2 \bmod M) \neg c) \oplus H_i$$

- **Dedykowane funkcje skrótu**

- wykorzystują własne funkcje kompresujące
- obecnie najczęściej stosowane
- znacznie szybsze



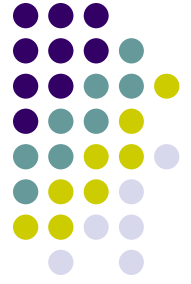
# Własności funkcji skrótu



- **Preimage resistance:**  
Dany jest skrót  $h(m)$ , wiadomość  $m$  jest nieznana.  
Znalezienie wiadomości  $m$  jest praktycznie niemożliwe.
- **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 praktycznie niemożliwe
- **Collision resistance (strong collision resistance):**  
Jest praktycznie niemożliwe znalezienie dowolnej pary  
różnych wiadomości  $m$  i  $m'$  takich że  $h(m) = h(m')$

OWHF  
 $2^n$

CRHF  
 $2^{n/2}$



## ***Klasy ataków na funkcje skrótów***

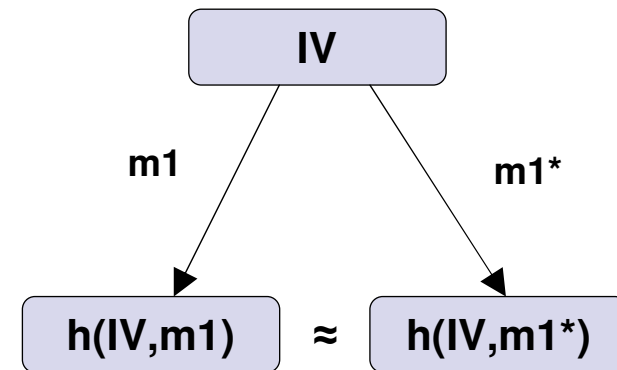
- **Atak brutalny  $O(2^n)$**
- **Atak urodzinowy  $O(2^{n/2})$**
  
- **Ataki strukturalne (structural attacks)**
  - struktura Merkle-Damgard
  - multi-block technique
  
- **Ataki różnicowe (differential attacks)**
  - funkcja kompresująca
  - differential attack on each block

# Ataki strukturalne

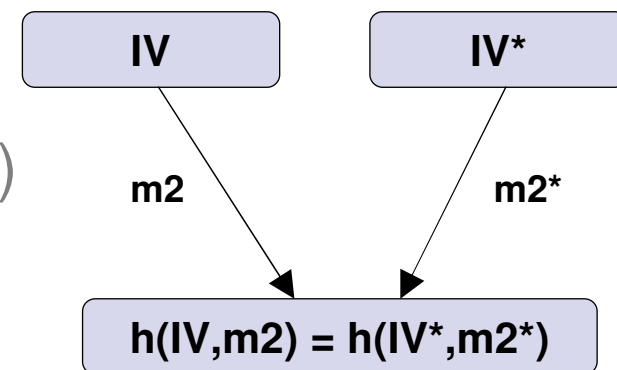
## multi-block technique



- prawie-kolizje (near-collisions)

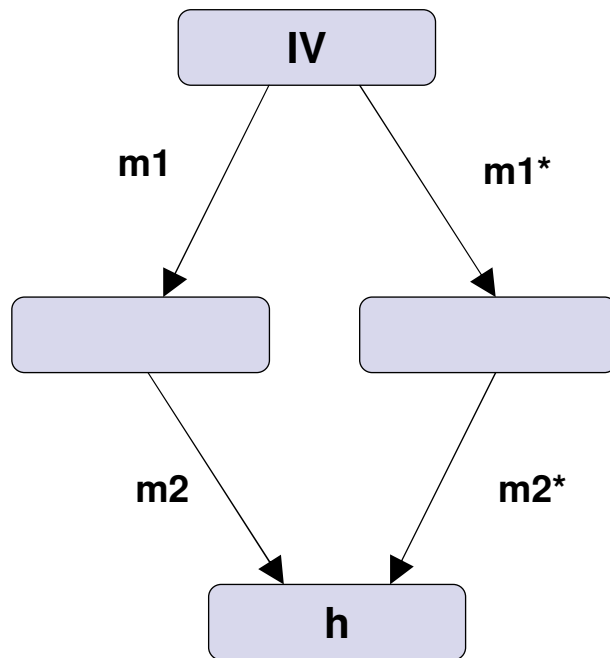
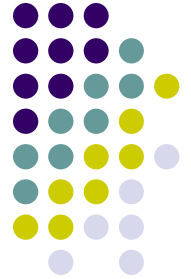


- pseudo-kolizje (pseudo-collisions)

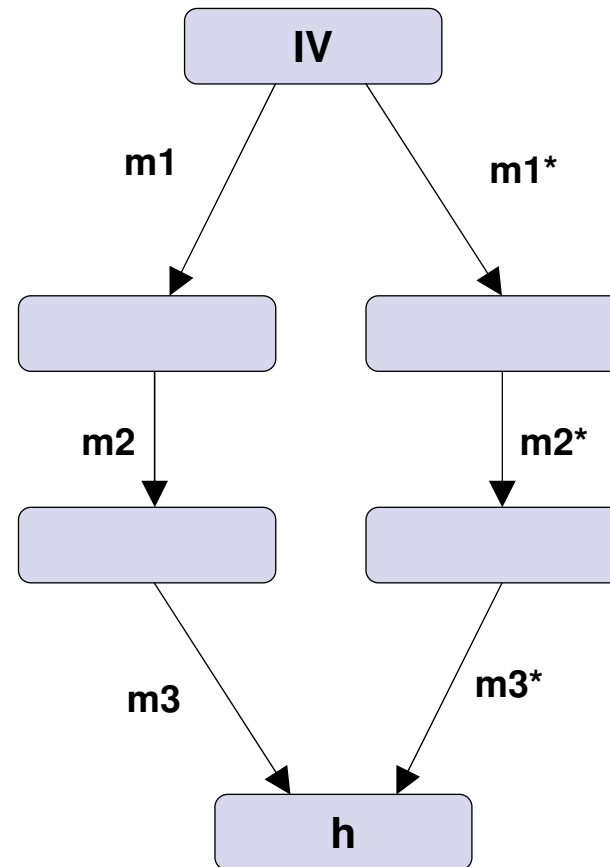


# Ataki strukturalne

## multi-block technique



$$h(m1||m2) = h(m1*||m2*)$$



$$h(m1||m2||m3) = h(m1*||m2*||m3*)$$

# ***Ataki różnicowe***

## ***differential attack on each block***



- współczesne ataki Bihama i Chena, Wanga są różnicowe
- poprzez zmianę kilku bitów w wiadomości prawdopodobne jest zniwelowanie „różnicy” po kilku rundach
- różnica jest zazwyczaj definiowana jako funkcja logiczna *xor*



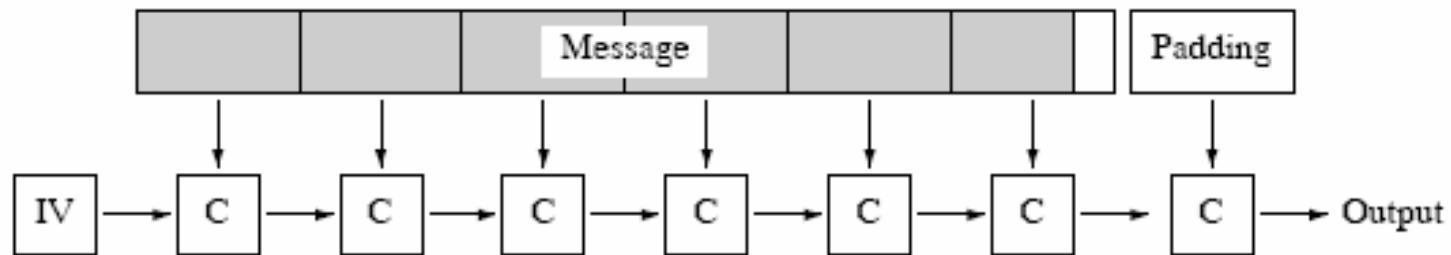
## ***Petra family – co to za rodzina?***

- Mohannada Najjar, „Petra Family of Cryptographic Hash Function”, 2002, Politechnika Poznańska
- rodzina 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 kompresującej.



## ***Petra family – struktura***

- blok długości 512 bitów
- ostatni blok jest uzupełniany ciągiem 100...0
- na ostatnich 64 bitach zapisywana jest długość oryginalnej wiadomości



# *Petra family*



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  $y^*$ :

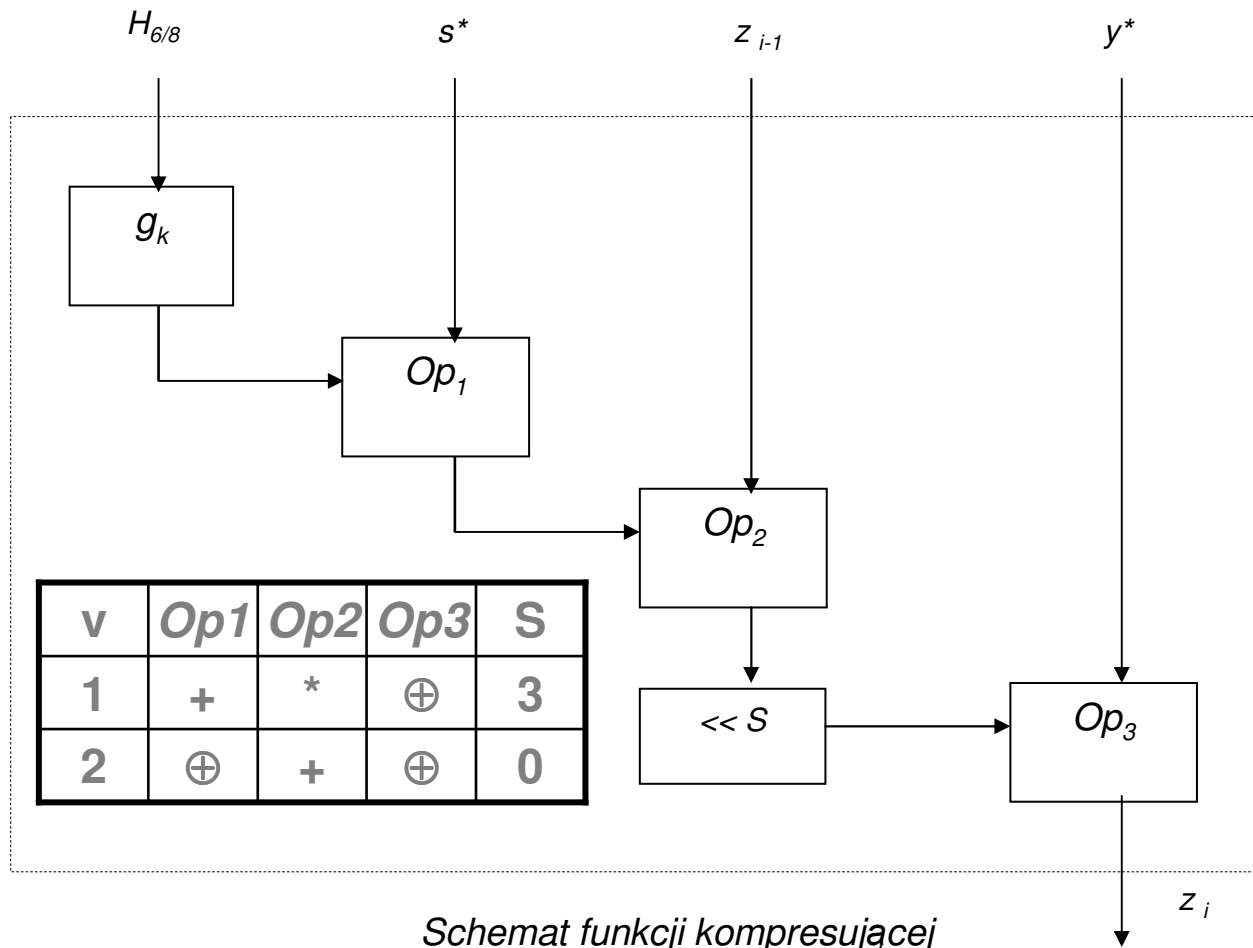
y[0] =\$4CC59886 y[1] =\$33166219 y[2] =\$F31941D9 y[3] =\$E9053E34  
y[4] =\$155DD64D y[5] =\$9809045A y[6] =\$44E8ECDD y[7] =\$B310C998  
y[8] =\$CA0ECF98 y[9] =\$29F1A748 y[10] =\$EEB268AA y[11] =\$4822D4C0  
y[12] =\$4766EA27 y[13] =\$864CC598 y[14] =\$767CC650 y[15] =\$8D3A414F

a także wykorzystują funkcje:

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



# Petra family – blok kompresujący





# Petra family – algorytm

WEJŚCIE:  $y^*, m, gk/r, IV$   
WYJŚCIE:  $H$

METODA:

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



## Petra – jednokierunkowość?

Założmy że znamy wyjście funkcji kompresującej w postaci zmiennych  $H_{16}[0], H_{16}[1], H_{16}[2], H_{16}[3], H_{16}[4], H_{16}[5]$ .

Z budowy rundy trzeciej algorytmu Petra-192/2 wynika iż

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

$$H_0 = IV$$

Z powyższego równania nie znamy  $H_i[0]$  oraz  $s[i]$ , jednak analizując funkcję  $g_{3/192}$  widzimy iż dla pewnych wartości zmiennych  $H[1], H[2], H[3], H[4], H[5]$  wartość funkcji  $g_{3/192}$  jest niezależna od zmiennej  $H[0]$ .



# ***Petra – jednokierunkowość?***

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

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



## ***Petra – przykład***

Niech wynikiem trzeciej rundy dla algorytmu Petra-192/2 będzie:

$$\begin{array}{ll} H_{16}[0] = \$DA5E522A & H_{16}[1] = \$5D77A7F7 \\ H_{16}[2] = \$DA5E522A & H_{16}[3] = \$18C8176A \\ H_{16}[4] = \$998D888A & H_{16}[5] = \$359D5C21 \end{array}$$

Podstawiając powyższe dane do równania

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

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

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



## ***Petra – funkcje $g$***

W rodzinie funkcji Petra w bloku kompresującym wykorzystywane są przekształcenia nieliniowe w postaci 6 funkcji drugiego rzędu ( $g$ ).

Jak łatwo policzyć stosunek wystąpień bitu 0 do 1 w powyższych funkcjach przedstawia się następująco:

<b>liczba</b>	<b><math>g_{1/192}</math></b>	<b><math>g_{2/192}</math></b>	<b><math>g_{3/192}</math></b>
<b>0</b>	<b>36</b>	<b>40</b>	<b>36</b>
<b>1</b>	<b>28</b>	<b>24</b>	<b>28</b>

*Wada: użycie funkcji niezbalansowanych (niezrównoważonych)*



# Petra – pseudo-kolizje

$$g_{1/192} = g_{2/192} = g_{3/192}$$

obecność  
bitów  
neutralnych

H[0]	H[1]	H[2]	H[3]	H[4]	H[5]
0	0	0	0	0	X
0	0	0	1	0	X
1	1	1	0	0	X
1	1	1	1	0	X
0	0	0	0	X	0
0	0	1	1	X	0
0	1	0	1	X	0
0	1	1	0	X	0
0	0	0	X	0	0
0	0	0	X	0	1
1	0	0	X	1	0
1	0	0	X	1	1
0	0	X	0	0	0
0	0	X	0	1	1
0	1	X	0	0	0
0	1	X	0	1	1
0	X	0	0	0	0
0	X	0	1	0	1
0	X	1	0	0	0
0	X	1	1	0	1
X	0	0	0	0	0
X	0	1	1	0	1
X	1	0	0	1	0
X	1	1	1	1	1



## *Petra – pseudo-kolizije*

IV[0] DAA4361D	110110101010010000110110 <b>0</b> 0011101
IV[1] B16EC431	101100010110111011000100 <b>0</b> 0110001
IV[2] 615BE562	011000010101101111100101 <b>0</b> 1100010
IV[3] 98256F2E	100110000010010101101111 <b>0</b> 0101110
IV[4] E565B322	111001010110010110110011 <b>0</b> 0100010
IV[5] 98D8EB <b>1</b> 8	100110001101100011101011 <b>0</b> 0011000

IV*[0] DAA4361D	110110101010010000110110 <b>0</b> 0011101
IV*[1] B16EC431	101100010110111011000100 <b>0</b> 0110001
IV*[2] 615BE562	011000010101101111100101 <b>0</b> 1100010
IV*[3] 98256F2E	100110000010010101101111 <b>0</b> 0101110
IV*[4] E565B322	111001010110010110110011 <b>0</b> 0100010
IV*[5] 98D8EB <b>9</b> 8	100110001101100011101011 <b>1</b> 0011000

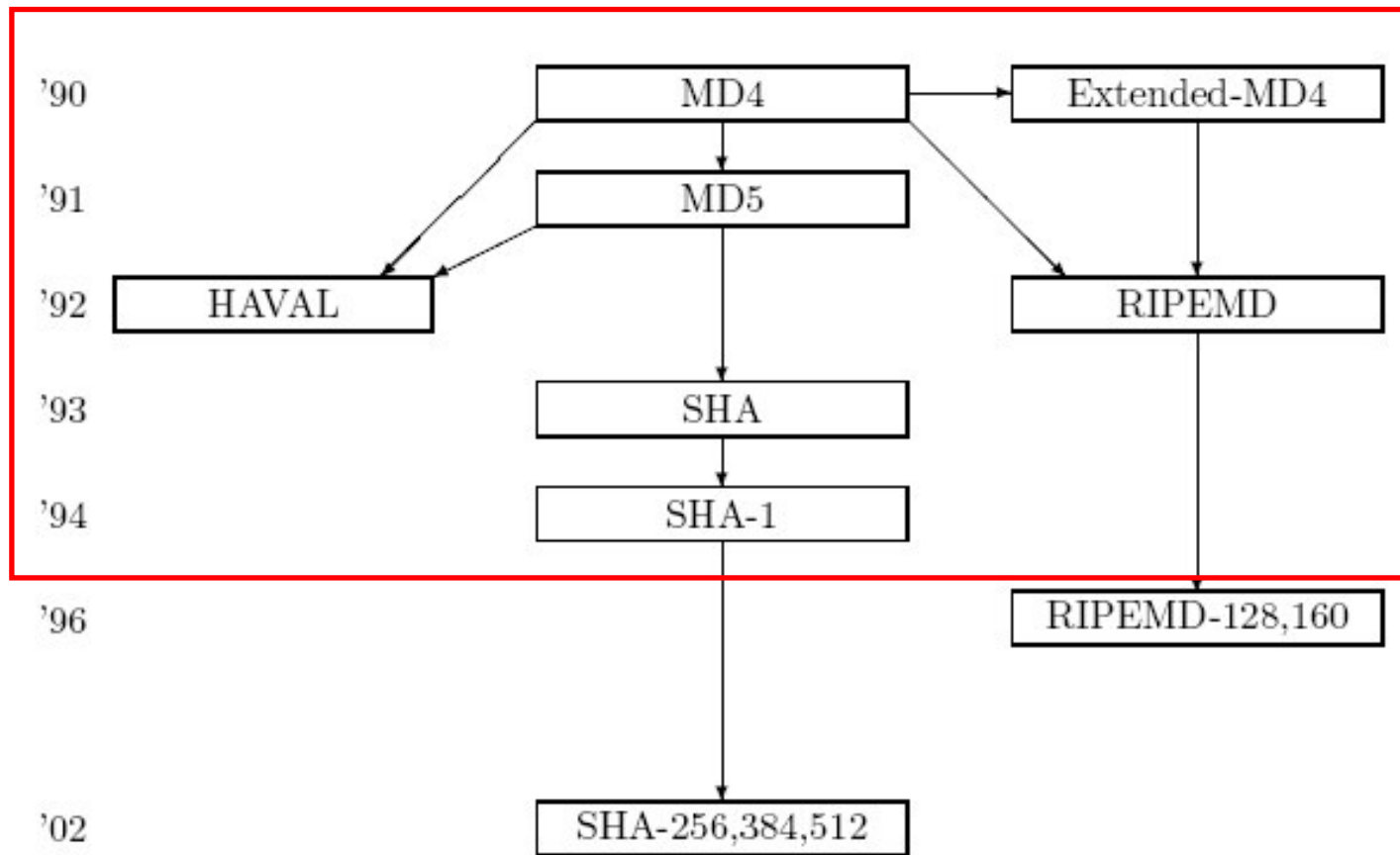
## *Co złamano do tej pory?*

- Słabości rodziny MD/SHA
- Potrzeba nowego podejścia
- Własne czy sprawdzone?





# Rodzina MD/SHA



# ***Słabości rodziny MD/SHA***



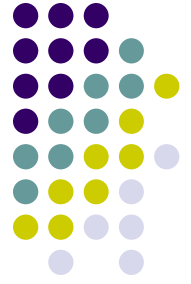
- Powolna dyfuzja
  - większość rejestrów jest tylko kopiowana w kolejnych rundach
  - wolny efekt lawinowy z powodu używanych funkcji (xor, add)
- Obecność neutralnych bitów
- Podatność na ataki różnicowe



## *Ataki różnicowe*

- Dwie rundy MD4 – kilka publikacji w latach 1990-1994
- MD4 – prawiekolizja, para wiadomości różniących się 3 bitami (Dobbertin, 1995)
- MD5 – kolizja ze zmienioną wartością wektora inicjującego IV (Dobbertin, 1996)
- SHA-0 – atak o złożoności  $2^{61}$  (1998)
  
- SHA-0 - reduced/extended versions, near-collisions (Biham, Chen, 2004)
- Multi-block, reduced versions of SHA-0 and SHA-1 (Biham, Chen, 2004)
- SHA-0 - a multi-block collision (Joux et al., 2004)
- SHA-1 - reduced versions (Rijmen, Oswald, 2004)
- MD5, MD4, and others — differential attacks (Wang et al., 2004)
- SHA-0 - a fast collision (Wang et al., 2005)
- SHA-1 – atak o złożoności  $2^{69}$  (Wang et al., 2005)

## ***Dlaczego tak późno?***



- ataki różnicowe były znane jako narzędzie do analizy szyfrów blokowych już w latach 90-tych
- *prawie* i *pseudo* kolizje znane były od wielu lat
- w latach 1997-2003 w zasadzie brak publikacji dotyczących kryptoanalizy funkcji skrótu,



## ***Potrzeba nowego podejścia***

- Funkcje skrótu muszą być projektowane z wykorzystaniem tej samej techniki jak szyfry blokowe, przy tych samych kryteriach projektowych.
- Kilka komercyjnych funkcji skrótu zostało zaprojektowanych w ten sposób (Tiger, Whirlpool)
- Rodzina funkcji MD4/SHA powinna być odrzucona.
- Bezpieczna długość  $> 160$  (2005)
- ? FORK-256 (FSE06)



## ***Własne czy sprawdzone?***

- Zasada Kerckhoffa, 1883, *La Cryptographie Militaire*  
„Zasada działania systemu (funkcji skrótu) musi być publicznie znana. Znajomość funkcji skrótu nie może wpływać na jej bezpieczeństwo”
- Podejście „wojskowe”

# Literatura



- ❑ Mohannada Najjar  
*Petra Family of Cryptographic Hash Function*
- ❑ A.J. Menezes, P. van Oorschot, S.A. Vanstone  
*Handbook of applied cryptography*
- ❑ S. Bakhtiari, R. Safavi-Naini, J. Pieprzyk  
*Cryptographic Hash Functions: A Survey*
- ❑ J. Pieprzyk T. Hardjono, J. Seberry  
*Fundamentals of Computer Security*
- ❑ Materiały konferencyjne  
*FSE06, Eurocrypt05,...*



# ***Dziękuję za uwagę***

## ***Pytania?***



## ***Kontakt***

[www.prodwald.prv.pl](http://www.prodwald.prv.pl)  
[prodwald@wp.pl](mailto:prodwald@wp.pl)

Wojskowy Instytut Łączności  
05-130 Zegrze  
[p.rodwald@wil.waw.pl](mailto:p.rodwald@wil.waw.pl)

