

# CRYPTANALYSIS OF PETRA'S FAMILY HASH FUNCTIONS

Przemysław Rodwald  
Military Communication Institute  
Cryptology Department  
05-130 Zegrze, Poland  
e-mail: [p.rodwald@wil.waw.pl](mailto:p.rodwald@wil.waw.pl)

## ABSTRACT

*Petra is a family of cryptographic hash functions proposed in 2002 by Mohannad Najjar in his PhD dissertation. In this paper I give a practical attack for finding pseudocollision in every round of Petra algorithm. In addition, I show that rounds of Petra are not one-way.*

## INTRODUCTION

Last two years brought a great progress in cryptanalysis of hash function. Especially breaking results of Wang et al. have changed the view of security of well-known iterated hash functions. Most hashes of MD/SHA family have succumbed to the Chinese attacks [7]-[11]. This huge progress in cryptanalysis of hash functions and weaknesses of most functions became a starting point for my cryptanalysis adventure. I decided to take into account others, less popular functions based on MD/SHA philosophy. My first hash function is Petra family proposed by Mohannad Najjar in Doctoral dissertation in 2002 [6].

The rest of the paper is organized as follows. In part 1 I proposed important definitions (hash function, classes of hashes). Methods of attacks on hash functions were described in part 2. In part 3 I introduced Petra family of hash functions, and finally I showed that compression block for Petra is not always one-way and I present an algorithm for finding pseudo-collision in compression block.

## 1. HASH FUNCTION

Hash function  $h$  is computationally efficient

function mapping an input  $m$  of arbitrary finite bitlength, to an output of fixed bitlength  $n$ .

$$h : \{0,1\}^* \rightarrow \{0,1\}^n,$$
$$\text{where } \{0,1\}^* = \bigcup_{i \in \mathbb{N}} \{0,1\}^i$$

For cryptographic hash function, the following properties are required [4]:

- A *preimage resistance (non-invertibility)* - it is computationally infeasible to find any input which hashes to that output,
- B *second-preimage resistance (weak collision resistance)* - it is computationally infeasible to find any second input which has the same output as any specified input,
- C *collision resistance (strong collision resistance)* - it is computationally infeasible to find any two distinct inputs  $m, m'$  which hash to the same output.

In literature we can find two classes of hashes:

- *OWHF – One Way Hash Function* – hash function with properties: A and B (above)
- *CRHF – Collision Resistance Hash Function* – hash function with properties: B and C

For an ideal hash function finding preimage or second-preimage require about  $2^n$  operations, and finding collision in birthday attack needs approximately  $2^{n/2}$  operations.

## 2. TYPES OF ATTACKS

Typical hash function has usually two components: a compression function and a method of iteration. Most of currently used hashes use Merkle[2] Damgård[5] iterative

structure. Taking into account design of hash functions we can distinguish three main types of attacks: general attacks, attacks against compression function and structural attacks. The first group depends only on the size of the hash, not on details of the algorithm. Second type analysis weaknesses of compressions block, for example differential attacks, slow diffusion or existence of neutral bits. Considering last group I focus my attention on Merkle-Damgård structure (see Figure 1).

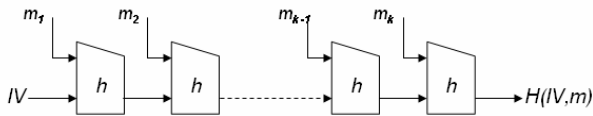


Figure 1. The Merkle-Damgård construction

Every hash function starts with the initial value ( $IV$ ). The message  $m$  is split into  $k$  blocks  $m_1, m_2, \dots, m_k$  of a fixed length. The hashed value  $H(IV, m) = H_k$  is computed according to the recursive equation:

$$H_i = h(H_{i-1}, m_i) \quad i = 1, 2, \dots, k \quad H_0 = IV$$

We can defined following attacks on iterated hash functions:

**Preimage attack:** given  $IV$  and  $H$ , find  $m$  such that  $h(IV, m) = H$ .

**Second preimage attack:** given  $IV$  and  $m$ , find  $m'$  such that  $m \neq m'$  and  $h(IV, m) = h(IV, m')$ .

**Free-start preimage attack:** given  $IV$  and  $H$ , find  $IV'$  and  $m'$  such that  $h(IV', m') = H$ .

**Free-start second preimage attack:** given  $IV$  and  $m$ , find  $IV'$  and  $m'$  such that  $m \neq m'$ ,  $IV \neq IV'$  and  $h(IV', m) = h(IV', m')$ .

Taking into account collisions we can distinguish following attacks:

**Collision attack:** given  $IV$ , find  $m$  and  $m'$  such that  $h(IV, m) = h(IV, m')$ .

**Semi-free-start collision attack:** find  $IV'$ ,  $m$  and  $m'$ , such that  $h(IV', m) = h(IV', m')$  and  $m \neq m'$ .

**Pseudo-collision attack:** find  $IV$ ,  $IV'$ ,  $m$  and  $m'$ , such that  $h(IV, m) = h(IV', m')$ ,  $IV \neq IV'$  and  $m \neq m'$  (see Figure 2).

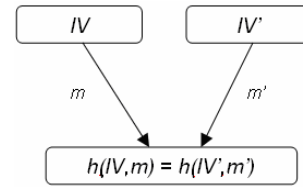


Figure 2. Pseudo-collision attack

Additionally one can define one more attack:

**Near-collision attack:** given  $IV$ , find  $m$  and  $m'$  such that  $h(IV, m) \approx h(IV, m')$  (see Figure 3).

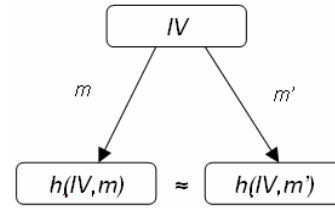


Figure 3. Near-collision attack

Finally I can propose multi-block collision attack. A multi-block collision attack technique finds two colliding messages each of the length of at least two blocks. It looks like concatenation of near-collision and pseudo-collision (see Figure 4).

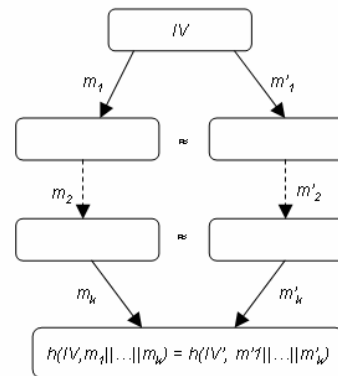


Figure 4. Multi-block technique

### 3. PETRA FAMILY

Petra family of hash functions is composed of four functions. Each function that belongs to the family is denoted as  $Petra-r/v$ , where  $r \in \{192, 256\}$  is a bitlength of hash result and  $v \in \{1, 2\}$  specifies one of two versions of the compression function. General structure of compression block is presented below (see Figure 5).

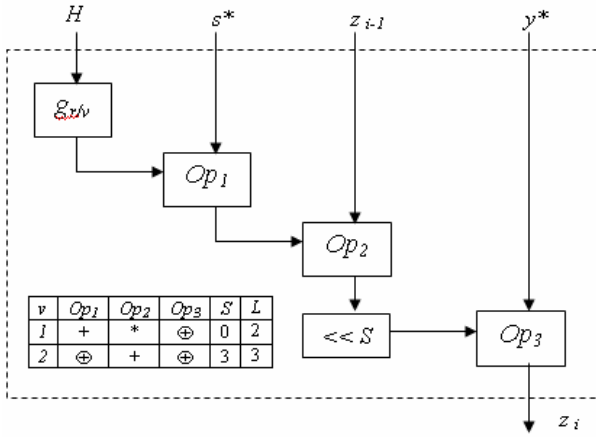


Figure 5. Compression block for Petra

All functions from Petra family use the same 32-bits initial value:

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

and 32-bits constant 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

Compression function for Petra-192 use perfect nonlinear 6-argument homogeneous functions:

$$g_{1/192}(x_1, x_2, x_3, x_4, x_5, x_6) = x_1x_3 \oplus x_1x_6 \oplus x_2x_4 \oplus x_2x_6 \oplus x_5x_6$$

$$g_{2/192}(x_1, x_2, x_3, x_4, x_5, x_6) = x_1x_2 \oplus x_1x_5 \oplus x_3x_4 \oplus x_5x_6$$

$$g_{3/192}(x_1, x_2, x_3, x_4, x_5, x_6) = x_1x_4 \oplus x_1x_6 \oplus x_2x_5 \oplus x_3x_5 \oplus x_3x_6 \oplus x_4x_5$$

and Petra-256 use 8-arguments functions:

$$g_{1/256}(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8) = x_1x_2 \oplus x_1x_3 \oplus x_1x_4 \oplus x_1x_6 \oplus x_1x_7 \oplus x_1x_8 \oplus x_2x_3 \oplus x_2x_4 \oplus x_2x_5 \oplus x_2x_6 \oplus x_2x_7 \oplus x_3x_6 \oplus x_4x_8 \oplus x_5x_6 \oplus x_5x_8 \oplus x_6x_7 \oplus x_6x_8 \oplus x_7x_8$$

$$g_{2/256}(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8) = x_1x_2 \oplus x_1x_4 \oplus x_1x_5 \oplus x_1x_6 \oplus x_1x_7 \oplus x_2x_3 \oplus x_2x_5 \oplus x_2x_6 \oplus x_2x_7 \oplus x_3x_4 \oplus x_3x_7 \oplus x_4x_5 \oplus x_4x_8 \oplus x_5x_8$$

$$g_{3/256}(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8) = x_1x_8 \oplus x_2x_3 \oplus x_2x_4 \oplus x_2x_8 \oplus x_3x_5 \oplus x_4x_6 \oplus x_5x_6 \oplus x_5x_7 \oplus x_7x_8$$

Before hashing each message is padded, but in the following I neglect this process because my attack has no relation with them. I focus only on the compression process. Additionally Petra family use own algorithm of order message words and constant y\*. I neglect this process too because it has no influence on finding pseudo-collisions.

A general algorithm of compression function of Petra-r/v is as follows:

```

H = IV;
II= IV[0];
for i=1 to L do begin //number of rounds L ∈ {2,3}
m = s*[0] || s*[1] || ... || s*[15];
for j=1 to 16 do begin // number of step
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 // number of step
H = H ⊕ IV;
IV = H;
end //number of rounds
  
```

In the following cryptanalysis I focus only on Petra-192/2, but similar analysis could be done for others members of the Petra family too.

### 3.1. ONE-WAYNESS?

I trace third round of compression block of simplified version Petra 192/2 (without permutation on input message). We know the output of compression block:  $H_{16}[0], H_{16}[1], H_{16}[2], H_{16}[3], H_{16}[4], H_{16}[5]$ .

And we know that (\*):

$$\forall_{i \in \{0,15\}} H_{i+1}[4] = shl_3(t) \oplus y[i]$$

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

$$H_0 = IV$$

We can notice that for some values of variables  $H[1], H[2], H[3], H[4], H[5]$ , value of the function  $g_{3/192}$  is independent of variable  $H[0]$ . Therefore we can find the value  $s[i]$ .

To better understand this idea I show an example. Let the output of the third round of compression block be as follows:

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

Let's substitute these variables to the equation (\*)

$$H_{16}[4] = shl_3(t) \oplus y[16]$$

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

For all following combinations function  $g_{3/192}$  is independent of the first input value.

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

We can observe that in our example function  $g_{3/192}$  is independent of the variable  $H_{15}[0]$ . We can calculate the value of last message word with probability equal to 1:

$$\$00000000 = s[15]$$

### 3.2. PSEUDO-COLLISIONS

We can observe that for some values of variables  $H[0], H[1], H[2], H[3], H[4], H[5]$ , value of the functions  $g_{1/192}$  is independent of one of this variables. Taking into account function  $g_{1/192}$  we see for example that value  $g_{1/192}$  is independent of variable  $H[5]$ , when other variables are equal to 0:

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

Now we can modify original initial vector IV:

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

on the following:

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

The output of the first step of compression block will be identical for every input message  $m$ . In the next step variable  $H[5]$  is copied in the place of  $H[0]$  and in the second step we will have the difference in one bit in the input of the function  $g_{1/192}$ . This function is independent of the variable  $H[0]$  for half combination of all input variables. So we can find pseudo-collision for the first round in probability  $1/2$ . Similar situation takes place in the next two rounds.

## 4. CONCLUSION

In this article I presented an attack on Petra family. I showed how to find a pseudo-collision in every round of the compression block and I showed that rounds are not always one-way. In my opinion finding real collision or pseudo-collision for Petra family is only a matter of time.

Taking into account last breaking results in hashes based on MD/SHA philosophy, we can say that this family is not fully trusted and should be discontinued. We need new standard. More secure and more resistant to collisions. Every hash function should be verified in the same way as block ciphers are.

Which way should we go to obtain good cryptographic hash functions? We can adopt all the design techniques that are used in block ciphers (S-boxes, fast diffusion etc.). But maybe some new design proposals should be invented? One of recently designed hash is FORK-256 [3], where compression function consists of 4 parallel branches. This means that FORK-256 can be implemented in parallel. Also it is difficult to analyze all branches simultaneously. Branch functions use nonlinear functions that are quite different from the Boolean function which have been used in existing hash functions, because they updates multiple words. Authors use different message-ordering in branches. This properties make it difficult to analyze FORK-256 with known attack methods including Wang's attack. Is it a good approach? Time will show.

## BIBLIOGRAPHY

- [1] S. Bakhtiari, R. Safavi-Naini, *J. Pieprzyk Cryptographic hash functions: a survey*. Technical Report 95-09, University of Wollongong, 1995
- [2] I. Damgård, *A design principle for hash functions*, Advances in Cryptology, CRYPTO'89, LNCS 435, Springer-Verlag, 1989
- [3] D. Hong, D. Chang, J. Sung, S. Lee, D. Moon, S. Chee, *A New Dedicated 256-bit Hash Function: FORK-256*, Proc FSE 2006, 2006
- [4] A.J. Menezes, P. van Oorschot, S.A. Vanstone, *Handbook of applied cryptography*, CRC Press, 1997
- [5] R. Merkle, *One way hash function and DES*, Advances in Cryptology, CRYPTO'89, LNCS 435, Springer-Verlag, 1989
- [6] M. Najjar, *Petra Family of Cryptographic Hash Functions*, Doctoral dissertation, Poznań University of Technology, Poznań, 2002
- [7] 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
- [8] 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

- [9] X.Wang, H.Yu, Y.Yin, *Efficient Collision Search Attacks on SHA-0*, Advances in Cryptology, Crypto'05, LNCS 3621, Springer-Verlag, 2005
- [10] X.Wang, A.L.Yin, H.Yu, *Finding Collisions in the Full SHA-1*, Advances in Cryptology, Crypto'05, LNCS 3621, Springer-Verlag, 2005
- [11] X.Wang, A.Yao, F.Yao, *New Collision search for SHA-1*, Rump Session Crypto'05, 2005

## BIOGRAPHY

Przemysław Rodwald was born in Lębork, Poland, in 1977. He was graduated from the Cybernetics Faculty of the Military University of Technology in Warsaw in 2001. Since 2002 he has been working at the Military Communication Institute. His main object of interest: cryptanalysis and hash functions.