Preimage Attack on Parallel FFT-Hashing

Donghoon Chang

Center for Information Security Technologies(CIST),
Korea University, Korea
dhchang@cist.korea.ac.kr

Abstract. Parallel FFT-Hashing was suggested by C. P. Schnorr and S. Vaudenay in 1993 [4]. That is a simple and light weight hash algorithm. Its basic component is a multi-permutation which helps to prove it's collision resistance. We show a preimage attack on Parallel FFT-Hashing with complexity $2^{t+64} + 2^{128-t}$ and memory 2^t which is less than the generic complexity 2^{128}. Our method can be described as "disseminative-meet-in-the-middle-attack"; we actually use the properties of multi-permutation to our advantage in the attack. Overall, this shows that the structure of Parallel FFT-Hashing has some weaknesses when the preimage attack is considered. To the best of our knowledge, this is the first attack on Parallel FFT-Hashing.

Keywords : Hash Function, Preimage Attack.

1 Introduction.

Nowadays, the novel construction of hash function is required because MD4-style hash functions were broken. In second NIST Hash Workshop in Aug. 2006, several new hash functions were sugested. Lyubashevsky *et al.* suggested 'Provably Secure FFT Hashing' which is a hash function family and one hash function can be randomly chosen from the family. Security of Provably Secure FFT Hashing is based on a solving certain lattice problems in the worst case. Bertoni *et al.* suggested a hash function 'RadioGatún' which is a belt-and-mill hash function. Their idea of design makes the size of inner state bigger than output size and uses only bitwise operations and rotations in order to implement easily and fastly in any flatform rather than using multiplication and addition operations. Gligoroski *et al.* suggested 'Edon-$\mathcal{R}$' which is an infinite family of hash functions based on the concept of shapeless quasigroup. Charles *et al.* suggested hash functions from expander graphs. Bentahar *et al.* suggested 'LASH'.

By the way, all above hash functions except LASH and hash function based on expander graph (we have not checked these two cases) require big memory sizes. For example, RadioGatún's word size is 64 bit by default. So its internal state is 64*58 bits. In case of Edon-$\mathcal{R}$, it uses shapeless quasigroup where the operation should be defined. As described, 256*256*8 bits are required when the order of the group is 2^8. In case of Provably Secure FFT Hashing, we need the

memory to store the key whose size is about 4600 bits in case of 513 bit hash output.

Even well-known hash functions such as Tiger and Whirlpool have big memory sizes because they use S-box.

This paper concern about Parallel FFT-Hashing suggested by C. P. Schnorr and S. Vaudenay in 1993 [4] improved from previous results [2, 3, 1, 6, 5]. Parallel FFT-Hashing uses a simple component 'multipermutation' repeatedly. Therefore, its algorithm can be implemented in low memory device environment such as RFID and the sensor network. But, this paper shows that we can find a premiage with complexity $2^{t+64}+2^{128-t}$ and memory 2^t less than exhaustive search complexity 2^{128}.

2 Parallel FFT-Hashing

In this section, we describe Parallel FFT-Hashing [4]. Here $+$ is the addition modulo 2^{16} on $E \cong \{0, 1, \cdots, 2^{16}-1\}$, $*$ is the multiplication in $E \cong \mathbb{Z}^*_{2^{16}+1}$. L is the one-bit circular left shift on $\{0,1\}^4$ (such that $L(i) = 2i$ for $i \leqslant 7$ and $L(i) = 1 + 2(i-8)$ for $i > 7$) and R is the one-bit circular right shift on E. And $c = 0^8 1^8$ and $s = 5$. In our attack, we can find a premiage for any s (even for big s). (c_0, c_1, c_2, c_3):=(oxef01, ox2345, ox6789, oxabcd), (c_4, c_5, c_6, c_7):=(oxdcba, ox9876, ox5432, ox10fe), c_{8+i}:=$\overline{c_i}$ for i=0,...,7 where c_i is the bitwise logical negation of c_i.

PaFFTHashing$(M) = o_1||o_2||\cdots||o_7$
M is the padded message for which $M = m_0||m_1||\cdots||m_{n-1} \in E^n$

1. **For** $i = 0, \ldots, 15$ **Do** $e_i := c_i$ ($c_0||\cdots||c_{15}$ is the initial value.)
2. **For** $j = 0, \ldots, \lceil n/3 \rceil$+s-2 **Do** (: Step j)
   - 2.1 **For** $i = 0, \ldots, 11$ **Do**
     - $e_{L(i)} := e_{L(i)} + m_{3j+(i \bmod 3)}$ for even i.
     - $e_{L(i)} := e_{L(i)} * m_{3j+(i \bmod 3)}$ for odd i.
   - 2.2 **For** $i = 0, \ldots, 7$ **Do** in parallel
     - $e_{2i} := e_{L(2i)} \oplus e_{L(2i+1)}$, $e_{2i+1} := e_{L(2i)} \oplus (e_{L(2i+1)} \wedge c) \oplus R^{2i+1}(e_{L(2i+1)})$
   - 2.3 **For** $i = 0, \ldots, 15$ **Do** $e_i := e_i * c_i$
3. **Output** $h_4(M) := o_1||o_2||\cdots||o_7$ for which $o_i = e_{L(2i)} * e_{L(2i+1)}$.

Fig. 1. Parallel FFT-hashing.

.

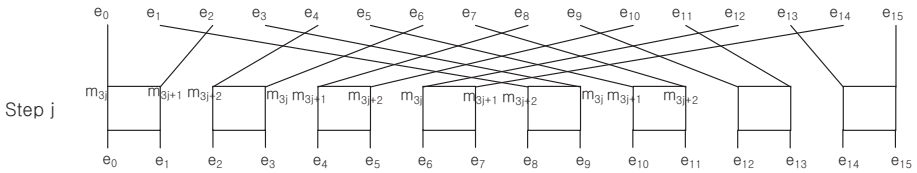

Fig. 2. Step j of Parallel FFT-Hashing.

3 Attack Strategy

4 Preimage Attack on Parallel FFT-Hashing

In this section, we show how to get a preimage for a given hash output $o_0||o_1||\cdots||o_7$. The padded preimage is $m_0||m_1||\cdots||m_{47}$ for which each m_i is 16 bits. Last four words $w_{44} \sim w_{47}$ indicate the message length. We let m_{43} '10^{15}'. Therefore, $m_0 \sim m_{42}$ is a real message. Our attack idea is a disseminative-meet-in-the-middle attack in location of output of Step 7. See Fig. 3 and Fig. 4.

First (part of obtaining partially fixed values with complexity 1): (0) $\sim$ (3) are already fixed values because they are initial values. We give (4) $\sim$ (19) to fixed values. Then the values of (20) $\sim$ (35) are determined. And then we give $w_0 \sim w_3$ to fixed values. So $w_0 \sim w_7$ are determined. Then we give m_{3i+2} to random value for $0 \leqslant i \leqslant 7$. (There are 2^{128} cases.) Then for randomly chosen m_{3i+2}'s, $m_0 \sim m_{35}$ satisfying the values of (4) $\sim$ (19) are automatically determined by the property of multi-permutation. A permutation $B : E^2 \to E^2$, $B(a,b) = (B_1(a,b), B_2(a,b))$, is a *multi-permutation* if for every $a, b \in E$ the mappings $B_i(a,*)$, $B_i(*,b)$ for $i = 1, 2$ are permutation on E. Therefore, we can get $m_0 \sim m_{35}$ satisfying $w_4 \sim w_7$ with complexity 1.

Second (Disseminative part): we repeat the first part of attack 2^{64} times. Then we can get a $m_0 \sim m_{35}$ satisfying $w_0 \sim w_7$ with complexity 2^{64}. We repeat this process 2^t times. Then we can get 2^t $m_0 \sim m_{35}$ satisfying $w_0 \sim w_7$ with complexity 2^{t+64}. We store such 2^t $(m_0, \cdots, m_{35}, d_0, \cdots, d_7)$.

Forth (Inverting part with complexity 1): Given a hash output $o_0||o_1||\cdots||o_7$, since the multipermutation is an invertable permutation, we can invert $o_0||o_1||\cdots||o_7$ upto the output of Step 11 by giving arbitrary random value to $m_{36} \sim m_{42}$. Then Since $w_0 \sim w_7$ is already fixed, (40)$\sim$(45) also are determined. And also through inverting process, (46) is also fixed. Here we give a value to m_{34} such that (45) and (46) are satisfied. Then we give arbitrary random values to m_{33} and m_{35}. Now we have the output of Step 5. Then we give a value to m_{31} such that (44) is satisfied. Then we give arbitrary random values to m_{30} and m_{32}. Then we give values to m_{27} and m_{28} such that (40) and (42) are satisfied. Then automatically, (36), (38) and (39) are determined. So, m_{26} and m_{24} are also determined. Then

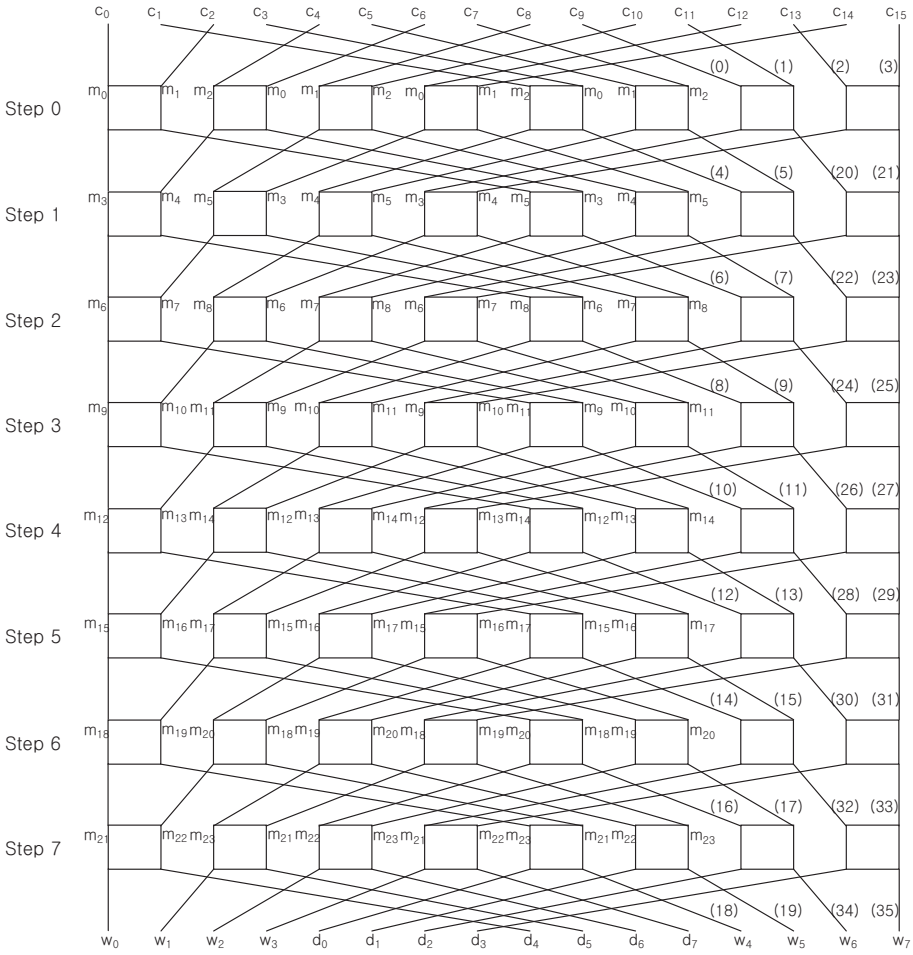

Fig. 3. First Part : Eight Steps.

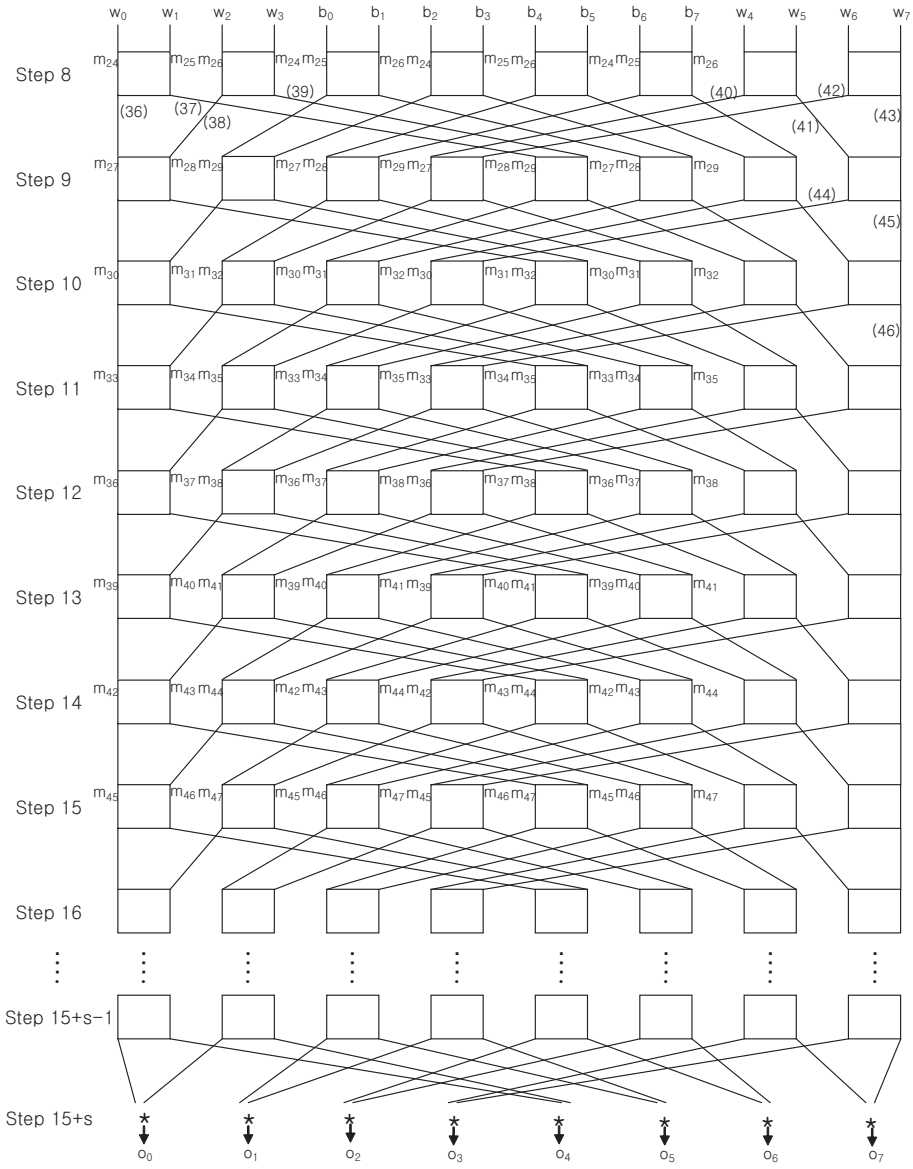

Fig. 4. Second Part.

with using the property of multi-permutation we give a value to m_{25} such that (36) is satisfied. Then (37) is automatically determined and then m_{29} is also determined. Therefore, we can get $m_{24} \sim m_{42}$ satisfying $w_0 \sim w_7$ with complexity 1. We know that since eleven words are free, there are 2^{176} $m_{24} \sim m_{47}$.

Fifth (Meet-in-the-middle attack part): For each $m_{24} \sim m_{47}$ satisfying $w_0 \sim w_7$, we get $b_0 \sim b_7$ and then check if $b_i = d_i$ for all i in the table stored at the third part of attack. we repeat the forth part of attack 2^{128-t} times.

Sixth (Connection part): According to birthday attack, with high probability we can find $(m_0, \cdots, m_{23}, d_0, \cdots, d_7)$ and $(m_{24}, \cdots, m_{47}, b_0, \cdots, b_7)$ such that $b_i = d_i$ for all i. Therefore, given a hash output $o_0||o_1||\cdots||o_7$, we can find a padded preimage $m_0 \sim m_{47}$ with complexity $2^{t+64} + 2^{128-t}$ and memory 2^t.

5 Conclusion

In this paper, we described a premiage attack on Parallel FFT-Hashing. Our attack does not depend on the value of s, which means that the security analysis of collision resistance of Parallel FFT-Hashing [4, 5] can not guarantee the security against the preimage attack. And also our attack can be used in case of any word size (in this paper, we only consider 16-bit word size) in the same way.

Acknowledgement

We thank prof. Moti Yung for his valuable discussions and naming "disseminative-meet-in-the-middle-attack".

References

1. T. Baritaud, H. Gilbert and M. Girault, *FFT Hashing is not Collision-free*, Eurocrypt'92, LNCS 658, Springer-Verlag, pp. 35-44, 1992.
2. C.P. Schnorr, *FFT-Hashing : An Efficient Cryptographic Hash Function*, Presented at the rump session of the Crypto'91.
3. C.P. Schnorr, *FFT-Hash II, efficient hashing*, Eurocrypt'92, LNCS 658, Springer-Verlag, pp. 45-54, 1992.
4. C.P. Schnorr and S. Vaudenay, *Parallel FFT-Hashing*, FSE'93, LNCS 809, Springer-Verlag, pp. 149-156, 1994.
5. C.P. Schnorr and S. Vaudenay, *Black Box Cryptanalysis of Hash Networks based on Mulitipermutations*, Eurocrypt'94, LNCS 950, Springer-Verlag, pp. 47-57, 1995.
6. S. Vaudenay, *FFT-Hash II is not yet Collision-free*, Crypto'92, LNCS 740, Springer-Verlag, pp. 587-593, 1993.