Preimage Attacks On Provably Secure FFT Hashing proposed at Second Hash Workshop in 2006

Donghoon Chang

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

Abstract. 'Provably Secure FFT Hashing' (We call FFT-Hash in this paper) was suggested by Lyubashevsky et al.. in Second Hash Workshop in Aug. 2006. This paper shows preimage attacks on hash functions based on three modes of FFT-Hash. In case of 'Nano' whose output size is 513 bits, we can find a preimage with complexity 2^{385}. In case of 'Mini' whose output size is 1025 bits, we can find a preimage with complexity 2^{769}. In case of 'Mini' whose output size is 28672 bits, we can find a preimage with complexity 2^{24576}. This means that the structure of FFT-Hash is weak in the viewpoint of the preimage resistance. We recommend that FFT-Hash can not be used in case of the output size less than 256 bits because the full security against the preimage attack are crucial in such a short output size. And also we should not chop the hash output in order to get a short hash output like SHA-224 and SHA-384, because for example we can find a preimage with complexity 2^{128} (not 2^{256}) in case of 'Nano' with chopping 257 bits whose hash output is 256 bits.

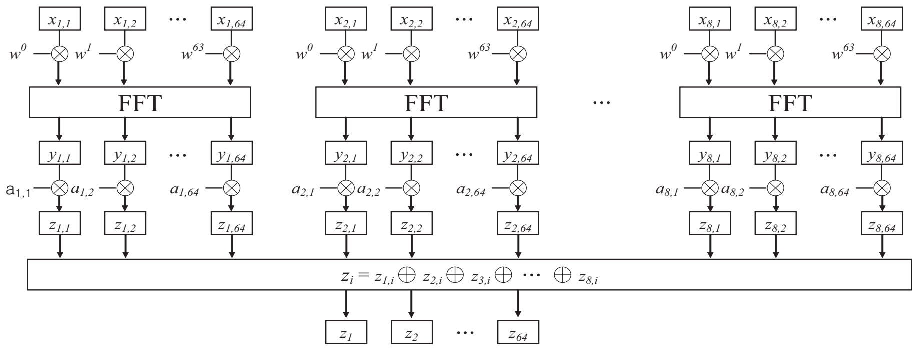
Keywords : Hash Function, Preimage Attack.

1 Introduction

Since MD4-style hash functions were broken, the research on the provably secure hash functions have been required. We can construct a hash function based on a hard problem. By the way, when we design a hash function based on a hard problem, it is difficult to construct a hash function satisfying several properties such as preimage resistance, collision resistance, second-preimage resistance, partial preimage resistance and pseudorandomness because we have to design a hash function with only one hard problem. FFT-Hash has provably secure against the collision attack. But FFT-Hash can not say the security against other properties. In this paper we show that FFT-Hash is not secure against the preimage attack. Especially, we should not use the chopping method in case of FFT-Hash like SHA-224 and SHA-256.
The organization of this paper is as follows. In section 2, we explain FFT-Hash. And then, in section 3, we show the preimage attacks on it. In section 4, we conclude.

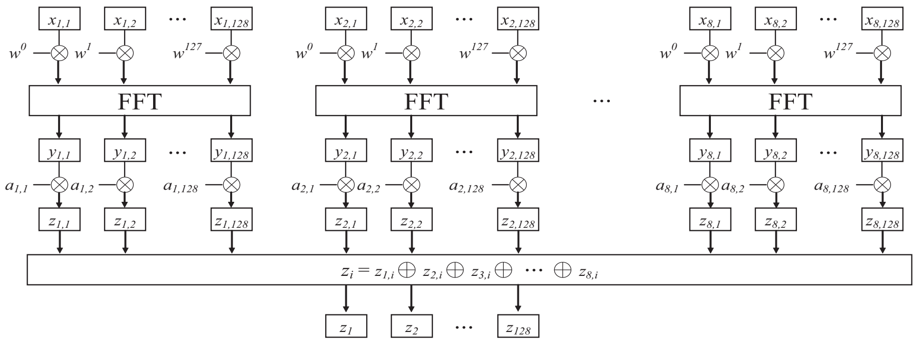
2 FFT-Hash

Lyubashevsky *et al.* [1] suggested 'FFT-Hash' in Second Hash Workshop in Aug. 2006. They suggested three modes which are 'Mini', 'Nano' and 'Bulk'. They are not hash functions but compression functions. They are keyed compression functions. 'Mini', 'Nano' are described in Fig. 1 and 2 where $a_{i,j}$'s are keys and w^i's are fixed constants and FFT operation is a bijection and can be efficiently invertible. In case of 'Bulk', $n = 1024$ and $m = 16$ and $d = 15$ and $p \approx 2^{28}$ and the input size is 65536 bits and the output size is 28672 bits.



$x_{1,1}$ $x_{1,2}$ ⋯ $x_{1,64}$
w^0 w^1 w^{63}
FFT
$y_{1,1}$ $y_{1,2}$ ⋯ $y_{1,64}$
$a_{1,1}$ $a_{1,2}$ $a_{1,64}$
$z_{1,1}$ $z_{1,2}$ $z_{1,64}$
$x_{2,1}$ $x_{2,2}$ ⋯ $x_{2,64}$
$y_{2,1}$ $y_{2,2}$ ⋯ $y_{2,64}$
$a_{2,1}$ $a_{2,2}$ $a_{2,64}$
$z_{2,1}$ $z_{2,2}$ $z_{2,64}$
$x_{8,1}$ $x_{8,2}$ ⋯ $x_{8,64}$
$y_{8,1}$ $y_{8,2}$ ⋯ $y_{8,64}$
$a_{8,1}$ $a_{8,2}$ $a_{8,64}$
$z_{8,1}$ $z_{8,2}$ $z_{8,64}$
$z_i = z_{1,i} \oplus z_{2,i} \oplus z_{3,i} \oplus \cdots \oplus z_{8,i}$
z_1 z_2 ⋯ z_{64}

Fig. 1. Nano ($n = 64$, $m = 8$, $d = 3$, $p = 257$) : $x_{i,j} \in \{0, 1, 2, 3\}$, $w^i, a_{i,j}, y_{i,j}, z_{i,j}, z_i \in \{0, 1, 2, \cdots, 256\}$. Input size is 1024 bits. Output size is 513 bits.



$x_{1,1}$ $x_{1,2}$ ⋯ $x_{1,128}$
w^0 w^1 w^{127}
FFT
$y_{1,1}$ $y_{1,2}$ ⋯ $y_{1,128}$
$a_{1,1}$ $a_{1,2}$ $a_{1,128}$
$z_{1,1}$ $z_{1,2}$ $z_{1,128}$
$x_{2,1}$ $x_{2,2}$ ⋯ $x_{2,128}$
$y_{2,1}$ $y_{2,2}$ ⋯ $y_{2,128}$
$a_{2,1}$ $a_{2,2}$ $a_{2,128}$
$z_{2,1}$ $z_{2,2}$ $z_{2,128}$
$x_{8,1}$ $x_{8,2}$ ⋯ $x_{8,128}$
$y_{8,1}$ $y_{8,2}$ ⋯ $y_{8,128}$
$a_{8,1}$ $a_{8,2}$ $a_{8,128}$
$z_{8,1}$ $z_{8,2}$ $z_{8,128}$
$z_i = z_{1,i} \oplus z_{2,i} \oplus z_{3,i} \oplus \cdots \oplus z_{8,i}$
z_1 z_2 ⋯ z_{128}

Fig. 2. Mini ($n = 128$, $m = 8$, $d = 3$, $p = 257$) : $x_{i,j} \in \{0, 1, 2, 3\}$, $w^i, a_{i,j}, y_{i,j}, z_{i,j}, z_i \in \{0, 1, 2, \cdots, 256\}$. Input size is 2048 bits. Output size is 1025 bits.

3 Preimage Attack on FFT-Hash

As we said in the previous section, FFT operation is a bijection and can be efficiently invertible. Using this property, we can find preimages of 'Nano', 'Mini' and 'Bulk'. In this section, we will show that the complexity to find a preimage of given t-bit hash output is $2^{t-n\cdot\log_2(d+1)}$.

Weak Property of FFT-Hash

With 'Nano', we explain the weak property of FFT-Hash. Other cases can be explained in the same way. In Fig. 3, we let $x_i = x_{i,1}||x_{i,1}||\cdots||x_{i,64}$. Given z_i ($1 \leqslant i \leqslant 64$) and x_j ($2 \leqslant j \leqslant 8$), x_1 is determined and we can find x_1 with complexity 1 because FFT operation is efficiently invertible. For any x_j, we can say in the same way.

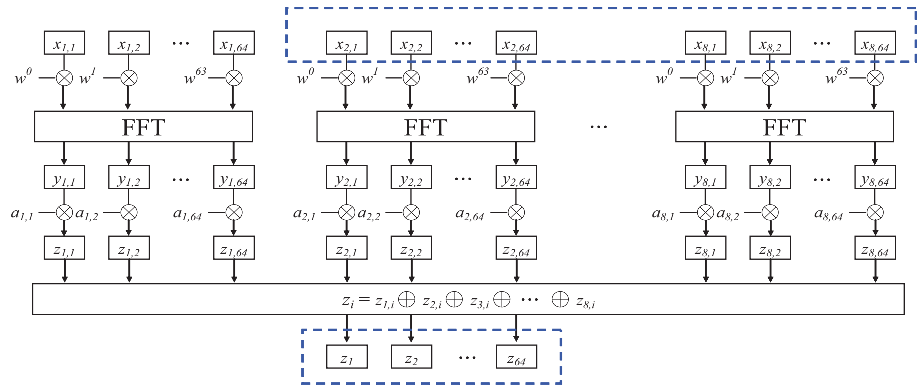


Fig. 3. Weak Property of Nano.

preimage attacks on FFT-Hash

Let f be each compression function. IV is the initial value. b is the message block size. In case of Nano, IV is 513 bits and $b = 511$. In case of Mini, IV is 1025 bits and $b = 1023$. In case of Bulk, IV is 28672 bits and $b = 36864$. Then we can find a padded 2-block preimage $g(M) = M_1||M_2$ of a given hash output O. Let g be a padding method. $g(x) = x||10^t||\mathsf{bin}_{64}(x)$ where t is the smallest non-negative integer such that $g(x)$ is a multiple of b and $\mathsf{bin}_i(x)$ is the i-bit binary representation of x.

Nano. We want to find a padded 2-block preimage $g(M) = M_1||M_2$ of a given hash output O where M is $2 \cdot 511 - 65$ bits and last 65 bits of M_2 is '$1||\mathsf{bin}_{64}(M)$'. We choose randomly M_1. Then we know the value of h (513 bits)

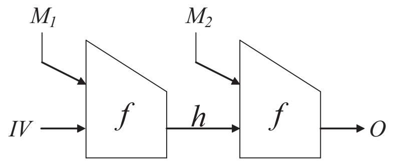


Fig. 4. Preimage Attack : Given a hash output O, we want to find a padded 2-block preimage $M_1||M_2$.

in Fig. 4. Then, we have 2^{446} cases of M_2 because last 65 bits are fixed. In Fig. 1, $h||M_2 = x_1||x_2||\cdots||x_8$ where x_1, x_2, x_3, x_4, the fist bit of x_4 and the last 65 bits of x_8 are fixed. So, except 128-bit x_6, we do the exhaustive searching with the complexity 2^{318}. For each case, x_6 is automatically determined as described above. So we can reduce the complexity 2^{128} for each of 2^{318} cases. If $x_{6,i} \in \{0, 1, 2, 3\}$ for $1 \leqslant i \leqslant 64$, then we have a preimage M of O. Since output size is 513 bits, in order to get a preimage on average, we have to repeat to choose M_1 randomly 2^{67} times. Therefore, the total complexity to find a preimage is 2^{385}.

Mini. We want to find a padded 2-block preimage $g(M) = M_1||M_2$ of a given hash output O where M is $2{\cdot}1023-65$ bits and last 65 bits of M_2 is '$1||\mathsf{bin}_{64}(M)$'. We choose randomly M_1. Then we know the value of h (1025 bits) in Fig. 4. Then, we have 2^{958} cases of M_2 because last 65 bits are fixed. In Fig. 2, $h||M_2 = x_1||x_2||\cdots||x_8$ where x_1, x_2, x_3, x_4, the fist bit of x_4 and the last 65 bits of x_8 are fixed. So, except 256-bit x_6, we do the exhaustive searching with the complexity 2^{702}. For each case, x_6 is automatically determined as described above. So we can reduce the complexity 2^{256} for each of 2^{702} cases. If $x_{6,i} \in \{0, 1, 2, 3\}$ for $1 \leqslant i \leqslant 128$, then we have a preimage M of O. Since output size is 1025 bits, in order to get a preimage on average, we have to repeat to choose M_1 randomly 2^{67} times. Therefore, the total complexity to find a preimage is 2^{769}.

Bulk. We want to find a padded 2-block preimage $g(M) = M_1||M_2$ of a given hash output O where M is $2{\cdot}36864-65$ bits and last 65 bits of M_2 is '$1||\mathsf{bin}_{64}(M)$'. We choose randomly M_1. Then we know the value of h (28672 bits) in Fig. 4. Then, we have 2^{36799} cases of M_2 because last 65 bits are fixed. And $h||M_2 = x_1||x_2||\cdots||x_8$ where $x_1 \sim x_7$ and the last 65 bits of x_{16} are fixed. So, except 4096-bit x_8, we do the searching with the complexity 2^{24576}. For each case, 4096-bit x_6 is automatically determined as described above. So we can reduce the complexity 2^{4096} for each of 2^{24576} cases. If $x_{6,i} \in \{0, 1, 2, 3\}$ for $1 \leqslant i \leqslant 128$, then we have a preimage M of O. Since the output size is 28672 bits, the total complexity to find a preimage is 2^{24576}.

preimage attacks on FFT-Hash with Chopping

Since the output sizes of three modes of FFT-Hash are big, the chopping of the hash output may be used like SHA-224 and SHA-384. However, we should not use the chopping method in case of FFT-Hash. Even though we use the chopping method, in case of each mode, the complexity to find a preimage of given $t-s$ hash output where s-bit is chopped is $2^{t-s-n\cdot\log_2(d+1)}$. When the mode is Nano ($t=513$, $n=64$ and $d=3$) and $s=257$, then the complexity is 2^{128}.

4 Conclusion

Standard Hash Function may be used in many areas. So we need to design a hash function satisfying several properties. And also the hash function should be efficient. Even though FFT-Hash is provably secure against the collision attack, FFT-Hash is not secure against the preimage attack and also not efficient. Therefore, in order to use FFT-Hash practically, FFT-Hash should be modified, which may make the security proof difficult.

References

1. Second Hash Workshop held by NIST, Aug. 2006. You can download all papers and presentations from http://www.csrc.nist.gov/pki/HashWorkshop.