Cryptanalysis of Cubehash

Vikash Jha
Aalto School of Science and Technology
vjha@cc.hut.fi

Abstract

This paper briefly explains the cryptographic hash function CubeHash.It analyzes the strength of CubeHash against the various cryptanaysis technique.The paper summarizes all the effective attacks that has been designed against the reduced version of the CubeHash.Along with this, paper analyzes brute force attack on CubeHash with large message size.Beside this, relatively new techniques of applying SAT solvers to attack hash function is explored in context of CubeHash.

1 Introduction

The cryptographic hash function is a sequence of steps that can process an arbitrary block of data to produce a fixed length bit strings such that any change in the data block must produce a different bit string. The hash functions are primarily used in digital signatures message authentication and calculation of checksums etc. Currently, the cryptographic hash functions such as MD5 , SHA-1 and SHA-2 are mainly used. But of late, these hash functions have become vulnerable to different attacks as collision, failing second preimage resistance and thus forcing the need for new cryptographic hash algorithm. The national institute of standards and technology (NIST) has organized a competition to develop a new cryptographic hash algorithm which will be called SHA-3. CubeHash family of hash functions proposed by Daniel J. Bernstein is one of the submission which has advanced to second round of the competition. This paper discusses the strength of CubeHash algorithm against the various attacks and feasibility of those attack under current processing power constraints.

2 Hash Function Security

Any cryptographic hash function must be resistant to following attacks.

- Preimage attack: Given a hash value H, attackers try to find a message having the value of hash i.e H = hash(m)
- 2nd-preimage attack: Given a input m attackers try to find 2nd-preimage $m' \neq m$ such that h(m′) = h(m).
- Collision attack: Attackers try to find two different messages m and m′ such that hash(m′) = hash(m)
- Multicollison: Attackers try to generate a series of messages m_1, m_2, ... m_N, such that hash(m_1) = hash(m_2) = ... = hash(m_N).

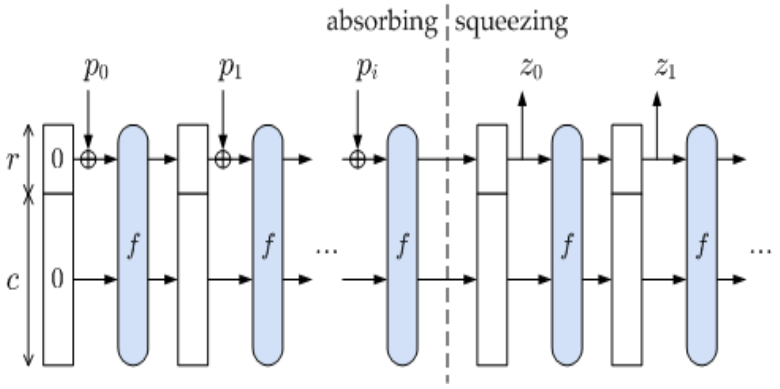

Figure 1: Sponge Construction [5]

Birthday paradox established that it is not possible to avoid collision attack. But if a cryptographic hash function makes the problem space large enough to be computed(searched) by the current available processors then it can be said that the cryptographic hash function is computationally secure.

3 CubeHash

CubeHashr/b-h has three parameters namely r specifying the number of state transformation rounds to be performed, b number of bytes per message block , and h the length of the hash bits.Apart from these parameters CubeHash maintains an internal state of 1024 bits.First round NIST submission recommends r = 8 , b = 1 and h ∈ {224,256,384,512} [3]. Later it was changed to r=16, b=32 for the second round[4]. CubeHash uses sponge construction, which is an iterated construction for giving an output of arbitrary length by taking a variable-length input and subjecting it to a fixed-length transformation (or permutation) [5]. The sponge construction is represented in figure-1[5]. As we can see the sponge construction operates on a state of b=r+c bits. In case of CubeHash r is nothing but message bytes b and c is the hidden bytes i.e. (128-b).

The CubeHash algorithm has five major steps : [4, 3]

- Initialize a 128-byte (1024-bit) state as a function of (h, b, r).
- Convert the input message into a padded message. The padded message consists of one or more b-byte blocks.

- For each b-byte block of the padded message: xor the block into the first b bytes of the state, and then transform the state invertibly through r identical rounds.
- Finalize the state.
- Output the first h/8 bytes of the state.

The state transformation involves following steps: [4, 3, 2]. State transformation maps to the function f in the sponge construction.

```
for i=0,....,15: x[i+16]=x[i+16]+x[i]
for i=0,....,15: y[i⊕8]=x[i]
for i=0,....,15: x[i]=y[i]⋘7
for i=0,....,15: x[i]=x[i]⊕x[i+16]
for i=0,....,15: y[i⊕2]=x[i+16]
for i=0,....,15: x[i+16]=y[i]
for i=0,....,15: x[i+16]=x[i+16]+x[i]
for i=0,....,15: y[i⊕4]=x[i]
for i=0,....,15: x[i]=y[i]⋘11
for i=0,....,15: x[i]=x[i]⊕ x[i+16]
for i=0,....,15: y[i⊕1]=x[i+16]
for i=0,....,15: x[i+16]=y[i]
```

4 Attacks

In this section, we will go through the various attacks that has been employed against CubeHash and we will subsequently measure the feasibility of these attacks. NIST is evaluating each and every SHA-3 candidate with respect to SHA-2, so we will also present a comparison with respect to collision and preimage attack between the SHA-2 and CubeHash.

4.1 Generic attack

First,we will discuss two primitive cryptanalysis technique that can be used to attack CubeHash(or any invertible hash algorithm) .As we have seen in the above section, CubeHash is the iteration of transformation of state and Xoring of first b bytes of state and message.It is easier to attack CubeHash for larger values of b and smaller rounds r as we will see in the two methods presented below. The first method which has been presented by the author of the CubeHash can be classified under generic attack for finding preimage of any message signed by CubeHash.[4]

- from (h, b, r) compute the initial state S_0
- from the h-bit image plus some arbitrary (1024-h) bits, invert 10r rounds and the "xor 1" to get a state S_f before finalization
- find two n-block sequences that map S_0 (forward) and S_f (backward), respectively, to two states that share the last (1024-8b) bits

If we analyze the above method then one has to look for 2^{nb} possible n-block input and search for a collision over (128-b)*8 bits.Using birthday paradox we can find the collision in 128-b part in $2^{((128-b)*8)/2}$. For larger message size, this problem space is quite small and preimage of CubeHash can be found easily but for small message size block the problem space grows exponentially and hence finding the preimage becomes very hard.

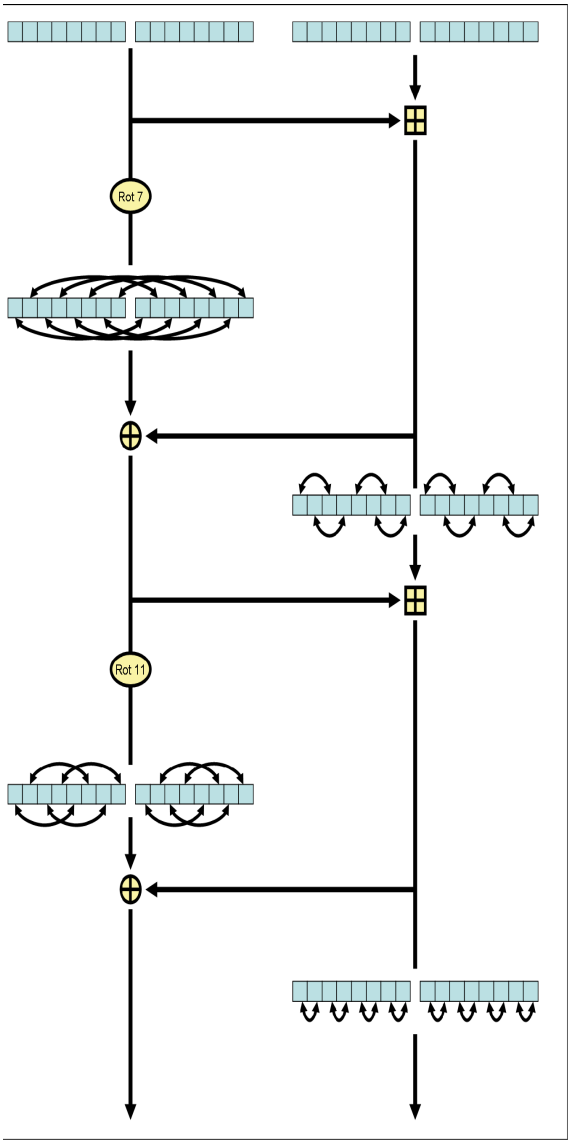

Figure 2: Schematic representation of state transformation

Second attack can emanate from the concept of multicollison [2]. From the initial state of S_0 derived from (h,b,r) one can find two n block sequence m and m′ that yield $state_0$ (from forward direction) and the zero state(from backward direction) respectively, to two intermediate states that share (128-b) bytes. This can be represented with following equations. [1]

$$S_0 \oplus m_1 \rightarrow S_1$$
$$S_1 \oplus m_2 \rightarrow .....$$
$$..... \rightarrow S_1$$
$$S_1 \oplus m'_2 \rightarrow 0 \oplus m'_1$$

Once we find any intermediate state whose last (128-b) bytes matches with the any of the previous state, we can form a colliding messages of the form.

m||m′||0||0.....||0||M

where M is arbitrary sequence of message. This attack is also of the complexity 2^{512-4b}. Now, keeping in view the above results we will compare CubeHash with the SHA-2 under the requirement of SHA-3.There has been successful collision and preimage attacks on the reduced version of SHA-2 e.g. preimage attack consisting of 19 steps(SHA-256 , SHA-512 consists of 64 steps and 80 steps respectively)[16].Later, this was extended to preimage attack on 46 steps for SHA-512 [18].The complexity of this attack is $2^{511.5}$ computation and 2^3*10 words of memory.Considering the fact that SHA-512 has only 80 steps , it is very likely that there will be improvements in future and hence weakening the SHA-2 further.CubeHash-512 has the best collision attack for 5/64(64 rounds-r and 5 bytes-b),having complexity of 2^{231}. CubeHash-512 has the preimage attack for 2^{513-4b} for any r/b. When we compare both SHA-512 and CubeHash-512, we can see that for increased number of rounds for CubeHash the collision attack seems to be very difficult.The submitted version of CubeHash-16/32 does not have a feasible collision attack and a preimage attack of greater than 2^{384} complexity. One of the advantages with CubeHash has that it is a parametrized algorithm so the complexity of the attack can be increased by increasing the number of rounds r or by decreasing number of bytes b to make it strong against the collision and preimage attack.

4.2 Exploiting symmetry of round function

Specification document of CubeHash has mentioned that the initialization values for the state bytes have been selected to avoid the symmetry of the round functions [4, 3].Aumasson et. al[2] has found 15 symmetries classes of 2^{512}(16 different bytes in each class) states each.

The 15 symmetries found by Aumasson et.al are[2, 14]:

```
C1 : AABBCCDD EEFFGGHH IIJJKKLL MMNNOOPP
C2 : ABABCDCD EFEFGHGH IJIJKLKL MNMNOPOP
C3 : ABBACDDC EFFEGHHG IJJIKLLK MNNMOPPO
C4 : ABCDABCD EFGHEFGH IJKLIJKL MNOPMNOP
C5 : ABCDBADC EFGHFEHG IJKLJILK MNOPNMPO
C6 : ABCDCDAB EFGHGHEF IJKLKLIJ MNOPOPMN
C7 : ABCDDCBA EFGHHGFE IJKLLKJI MNOPPONM
C8 : ABCDEFGH ABCDEFGH IJKLMNOP IJKLMNOP
C9 : ABCDEFGH BADCFEHG IJKLMNOP JILKNMPO
C10: ABCDEFGH CDABGHEF IJKLMNOP KLIJOPMN
C11: ABCDEFGH DCBAHGFE IJKLMNOP LKJIPONM
C12: ABCDEFGH EFGHABCD IJKLMNOP MNOPIJKL
C13: ABCDEFGH FEHGBADC IJKLMNOP NMPOJILK
C14: ABCDEFGH GHEFCDAB IJKLMNOP OPMNKLIJ
C15: ABCDEFGH HGFEDCBA IJKLMNOP PONMLKJI
```

If a state belongs to several classes then all its images(transformations, we will denote this by T) by the transformation function will also belong to transformation of these classes i.e. if S $\in$($C_i \cap C_j$), then T(S) $\in$($C_i \cap C_j$). We can exploit these symmetries for finding preimages. If we have a target hash digest we can find preimage for that hash as follows[2]:

- reach a symmetric state in forward direction of round function (of any class)
- reach (from backwards) another symmetric state (not necessarily of the same class)
- from these two symmetric states in classes C_i and C_j use null message blocks in both directions to reach two states in $C_i \cap C_j$
- find a collision by trying $\sqrt{x}$, (x=$C_i \cap C_j$ messages in each direction.

The complexity of this attack depends on the C_i considered. We will see the above step with an example, consider the symmetry class C_1 and message size of 5 bytes i.e. b=5.So our message will be of form X000X and we always need to preserve the equality s[i]=s[i+1]. Since each message contains 2^{512} states , so there has to be at least 2^{256} trials in each direction. We can also find collisions by exploiting the symmetry if we are allowed to initialize the state bytes such that they were symmetric and in $C_1 \cap C_2 \cap C_4 \cap C_8$ [2, 14]. For example if the the initial sates of the CubeHashr/b-h is of the form

AAAAAAAA AAAAAAAAA BBBBBBBBB
BBBBBBBB

then then each of the 2^{33} intermediate states for the message sequence of zeroes will also be an element of $C_1 \cap C_2 \cap C_4 \cap C_8$. We can find two same states with a probability of 0.63 , thus giving collision(assuming T behaves randomly).To summarize, this attack tries to find the symmetric state in the round function of CubeHash and then follows the method described in the generic attack to find the collision. The symmetry in the state word(if we can find any) helps in reducing the complexity of the attack. Ferguson et. AL [14] has described an improved attack using the above mentioned symmetry classes which is also based on the method described above.

5 Truncated Differential Path

One of the technique that has been utilized to cryptanalyze the hash function is truncated differences. This has been first studied by Kundsen[15] and is later extended by Peyrin et. AL [7]. In this technique the basic idea is to search for the non zero differences in the input and subsequent output states. For example if we have two input state words which differ say at word 5 and the resulting output word differ at 7 then we say that we have a differential path. if we explain it further with an example, representing each state word with a bit, then we will be able to represent the state word of CubeHash by 4 bytes.Suppose we have a relation as 0x05000000 $\rightarrow$ 0x00000800 then we can say that difference in word 5 and 7 in the input side resulted in difference of word 20 at the output side.We will denote such differential path by

$$A \rightarrow B$$

and the individual state word by X_i. For CubeHash-r/36 Brier et. AL[10] analyzed two differential path

$$\text{path}_1\text{: 0xa8000000} \rightarrow \text{0x0a020000}$$
$$\text{path}_2\text{: 0xa8000000} \rightarrow \text{0x0a020000}$$

From the above path we can see difference between the two state words is contained in only first 9 words. This fact can be used to launch collision attack on the CubeHash-r/36 easily.At the start of each iteration, we have control on X_0, X_1...X_8 so we can carefully select these words to have the same hash value for the two different message. Path $_1$ and path $_2$,when analyzed for one round function will generate a system of equation depicting their interdependency.Now, to verify the selection of message we just need to check the correctness of the equation for our selected pair of message.For example path$_1$ evaluates to

$Y \oplus (A+Y) = X_0 \lll 7 \oplus (B+X_0) \lll 7$ where A,B are constant, Y is the predetermined value and X_0 we can select such that it satisfies the above equation

One of the trick to solve the above equation is, if we take the right hand side of the equation then $Y \oplus (A+Y)$ is equal to 0xffffffff when y= $\overline{A}/2$ and least significant bit of A is 1, same is true for left hand side. So we can check for the valid A and B to find the colliding message. The above differential path is searched by hand for the message block size (b) of 36 bytes. But when the b reduces, finding the differential path becomes tougher. The idea over here is to find a differential path as above that has differences of state words contained within the length of message blocks. We can find such differential path by writing a program that heuristically looks for such truncated differential paths. Once the truncated differential path is settled, then we can use a program that will search for an input pair that verifies this path. Applying heuristic to search for such path has always resulted in long time and frequent stack overflows for smaller message blocks.

5.1 Linear Differential Path

Linear differential path has been the most simple attack on the CubeHash. The state transformation of CubeHash involves all linear operation except for the addition, but addition of two 32 bit words A and B can be replaced by xor as is done in various previous work to analyze hash functions[12, 11].In this technique first we insert the message pair with known bit difference and observe the difference in the state words after round function. For the next iteration we again choose message pair to erase all the difference caused in the previous round. This will lead to internal collision. We will illustrate this with an example, suppose we have inserted a single bit difference at position y then we will have to select the message pair that erases the difference at y+4,y+14, y+22 (all modulo 32).Similarly continue till the last round, erasing the last byte difference.This sounds to be simple for the linear operations but CubeHash has two nonlinear modular addition operation so the input differential may not satisfy the output differential constraint because of the carry generated by the addition of words. Basically there are two possibilities because of the carry, it can either create a difference or correct a difference at a particular bit position. This requires that whenever addition of two pair of words in CubeHash round function occurs we need to verify these two conditions for the entire differential path. Verification of these condition will ensure linear behavior of operation.We can also represent this condition mathematically e.g.we have A, B and A′ B′ then effect of non-linear addition can be handled all the taking into account of all the condition which is equal to hamming weight of $(A \oplus A') \vee (B \oplus B')$.The number of differential path depends upon the initial bit where the difference was first inserted e.g. for 0^{th} bit there will be total of 2^{46} conditions for total of two round of iteration. Using this attack Dai and Brier et. AL [11, 10] found collision attacks on CubeHash-2/89 and CubeHash-4/48 respectively.We can illustrate our attack with an example to make the concept more clear. We have these three differential path for 64 bytes and 4 rounds(since there is three differential path).

D_0 =
00000008000000000000000800000000
00000000000000000000000000000000
04000000000000000000000000000000
00000000000000000000000000000000

D_1 =
88008000000000008800800000000000
00000000000000000000000000000000
00440040000000000000000000000000
00000000000000000000000000000000

D_2 =
08000000000000000800000000000000
00000000000000000000000000000000
00040000000000000000000000000000
00000000000000000000000000000000

then the D_2 will erase the difference caused in state words by D_0 and D_1. We can have the two colliding message of the form

Message1= $M_0 \| M_1 \| ZERO$
Message2 = $(M_0 \oplus D_0) \| (M_1 \oplus D_1) \| D_2$

We need to continue the process of XORing and appending the two consecutive differential path till we get the collision.

5.2 Attacks using SAT solver

Boolean satisfiability(SAT) is one of the relatively new and unexplored technique in attacking the hash functions.Boolean satisfiability involves converting expression into a boolean formulae e.g Conjecutive Normal Form (CNF) and then checking if there exists any satisfiable assignment for the boolean expression.In case of CubeHash, we only need to convert the addition and XOR operation of round function because rotation and swap operation of the CubeHash just interchange the bits without affecting the truth table for the expression. The idea here is to represent the 128 bytes state of the round function as 1024(128*8) literals. We will take two message and create the CNF for these two messages and check whether there is any common satisfiable assignments for the two CNF.If we are able to find any such assignment then we have a collision. There are many SAT solver e.g Davis-Putnam-Longemann-Loveland (DPLL) algorithm, SAT solvers basically work by employing branching heuristics to look for the solution [13]. One of

Round	CNFs
1	18,024
2	36,048
3	54,072

Table 1: Number of CNF per round

the branching heuristics that can be used by DPLL is Maximum occurrences in clause of minimum sizes(MOMS).This approach is preferred because it finds the non-reachability of solution faster and is easier to implement as compared to others. The major problem while finding colliding using SAT solvers is that number of CNFs becomes larger with just few rounds as shown in below table-1. There are some solutions suggested in e.g using parallel DPLL and pruning the tree at regular intervals [17], using glucose(the most efficient SAT solvers), but none of these solutions have been able to scale the attacks to higher rounds[6]. The maximum round that the implementation with DPLL with MOMS could achieve was 2.

6 Results

We conducted few experiments with large values of message block(b=128-122 bytes,r=8),below(Figure 3) is the plot depicting the time taken to find the second pre-image for messages of length ranging from 128 bytes to 120 bytes. As it can be seen from the plot that finding preimage for

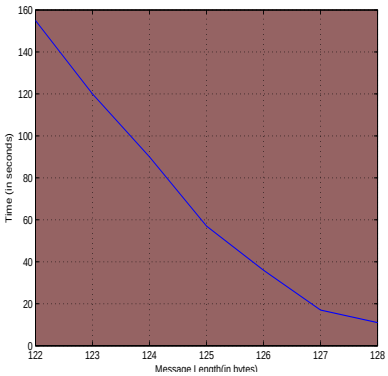

Figure 3: No of message bytes v/s Time taken

the large message blocks is easy but it becomes infeasible after message block size becomes less than than 120 bytes.The most impressive results were achieved using differential cryptanalysis methods . Brier et.AL. [8, 9] used differential cryptanalysis to find the collision for CubeHash-4/48 in total 2^{37} computational.Despite all the encouraging cryptanalysis result with the reduced version of the CubeHash,it seems nearly infeasible to break the submitted version of the CubeHash(b=16, r=8,h=512)and along with the fact that CubeHash is as fast as SHA-2.In

7 Conclusion

CubeHash is extensively parametrized hash algorithm which gives flexibility to apply it according to the situation. For the message of size 1 byte and 8 bytes , no successful cryptanalysis has come to the light to date.In current paper, we have presented various methods of attacking CubeHash for the reduced versions and have conducted few experiments for larger message size with an intention to measure the time to find the preimage.It has been fairly easy to get the preimage for larger block size(b), but as the block size reduces the attack using these two methods becomes infeasible. Examination of attacks using SAT solvers lead to the conclusion that it is very difficult to find the collision for higher number of rounds because the solution tree increases exponentially.The other methods that have been discussed in this paper has been able to find preimage or collision attack for the slightly reduced version of CubeHash. If these attacks are extended to the submitted version of CubeHash, then they result in infeasible complexity ($> 2^{384}$).

References

[1] J.-P. Aumasson. Collision for cubehash2/120-512. NIST mailing list (local link), 2008.

[2] J.-P. Aumasson, E. Brier, W. Meier, M. Naya-Plasencia, and T. Peyrin. Inside the hypercube. In C. Boyd and J. González Nieto, editors, *Information Security and Privacy*, volume 5594 of *Lecture Notes in Computer Science*, pages 202–213. Springer Berlin / Heidelberg, 2009. 10.1007/978-3-642-02620-1_14.

[3] D. J. Bernstein. Cubehash specification (2.b.1). Submission to NIST (Round 1), 2008.

[4] D. J. Bernstein. Cubehash specification (2.b.1). Submission to NIST (Round 2), 2009.

[5] G. Bertoni, J. Daemen, M. Peeters, G. V. Assche, G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche. Assche. sponge functions, 2007.

[6] B. W. Bloom. Sat solver attacks on cubehash. Technical report, Rochester Institute of Technology, Apr. 2010.

[7] E. Brier, S. Khazaei, W. Meier, and T. Peyrin. Linearization framework for collision attacks: Application to cubehash and md6. Cryptology ePrint Archive, Report 2009/382, 2009. urlhttp://eprint.iacr.org/.

[8] E. Brier, S. Khazaei, W. Meier, and T. Peyrin. Real collisions for cubehash-4/48. NIST mailing list (local link), 2009.

[9] E. Brier, S. Khazaei, W. Meier, and T. Peyrin. Real collisions for cubehash-4/64. NIST mailing list (local link), 2009.

[10] E. Brier and T. Peyrin. Cryptanalysis of cubehash. Available online, 2009.

[11] F. Chabaud and A. Joux. Differential collisions in sha-0. In H. Krawczyk, editor, *CRYPTO*, volume 1462 of *LNCS*, pages 56–71. Springer, 1998.

[12] W. Dai. Collisions for cubehash1/45 and cubehash2/89. Available online, 2008.

[13] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem-proving. *Commun ACM*, 5:394–397, July 1962.

[14] N. Ferguson, S. Lucks, and K. A. McKay. Symmetric states and their structure: Improved analysis of cubehash. Cryptology ePrint Archive, Report 2010/273, 2010. urlhttp://eprint.iacr.org/.

[15] L. R. Knudsen. Truncated and higher order differentials. In *Fast Software Encryption - Second International Workshop, Leuven, Belgium, LNCS 1008*, pages 196–211. Springer-Verlag, 1995.

[16] F. Mendel, N. Pramstaller, C. Rechberger, and V. Rijmen. Analysis of step-reduced sha-256. In *FSE*, volume 4047 of *LNCS*, pages 126–143. Springer, 2006.

[17] O. Salaev and S. Rao. Logical cryptanalysis of cubehash using a sat solver, May 2009.

[18] Y. Sasaki, L. Wang, and K. Aoki. Preimage attacks on 41-step sha-256 and 46-step sha-512. Cryptology ePrint Archive, Report 2009/479, 2009. `http://eprint.iacr.org/`.