A New Distinguisher for CubeHash-8/b and CubeHash-15/b Compression Functions

Javad Alizadeh[1] and Abdolrasoul Mirghadri[2]

[1] Faculty and Research Center of Communication and Information Technology, IHU
Tehran, Iran

[2] Faculty and Research Center of Communication and Information Technology, IHU
Tehran, Iran

Abstract

CubeHash is one of the round 2 candidates of the public SHA-3 competition hosted by NIST. It was designed by Bernstein. In this paper we find a new distinguisher to distinguish CubeHash compression function from a random function. This distinguisher principle is based on rotational analysis that formally introduced by Khovratovich and Nikolic. In order to use this technique, we need to compute the probability that four swap functions in CubeHash round function preserve the rotational property for any input pair. We compute these probabilities and find a new distinguisher that distinguish CubeHash-8/b and CubeHash-15/b compression function from a random function with probability greater than $2^{-538.5}$ and $2^{-1009.7}$, respectively. Until we know this is the first distinguisher for CubeHash compression function with more than 14 rounds.

Keywords: *SHA-3 candidate, CubeHash, rotational analysis, distinguisher.*

1. Introduction

Hash functions have a very important role in modern cryptography that used in many areas as digital signatures and various forms of authentication. A hash function is a transformation which maps a variable-length input to a fixed-size output, called message digest. After developments in the field of hash function cryptanalysis [13, 14, 15] along with new results targeted against commonly used hash functions has urged National Institute of Standards and Technology to announce a competition for the development of a new hash standard, SHA-3 [10].

The SHA-3 competition attracted a lot of attention in the cryptographic community. A total number of 64 hash function proposals was submitted, 51 of them advanced to the first round, 14 of them to the second round, and 5 of them advanced to the third, final round. One of the main requirements was the evaluation of the security of the submitted primitive, i.e. providing a detailed analysis on the resistance of the function against various attacks. Being one of the most powerful forms of attacks, the differential [4] and linear [9] analysis got most of the submitter's focus, while less attention was put on the other attacks.

Rotational analysis is a relatively new type of attack. Although this technique was mentioned and applied in previous works, such as [6], but it formally introduced by Khovratovich and Nikolic in [7]. It is also used for cryptanalysis of modified versions of BMW [5] and SIMD [8] hash functions in [11]. Unlike differential analysis, where for a pair (x,y), the attacker follows the propagation of the difference $x \oplus y$ through some transformation, in rotational analysis the adversary studies the propagation of the rotational pair $(x, x <<< r)$ through the transformation.

Khovratovich and Nikolic in [7] analyze the primitives composed of only three operations: addition, rotation, XOR (ARX). For these primitives, they prove that the probability that a rotational pair of inputs will produce a rotational pair of outputs depends only on the number of additions.

Previous Results on CubeHash: We refer to part 2. B. 5 of [2] and CubeHash profile in the SHA-3 Zoo [12] for a complete survey of cryptanalytic results on CubeHash. The currently best distinguisher attacks on CubeHash-r/b compression function for r=11 and r=14 were presented by Ashur and Dunkelman in [1]. They present a distinguisher of complexity 2^{470} for Cubehash-11/b compression function and another distinguisher of complexity 2^{812} for Cubehash-14/b compression function. In this work they use linear cryptanalysis technique [9] and linear approximations of CubeHash. Until we know no distinguisher for more than 14 rounds was presented so far.

Contribution of this Paper. The goal of this work is to extend the application of rotational analysis to CubeHash [2], one of the second round of the SHA-3 competition candidates that have four swap transformations other than additions, rotations, and XORs. In particular, we find the rotational probabilities of the four swap transformations in the CubeHash round function. This allows us that for several round parameters r and message block sizes b we present a new distinguisher for CubeHash-r/b compression function. Specially, we find a distinguisher for CubeHash-8/b and CubeHash-15/b compression functions with probability greater than $2^{-538.5}$ and $2^{-1009.7}$, respectively. In CubeHash-r/b, $b \in \{1,2,3,...,128\}$ and if in CubeHash-8/b compression function we set b=1, indeed we distinguish compression function of CubeHash-8/1, the first version of CubeHash that suggested to SHA-3 competition, from a random function. Also if in CubeHash-15/b we set b=32, we distinguish a reduced version of compression function of CubeHash-16/32 [3], only with one round less, from a random function. CubeHash-16/32 is a tweaked version of CubeHash-8/1 which is about 16 times faster than it and after presentation of cryptanalysis results on CubeHash-8/1 announced the official proposal for all digest lengths h=224, 256, 384 or 512.
Until we know this is the first distinguisher for CubeHash compression function with greater than 14 rounds.

Organization: In section 2 we review the concept of a distinguisher that use rotational analysis technique presented in [7]. In Section 3 we describe the hash function CubeHash and define its compression function. Section 4 specifies the probabilities that four swap functions in CubeHash round function preserve the rotational property for any input pair. In Section 5 we analyze CubeHash-8/b and CubeHash-15/b compression functions and present a new distinguisher for them. We conclude in Section 6 finally.

2. Distinguishers with Rotational Analysis

A rotational distinguisher explores the idea that some transforms on rotated inputs produce rotated outputs. Let $(X, X <<< r)$ be a pair of input words, it called a rotational pair, for some transform F, where $<<< r$ is a cyclic rotation to the left by r bits. If for an arbitrary input X, $F(X <<< r) = F(X) <<< r$ then it said that F preserves the rotational property, in other words, when the input composes a rotational pair, the output also composes a rotational pair. The input and the output of a transform can be a single word or a vector of words, i.e. $\tilde{X} = (X_1,...,X_n)$. Then, a rotational input/output pair is defined as $(\tilde{X}, \tilde{Y})$, where $Y_i = X_i <<< r, i = 1,...,n$. A system $\Phi(\tilde{X})$ composed of transforms $F_1,...,F_k$ preserves the rotational property, if on rotational input pair, produces a rotational output pair.
It is better that we attend two important issues. First, unlike differential analysis where usually out of the whole input pair only a few words have differences, in rotational analysis all the input pairs of words have to compose rotational pairs. Second, there are a few transforms that preserve the rotational property for any input pair. Usually, for an arbitrary X, $F(X <<< r) = F(X) <<< r$ only with some probability p_F, further called a rotational probability of F, that depends on the rotational amount r. If we assume that the outputs of the transforms are independent, then a system Φ composed of transforms $F_1,...,F_k$ preserves the rotational property with a probability $p_\Phi = p_{F_1} \cdot p_{F_2} .....p_{F_k}$ [11]. Hence, in order to find the probability that a system preserves the rotational property, one only has to find the probabilities that each instance of the underlying transforms preserves this property. For a random system with n-bit output, the probability that a rotational input will produce rotational output is 2^{-n}. Therefore, if a system Φ with n-bit output has a rotational probability $p_\Phi > 2^{-n}$, then this system can be distinguished from a random system [11].

2.1 Rotational Analysis of ARX Constructions

A thorough rotational analysis of ARX systems was given in the work of Khovratovich and Nikolic [7]. These systems are composed only of three transforms: addition, rotation and XOR. For each of them, the probabilities they preserve the rotational property were given by:

Lemma 1 (Addition): For n-bit words x, y, and a positive integer r

$$\Pr[(x+y) <<< r = x <<< r + y <<< r] = \frac{1}{4}(1 + 2^{r-n} + 2^{-r} + 2^{-n}).$$

Lemma 2 (Rotation): For n-bit word x and positive integers r, r'

$$\Pr[(x <<< r) <<< r' = (x <<< r') <<< r] = 1$$

Lemma 3 (XOR): For n-bit words x, y, and a positive integer r

$$\Pr[(x \oplus y) <<< r = x <<< r \oplus y <<< r] = 1$$

Hence, rotations and XORs preserve the rotational property with probability 1, while the probability of addition depends on the size of the words and the rotation amount. Further in our analysis, the rotation amount will be fixed to 1.

The proofs of lemma 1 and 2 are simple, and the proof of lemma 3 can be finding in [7].
In [11], the authors find the probabilities that subtractions, shifts and bitwise Boolean functions preserve the rotational property, too, and in this paper we find this probability for four swap functions used in CubeHash.

3. CubeHash Description

CubeHash [2] is Bernstein's proposal for the NIST SHA-3 competition [10]. CubeHash works with 32-bit words (n=32) and uses three simple operations of XOR, rotation and modular addition and four swap functions showed by $SWAP_1, SWAP_2, SWAP_3$, and $SWAP_4$. It has an internal state $S = (S_0, S_1, ..., S_{31})$ of 32 words and its variants, denoted by CubeHash-r/b, are identified by two parameters $r \in \{1,2,...,128\}$ and $b \in \{1,2,...,128\}$ which at each iteration process b bytes in r rounds. Selecting different values of r and b, allow the selection of a range of security/performance tradeoffs. The internal state S is set to a specified value which depends on the digest length (limited to 512 bits) and parameters r and b. The message to be hashed is appropriately padded and divided into b-byte message blocks. At each iteration one message block is processed as follows. The 32-word internal state S is considered as a 128-byte value and the message block is XORed into the first b bytes of the internal state. Then, the following fixed permutation is applied r times to the internal state to prepare it for the next iteration. This permutation called CubeHash round function and denoted by ROUND in this paper.

1. Add S_i into $S_{i\oplus 16}$, for $0 \le i \le 15$.
2. Rotate S_i to the left by seven bits, for $0 \le i \le 15$.
3. Swap S_i and $S_{i\oplus 8}$, for $0 \le i \le 7$ (We call this swap function $SWAP_1$).
4. XOR $S_{i\oplus 16}$ into S_i, for $0 \le i \le 15$.
5. Swap S_i and $S_{i\oplus 2}$ for $i \in \{16,17,20,21,24,25,28,29\}$ (We call this swap function $SWAP_2$).
6. Add S_i into $S_{i\oplus 16}$, for $0 \le i \le 15$.
7. Rotate S_i to the left by eleven bits, for $0 \le i \le 15$.
8. Swap S_i and $S_{i\oplus 4}$, for $i \in \{0,1,2,3,8,9,10,11\}$ (We call this swap function $SWAP_3$).
9. XOR $S_{i\oplus 16}$ into S_i, for $0 \le i \le 15$.
10. Swap S_i and $S_{i\oplus 1}$, for $i \in \{16,18,20,22,24,26,28,30\}$ (We call this swap function $SWAP_4$).

Having processed all message blocks, a fixed transformation is applied to the final internal state to extract the hash value as follows. First, the last state word S_{31} is XORed with integer 1 and then the above permutation is applied $10 \times r$ times to the resulting internal state. Finally, the internal state is truncated to produce the message digest of desired hash length. Refer to [2] for the full specification.

3.1 CubeHash Compression Function

Compression function of CubeHash-r/b that we denoted by COMP-r, gives a state of 1024 bits (128-byte) as input and then applies CubeHash round function, ROUND, r times to this state and output a new state of 1024 bits (128-byte).

COMP-8 and COMP-11: COMP-8 is the compression function of CubeHash-8/b and COMP-15 is the compression function of CubeHash-15/b.

4. Rotational Analysis of CubeHash Swap Functions

As mentioned, in order to extend the application of rotational analysis to CubeHash compression function we have to find the probabilities that four swap functions ($SWAP_1, SWAP_2, SWAP_3$ and $SWAP_4$) in CubeHash round function (ROUND) preserve the rotational property for any input pair. In this section we compute the probabilities in lemma 4 to lemma 7.

Lemma 4 ($SWAP_1$): Suppose $SWAP_1$ is the swap function used in step 3 of CubeHash round function. We choose a random X, rotate it to the left by 1 bit and produce a pair of rotational input, $(X, X <<< 1)$. For this pair, $SWAP_1$ preserves the rotational property with probability 1. In other words:

$$\Pr[SWAP_1(X <<< 1) = SWAP_1(X) <<< 1] = 1.$$

Lemma 5 ($SWAP_2$): Suppose $SWAP_2$ is the swap function used in step 5 of CubeHash round function. We choose a random X, rotate it to the left by 1 bit and produce a pair of rotational input, $(X, X <<< 1)$. For this pair, $SWAP_2$ preserves the rotational property with probability of 2^{-6}. In other words:

$$\Pr[SWAP_2(X <<< 1) = SWAP_2(X) <<< 1] = 2^{-6}.$$

Lemma 6 ($SWAP_3$): Suppose $SWAP_3$ is the swap function used in step 8 of CubeHash round function. We choose a random X, rotate it to the left by 1 bit and produce a pair of rotational input, $(X, X <<< 1)$. For this pair, $SWAP_3$ preserves the rotational property with probability of 2^{-2}. In other words:

$$\Pr[SWAP_3(X <<< 1) = SWAP_3(X) <<< 1] = 2^{-2}.$$

Lemma 7 ($SWAP_4$): Suppose $SWAP_4$ is the swap function used in step 10 of CubeHash round function. We choose a random X, rotate it to the left by 1 bit and produce a pair of rotational input, $(X, X <<< 1)$. For this pair, $SWAP_4$ preserves the rotational property with probability of 2^{-14}. In other words:

$$\Pr[SWAP_4(X <<< 1) = SWAP_4(X) <<< 1] = 2^{-14}.$$

Proof: The above lemmas will be proved in the Appendix A.

5. Rotational analysis of CubeHash-8/b and CubeHash-15/b compression functions

In this section we applied the rotational analysis technique to CubeHash and find a new distinguisher for CubeHash-8/b compression function (COMP-8) and CubeHash-15/b compression function (COMP-15) that distinguish these functions from a random function. For this purpose we use the results of lemma 1 to lemma 7.

Now, we consider the COMP-8 function that using three simple operations of modular addition, rotation and XOR, and four swap functions of $SWAP_1, SWAP_2, SWAP_3$, and $SWAP_4$. Suppose we want to find the probability that COMP-8 preserves the rotational property. In other words if $(X, X <<< 1)$ be a pair of rotational input, we have to compute the probability that (Comp-8(X), Comp-8(X)<<<1) is a pair of rotational output (the probability that Comp-8(X<<<1) = Comp-8(X)<<<1).

By lemma 2 and lemma 3, rotations and XORs preserve the rotational property with probability 1. So the desired probability is depending on the additions and swap functions. In the rotational analysis of COMP-8 if we consider left rotation amount fixed to 1, by lemma 1 we can find the probability that one addition (on the 32-bit words) in COMP-8 preserve the rotational property. This probability is:

$$\Pr[(x + y) <<< 1 = x <<< 1 + y <<< 1] = \frac{1}{4}(1 + 2^{1-32} + 2^{-1} + 2^{-32}) > 2^{-1.416}$$

Each round of COMP-8 has 32 additions on the 32-bit words. Hence the probability that in the one round of COMP-8 the rotational property is preserved by all addition (Pr_{ADD}) will be:

$$Pr_{ADD} = 2^{(-1.416)^{32}} > 2^{-45.312}$$

On the other hand in the each round of COMP-8 four swap functions ($SWAP_1, SWAP_2, SWAP_3$, and $SWAP_4$) is used too. For these functions we compute the probabilities that they preserve the rotational property. These probabilities are respectively:

$$Pr_{SWAP_1} = 1,$$
$$Pr_{SWAP_2} = 2^{-6},$$
$$Pr_{SWAP_3} = 2^{-2},$$
$$Pr_{SWAP_4} = 2^{-14}.$$

According to section 2 the probability (Pr_{ROUND}) that one round of COMP-8 using 32 additions on the 32-bit words, XORs, rotations, and swap functions $SWAP_1, SWAP_2$, $SWAP_3$, and $SWAP_4$ preserve the rotational property is:

$$Pr_{ROUND} = (Pr_{ADD})^{32} \times (Pr_{SWAP_1}) \times (Pr_{SWAP_2}) \times (Pr_{SWAP_3}) \times (Pr_{SWAP_4}) > 2^{-67.312}.$$

Finally the COMP-8 function has 8 rounds. Hence the probability (Pr_{COMP-8}) that this function preserve the rotational property for a pair of rotational input such as $(X, X <<< 1)$ and produce a pair of rotational output such as $(COMP-8(X), COMP-8(X) <<< 1)$ will be:

$$Pr_{COMP-8} = (Pr_{ROUND})^8 > 2^{-538.496} >>> 2^{-1024}$$

The Pr_{COMP-8} is very greater than 2^{-1024} (the probability that we can distinguish the CubeHash compression function when it is indistinguishable from a random function) and allows distinguishing 8-round CubeHash compression function using about 2^{539} call of it. The distinguisher of COMP-8 based on rotational analysis technique is working this way:

1- He chooses a random input such as X, computes $X <<< 1$ and produces a pair of the rotational input, $(X, X <<< 1)$.

2- He computes COMP-8 (X), COMP-8($X <<< 1$), and COMP-8(X)<<<1 .

3- He verifies whether

COMP-8(X<<<1)=COMP-8(X)<<<1

or not.

4- If COMP-8(X<<<1)=COMP-8(X)<<<1 , the distinguisher can produce a pair of the rotational output, $(COMP-8(X), COMP-8(X) <<< 1)$, and distinguishes the COMP-8, otherwise go to 1.

COMP-15 Distinguisher: Similar to whatever we said about distinguishing of COMP-8, we can construct a distinguisher based on rotational analysis for 15-round CubeHash compression function (COMP-15) and distinguish it from a random function.
Note that the only difference between COMP-8 and COMP-15 is the number of their rounds. COMP-8 using 8 times of the CubeHash round function (ROUND) while COMP-15 using 15 times of it. Hence the probability ($Pr_{COMP-15}$) that this function preserve the rotational property for a pair of rotational input such as $(X, X <<< 1)$ and produce a pair of rotational output such as $(COMP-15(X), COMP-15(X) <<< 1)$ will be:

$$Pr_{COMP-15} = (Pr_{ROUND})^{15} > 2^{-1009.68} >> 2^{-1024}$$

The $Pr_{COMP-15}$ is greater than 2^{-1024} (the probability that we can distinguish the CubeHash compression function when it is indistinguishable from a random function) and allows distinguishing 15-round CubeHash compression function using about 2^{1010} call of it. The distinguisher of COMP-15 based on rotational analysis technique, too and is working as the COMP-8 distinguisher.

6. Conclusion

In this paper we find a new distinguisher based on rotational analysis technique for CubeHash compression function and distinguish the compression functions of CubeHash-8/b and CubeHash-15/b with probability greater than $2^{-538.5}$ and $2^{-1009.7}$, respectively. If in the CubeHash-8/b compression function we set b=1, indeed we distinguish the compression function of CubeHash-8/1, the first version of CubeHash that suggested to SHA-3 competition, from a random function. Also if we set b=32 in CubeHash-15/b, then we distinguish a reduced version of the compression function of CubeHash-16/32 (a tweaked version of CubeHash-8/1) only with one round less, from a random function. The distinguisher of CubeHash-15/b compression function presented in this paper is the first distinguisher that distinguish the CubeHash compression function with more than 14 rounds from a random function.

References

[1] T. Ashur, O. Dunkelman, "Linear Analysis of Reduced-Round CubeHash", Cryptology ePrint Archive, Report 2010/535 (2010).
[2] D.J. Bernstein, "Cubehash", Submission to NIST, Round 2 (2009).
[3] D.J. Bernstein, "CubeHash parameter tweak: 16 times faster".
[4] E. Biham, A. Shamir, "Differential Cryptanalysis of DES-like Cryptosystems", J. Cryptology, 4(1):3-72 (1991).
[5] D. Gligoroski, V. Klima, S. J. Knapskog, M. El-Hadedy, J., S. Amundsen, F. Mj lsnes, "Cryptographic Hash Function BLUE MIDNIGHT WISH", Submission to NIST (Round 2), (2009). Available at http://people.item.ntnu.no/~danilog/Hash/BMW-SecondRound/Supporting_Documentation/BlueMidnightWishDocumentation.pdf.
[6] L. R. Knudsen, K. Matusiewicz, S. S. Thomsen, "Observations on the Shabal keyed permutation", OFFICIAL COMMENT, (2009). Available at http://www.mat.dtu.dk/people/S.Thomsen/shabal/shabal.pdf.
[7] D. Khovratovich, I. Nikolic, "Rotational Cryptanalysis of ARX", Fast Sotftware Encryption (FSE 2010), Springer (2010).
[8] G. Leurent, C. Bouillaguet, P.-A, Fouque, " SIMD Is a Message Digest", Submission to NIST (Round 2), (2009).
[9] M. Matsui, "Linear Cryptoanalysis Method for DES Cipher", T. Helleseth, editor, EUROCRYPT, volume 765 of Lecture Notes in Computer Science, pages 386-397. Springer, (1993).
[10] National Institute of Standards and Technology, Announcing Request for Candidate Algorithm Nominations for a New Cryptographic Hash Algorithm (SHA-3) Family, Federal Register Notice (November 2007), available online at: http://csrc.nist.gov
[11] I. Nikolić, J. Pieprzyk, P. Sokołowski, Ron. Steinfeld, "Rotational Cryptanalysis of (Modified) Versions of BMW and SIMD", Available online, (2010).
[12] ECRYPT II, The SHA-3 Zoo, CubeHash Profile, Available at: http://ehash.iaik.tugraz.at/wiki/CubeHash
[13] X. Wang, H. Yu, "How to Break MD5 and Other Hash Functions", Advances in Cryptology, EUROCRYPT 2005, LNCS, Springer-Verlag, (2005).
[14] X. Wang, H. Yu, Y.L. Yin, "Efficient Collision Search Attacks on SHA-0", Advances in Cryptology, Crypto 2005, LNCS 3621, Pages 1-16, Springer, (2005).
[15] X. Wang, Y.L. Yin, H. Yu, "Finding Collisions in the Full SHA-1", Advances in Cryptology, Crypto 2005, LNCS 3621, Pages 17- 36, Springer, (2005).

Javad Alizadeh received the Bachelor's degree in Applied Mathematics with the honor degree from IHU, Tehran, Iran in 2007 and Master's degree in Telecommunication in the field of

Cryptography with the honor degree from IHU, Tehran, Iran in 2010. He was chosen as a superior researcher student of IHU, in 2010. Currently, he is a researcher in the field of cryptography and teacher assistant (TA) at the faculty and research center of communication and information technology, IHU, Tehran, Iran. His research interest includes: Cryptography, Cryptanalysis, Information Systems Security, Mathematics of Cryptography. He is a member of ISC.

Abdolrasoul Mirghadri received the B.Sc., M.Sc. and PHD degrees in Mathematical Statistics, from the faculty of Science, Shiraz University in 1986, 1989 and 2001, respectively. He is an assistant professor at the faculty and research center of communication and information technology, IHU, Tehran, Iran since 1989. His research interest includes: Cryptography, Cryptanalysis, Statistics and Stochastic Processes. He is a member of ISC, ISS and IMS.

Appendix

A. Proofs of Lemma 4 to Lemma 7

Proof of lemma 4 ($SWAP_1$): We would like to prove that for $SWAP_1$ function by any input such as X:

$$\Pr[SWAP_1(X <<< 1) = SWAP_1(X) <<< 1] = 1.$$

For the proof suppose that $S = (S_0, S_1, ..., S_{31})$ be the state of CubeHash round function. By attention to definition of $SWAP_1$ in the step 3 of the round function we have

$SWAP_1$:
Swap S_i and $S_{i\oplus 8}$, for $0 \le i \le 7$.

In fact the $SWAP_1$ function operates on the left half of the S. Consider this half as

$$X = S_0 \| S_1 \| S_2 \| S_3 \| S_4 \| S_5 \| S_6 \| S_7 \| S_8 \| S_9 \| S_{10} \| S_{11} \| S_{12} \| S_{13} \| S_{14} \| S_{15}$$

where the means of the notation $\|$ is concatenating. Using the definition of X and $SWAP_1$ function we have

$$SWAP_1(X) = S_8 \| S_9 \| S_{10} \| S_{11} \| S_{12} \| S_{13} \| S_{14} \| S_{15} \| S_0 \| S_1 \| S_2 \| S_3 \| S_4 \| S_5 \| S_6 \| S_7.$$

Without loss of generality and for simplicity consider

$$X_1 = S_0 \| S_1 \| S_2 \| S_3 \| S_4 \| S_5 \| S_6 \| S_7,$$
$$X_2 = S_8 \| S_9 \| S_{10} \| S_{11} \| S_{12} \| S_{13} \| S_{14} \| S_{15},$$

where X_1 and X_2 have 512-bit length. Consider the bit representation of them as

$$X_1 = x_1^0 x_1^1 x_1^2 ..... x_1^{254} x_1^{255},$$
$$X_2 = x_2^0 x_2^1 x_2^2 ..... x_2^{254} x_2^{255}.$$

By rewriting the input of $SWAP_1$ function as $X = X_1 \| X_2$, we have $SWAP_1(X) = X_2 \| X_1$. Now we show that

$$\Pr[SWAP_1(X <<< 1) = SWAP_1(X) <<< 1] = 1.$$

In order to show it we compute

$$X <<< 1 = x_1^1 x_1^2 ..... x_1^{254} x_1^{255} x_2^0 \| x_2^1 x_2^2 ..... x_2^{254} x_2^{255} x_1^0,$$

$$SWAP_1(X <<< 1) = x_2^1 x_2^2 ..... x_2^{254} x_2^{255} x_1^0 \| x_1^1 x_1^2 ..... x_1^{254} x_1^{255} x_2^0, \quad (4.1)$$

and

$$SWAP_1(X) <<< 1 = x_2^1 x_2^2 ..... x_2^{254} x_2^{255} x_1^0 \| x_1^1 x_1^2 ..... x_1^{254} x_1^{255} x_2^0 \quad (4.2)$$

By attention to (4.1) and (4.2) we see that $SWAP_1(X <<< 1) = SWAP_1(X) <<< 1$ and for this equality no condition is need. Consequently, we showed that

$$\Pr[SWAP_1(X <<< 1) = SWAP_1(X) <<< 1] = 1.$$

■

Proof of lemma 5 ($SWAP_2$): We would like to prove that for $SWAP_2$ function by any input such as X:

$$\Pr[SWAP_2(X <<< 1) = SWAP_2(X) <<< 1] = 2^{-6}.$$

For the proof suppose that $S = (S_0, S_1, ..., S_{31})$ be the state of CubeHash round function. By attention to definition of $SWAP_2$ in the step 5 of the round function we have

$SWAP_2$:
Swap S_i and $S_{i\oplus 2}$, for $i \in \{16,17,20,21,24,25,28,29\}$.

In fact the $SWAP_2$ function operates on the Right half of the S. Consider this half as

$$X = S_{16} \square S_{17} \square S_{18} \square S_{19} \square S_{20} \square S_{21} \square S_{22} \square S_{23} \square S_{24} \square S_{25} \square S_{26} \square S_{27} \square S_{28} \square S_{29} \square S_{30} \square S_{31}$$

Using the definition of X and $SWAP_2$ function we have

$$SWAP_2(X) = S_{18} \square S_{19} \square S_{16} \square S_{17} \square S_{22} \square S_{23} \square S_{20} \square S_{21} \square S_{26} \square S_{27} \square S_{24} \square S_{25} \square S_{30} \square S_{31} \square S_{28} \square S_{29}$$

Without loss of generality and for simplicity consider

$$X_1 = S_{16} \square S_{17}, \quad X_5 = S_{24} \square S_{25},$$
$$X_2 = S_{18} \square S_{19}, \quad X_6 = S_{26} \square S_{27},$$
$$X_3 = S_{20} \square S_{21}, \quad X_7 = S_{28} \square S_{29},$$
$$X_4 = S_{22} \square S_{23}, \quad X_8 = S_{30} \square S_{31},$$

where $X_i, 1 \le i \le 8,$ have 64-bit length. Consider the bit representation of them as

$$X_i = x_i^0 x_i^1 x_i^2 .....x_i^{62} x_i^{63}$$

By rewriting the input of $SWAP_1$ function as

$$X = X_1 \square X_2 \square X_3 \square X_4 \square X_5 \square X_6 \square X_7 \square X_8,$$

we have

$$SWAP_2(X) = X_2 \square X_1 \square X_4 \square X_3 \square X_6 \square X_5 \square X_8 \square X_7.$$

Now we show that

$$\Pr[SWAP_2(X <<< 1) = SWAP_2(X) <<< 1] = 2^{-6}$$

In order to show it we compute

$$X <<< 1 = x_1^1 x_1^2 ... x_1^{63} x_2^0 \square x_2^1 x_2^2 ... x_2^{63} x_3^0 \square x_3^1 x_3^2 ... x_3^{63} x_4^0 \square x_4^1 x_4^2 ... x_4^{63} x_5^0 \square x_5^1 x_5^2 ... x_5^{63} x_6^0 \square x_6^1 x_6^2 ... x_6^{63} x_7^0 \square x_7^1 x_7^2 ... x_7^{63} x_8^0 \square x_8^1 x_8^2 ... x_8^{63} x_1^0$$

$$SWAP_2(X <<< 1) = x_2^1 x_2^2 ... x_2^{63} x_3^0 \square x_1^1 x_1^2 ... x_1^{63} x_2^0 \square x_4^1 x_4^2 ... x_4^{63} x_5^0 \square x_3^1 x_3^2 ... x_3^{63} x_4^0 \square x_6^1 x_6^2 ... x_6^{63} x_7^0 \square x_5^1 x_5^2 ... x_5^{63} x_6^0 \square x_8^1 x_8^2 ... x_8^{63} x_1^0 \square x_7^1 x_7^2 ... x_7^{63} x_8^0 \quad (5.1)$$

and

$$SWAP_2(X) <<< 1 = x_2^1 x_2^2 ... x_2^{63} x_1^0 \square x_1^1 x_1^2 ... x_1^{63} x_4^0 \square x_4^1 x_4^2 ... x_4^{63} x_3^0 \square x_3^1 x_3^2 ... x_3^{63} x_6^0 \square x_6^1 x_6^2 ... x_6^{63} x_5^0 \square x_5^1 x_5^2 ... x_5^{63} x_8^0 \square x_8^1 x_8^2 ... x_8^{63} x_7^0 \square x_7^1 x_7^2 ... x_7^{63} x_2^0 \quad (5.2)$$

By attention to (5.1) and (5.2) we see that for the equality of $SWAP_2(X <<< 1) = SWAP_2(X) <<< 1$ we need the following conditions on some bits of X:

$$x_1^0 = x_3^0 = x_5^0 = x_7^0$$

and

$$x_2^0 = x_4^0 = x_6^0 = x_8^0$$

Each of these conditions satisfies with probability of $\left(\frac{1}{2}\right)^3$. Consequently, we showed that

$$\Pr[SWAP_2(X <<< 1) = SWAP_2(X) <<< 1] = 2^{-6}.$$

■

Proof of lemma 6 ($SWAP_3$): We would like to prove that for $SWAP_3$ function by any input such as X:

$$\Pr[SWAP_3(X <<< 1) = SWAP_3(X) <<< 1] = 2^{-2}.$$

For the proof suppose that $S = (S_0, S_1, ..., S_{31})$ be the state of CubeHash round function. By attention to definition of $SWAP_3$ in the step 8 of the round function we have

$SWAP_3$:
Swap S_i and $S_{i \oplus 4}$, for $i \in \{0,1,2,3,8,9,10,11\}$

In fact the $SWAP_3$ function operates on the left half of the S. Consider this half as

$$X = S_0 \square S_1 \square S_2 \square S_3 \square S_4 \square S_5 \square S_6 \square S_7 \square S_8 \square S_9 \square S_{10} \square S_{11} \square S_{12} \square S_{13} \square S_{14} \square S_{15}$$

Using the definition of X and $SWAP_3$ function we have

$$SWAP_3(X) = S_4 \square S_5 \square S_6 \square S_7 \square S_0 \square S_1 \square S_2 \square S_3 \square S_{12} \square S_{13} \square S_{14} \square S_{15} \square S_8 \square S_9 \square S_{10} \square S_{11}$$

Without loss of generality and for simplicity consider

$$X_1 = S_0 \Box S_1 \Box S_2 \Box S_3,$$
$$X_2 = S_4 \Box S_5 \Box S_6 \Box S_7,$$
$$X_3 = S_8 \Box S_9 \Box S_{10} \Box S_{11},$$
$$X_4 = S_{12} \Box S_{13} \Box S_{14} \Box S_{15},$$

where $X_i, 1 \le i \le 4$, have 128-bit length. Consider the bit representation of them as

$$X_i = x_i^0 x_i^1 x_i^2 .....x_i^{126} x_i^{127}$$

By rewriting the input of $SWAP_3$ function as

$$X = X_1 \Box X_2 \Box X_3 \Box X_4,$$

we have

$$SWAP_3(X) = X_2 \Box X_1 \Box X_4 \Box X_3.$$

Now we show that

$$\Pr[SWAP_3(X <<< 1) = SWAP_3(X) <<< 1] = 2^{-2}$$

In order to show it we compute

$$X <<< 1 = x_1^1 x_1^2 ... x_1^{127} x_2^0 \Box x_2^1 x_2^2 ... x_2^{127} x_3^0 \Box x_3^1 x_3^2 ... x_3^{127} x_4^0 \Box x_4^1 x_4^2 ... x_4^{127} x_1^0,$$

$$SWAP_3(X <<< 1) = x_2^1 x_2^2 ... x_2^{127} x_3^0 \Box x_1^1 x_1^2 ... x_1^{127} x_2^0 \Box x_4^1 x_4^2 ... x_4^{127} x_1^0 \Box x_3^1 x_3^2 ... x_3^{127} x_4^0 \quad (6.1)$$

and

$$SWAP_3(X) <<< 1 = x_2^1 x_2^2 ... x_2^{127} x_1^0 \Box x_1^1 x_1^2 ... x_1^{127} x_4^0 \Box x_4^1 x_4^2 ... x_4^{127} x_3^0 \Box x_3^1 x_3^2 ... x_3^{127} x_2^0 \quad (6.2)$$

By attention to (6.1) and (6.2) we see that for the equality of $SWAP_3(X <<< 1) = SWAP_3(X) <<< 1$ we need the following conditions on some bits of X:

$$x_1^0 = x_3^0$$

and

$$x_2^0 = x_4^0$$

Each of these conditions satisfies with probability of $\frac{1}{2}$. Consequently, we showed that

$$\Pr[SWAP_3(X <<< 1) = SWAP_3(X) <<< 1] = 2^{-2}.$$

■

Proof of lemma 7 ($SWAP_4$): We would like to prove that for $SWAP_4$ function by any input such as X:

$$\Pr[SWAP_4(X <<< 1) = SWAP_4(X) <<< 1] = 2^{-14}$$

For the proof suppose that $S = (S_0, S_1, ..., S_{31})$ be the state of CubeHash round function. By attention to definition of $SWAP_4$ in the step 10 of the round function we have

$SWAP_4$:
Swap S_i and $S_{i \oplus 1}$, for $i \in \{16, 18, 20, 22, 24, 26, 28, 30\}$

In fact the $SWAP_4$ function operates on the right half of the S. Consider this half as

$$X = S_{16} \Box S_{17} \Box S_{18} \Box S_{19} \Box S_{20} \Box S_{21} \Box S_{22} \Box S_{23} \Box S_{24} \Box S_{25} \Box S_{26} \Box S_{27} \Box S_{28} \Box S_{29} \Box S_{30} \Box S_{31}$$

Using the definition of X and $SWAP_3$ function we have

$$SWAP_4(X) = S_{17} \Box S_{16} \Box S_{19} \Box S_{18} \Box S_{21} \Box S_{20} \Box S_{23} \Box S_{22} \Box S_{25} \Box S_{24} \Box S_{27} \Box S_{26} \Box S_{29} \Box S_{28} \Box S_{31} \Box S_{30}$$

Without loss of generality and for simplicity and also similarity to the previous proofs, consider

$$X_1 = S_{16}, \quad X_9 = S_{24},$$
$$X_2 = S_{17}, \quad X_{10} = S_{25},$$
$$X_3 = S_{18}, \quad X_{11} = S_{26},$$
$$X_4 = S_{19}, \quad X_{12} = S_{27},$$
$$X_5 = S_{20}, \quad X_{13} = S_{28},$$
$$X_6 = S_{21}, \quad X_{14} = S_{29},$$
$$X_7 = S_{22}, \quad X_{15} = S_{30},$$
$$X_8 = S_{23}, \quad X_{16} = S_{31},$$

where $X_i, 1 \le i \le 16$, have 32-bit length. Consider the bit representation of them as

$$X_i = x_i^0 x_i^1 x_i^2 .....x_i^{30} x_i^{31}$$

By rewriting the input of $SWAP_3$ function as

$$X = X_1 \| X_2 \| X_3 \| X_4 \| X_5 \| X_6 \| X_7 \| X_8 \| X_9 \| X_{10} \| X_{11} \| X_{12} \| X_{13} \| X_{14} \| X_{15} \| X_{16},$$

we have

$$SWAP_4(X) = X_2 \| X_1 \| X_4 \| X_3 \| X_6 \| X_5 \| X_8 \| X_7 \| X_{10} \| X_9 \| X_{12} \| X_{11} \| X_{14} \| X_{13} \| X_{16} \| X_{15}$$

Now we show that

$$\Pr[SWAP_4(X <<< 1) = SWAP_4(X) <<< 1] = 2^{-14}$$

In order to show it we compute

$$\begin{aligned} X <<< 1 = {} & x_1^1 x_1^2 \ldots x_1^{31} x_2^0 \| x_2^1 x_2^2 \ldots x_2^{31} x_3^0 \| x_3^1 x_3^2 \ldots x_3^{31} x_4^0 \\ & \| x_4^1 x_4^2 \ldots x_4^{31} x_5^0 \| x_5^1 x_5^2 \ldots x_5^{31} x_6^0 \| x_6^1 x_6^2 \ldots x_6^{31} x_7^0 \\ & \| x_7^1 x_7^2 \ldots x_7^{31} x_8^0 \| x_8^1 x_8^2 \ldots x_8^{31} x_9^0 \| x_9^1 x_9^2 \ldots x_9^{63} x_{10}^0 \\ & \| x_{10}^1 x_{10}^2 \ldots x_{10}^{63} x_{11}^0 \| x_{11}^1 x_{11}^2 \ldots x_{11}^{63} x_{12}^0 \| x_{12}^1 x_{12}^2 \ldots x_{12}^{63} x_{13}^0 \\ & \| x_{13}^1 x_{13}^2 \ldots x_{13}^{63} x_{14}^0 \| x_{14}^1 x_{14}^2 \ldots x_{14}^{63} x_{15}^0 \| x_{15}^1 x_{15}^2 \ldots x_{15}^{63} x_{16}^0 \\ & \| x_{16}^1 x_{16}^2 \ldots x_{16}^{63} x_1^0 \end{aligned}$$

$$\begin{aligned} SWAP_4(X <<< 1) = {} & x_2^1 x_2^2 \ldots x_2^{31} x_3^0 \| x_1^1 x_1^2 \ldots x_1^{31} x_2^0 \| x_4^1 x_4^2 \ldots x_4^{31} x_5^0 \\ & \| x_3^1 x_3^2 \ldots x_3^{31} x_4^0 \| x_6^1 x_6^2 \ldots x_6^{31} x_7^0 \| x_5^1 x_5^2 \ldots x_5^{31} x_6^0 \\ & \| x_8^1 x_8^2 \ldots x_8^{31} x_9^0 \| x_7^1 x_7^2 \ldots x_7^{31} x_8^0 \| x_{10}^1 x_{10}^2 \ldots x_{10}^{63} x_{11}^0 \\ & \| x_9^1 x_9^2 \ldots x_9^{63} x_{10}^0 \| x_{12}^1 x_{12}^2 \ldots x_{12}^{63} x_{13}^0 \| x_{11}^1 x_{11}^2 \ldots x_{11}^{63} x_{12}^0 \\ & \| x_{14}^1 x_{14}^2 \ldots x_{14}^{63} x_{15}^0 \| x_{13}^1 x_{13}^2 \ldots x_{13}^{63} x_{14}^0 \| x_{16}^1 x_{16}^2 \ldots x_{16}^{63} x_1^0 \\ & \| x_{15}^1 x_{15}^2 \ldots x_{15}^{63} x_{16}^0 \end{aligned} \quad (7.1)$$

and

$$\begin{aligned} SWAP_4(X) <<< 1 = {} & x_2^1 x_2^2 \ldots x_2^{31} x_1^0 \| x_1^1 x_1^2 \ldots x_1^{31} x_4^0 \| x_4^1 x_4^2 \ldots x_4^{31} x_3^0 \\ & \| x_3^1 x_3^2 \ldots x_3^{31} x_6^0 \| x_6^1 x_6^2 \ldots x_6^{31} x_5^0 \| x_5^1 x_5^2 \ldots x_5^{31} x_8^0 \\ & \| x_8^1 x_8^2 \ldots x_8^{31} x_7^0 \| x_7^1 x_7^2 \ldots x_7^{31} x_{10}^0 \| x_{10}^1 x_{10}^2 \ldots x_{10}^{63} x_9^0 \\ & \| x_9^1 x_9^2 \ldots x_9^{63} x_{12}^0 \| x_{12}^1 x_{12}^2 \ldots x_{12}^{63} x_{11}^0 \| x_{11}^1 x_{11}^2 \ldots x_{11}^{63} x_{14}^0 \\ & \| x_{14}^1 x_{14}^2 \ldots x_{14}^{63} x_{13}^0 \| x_{13}^1 x_{13}^2 \ldots x_{13}^{63} x_{16}^0 \| x_{16}^1 x_{16}^2 \ldots x_{16}^{63} x_{15}^0 \\ & \| x_{15}^1 x_{15}^2 \ldots x_{15}^{63} x_2^0 \end{aligned} \quad (7.2)$$

By attention to (7.1) and (7.2) we see that for the equality of $SWAP_4(X <<< 1) = SWAP_4(X) <<< 1$ we need the following conditions on some bits of X:

$$x_1^0 = x_3^0 = x_5^0 = x_7^0 = x_9^0 = x_{11}^0 = x_{13}^0 = x_{15}^0$$

and

$$x_2^0 = x_4^0 = x_6^0 = x_8^0 = x_{10}^0 = x_{12}^0 = x_{14}^0 = x_{16}^0$$

Each of these conditions satisfies with probability of $\left(\frac{1}{2}\right)^7$. Consequently, we showed that

$$\Pr[SWAP_4(X <<< 1) = SWAP_4(X) <<< 1] = 2^{-14}.$$

■