Double-A – A New Cryptographic Hash Function Its Security

Abdullah Al-Saleh[1,a], Mohammed Al-Ahmmad[1,b], Abdullah Issa[1,c], Adel Al-Foudery[1,d]

[1]Computer Science Department, College of Basic Education, Public Authority for Applied Education and Training, Kuwait City, Kuwait

[a]abdullah.n.sy@gmail.com, [b]malahmads@yahoo.com, [c]a.issa795@gmail.com, [d]adel_2004@yahoo.com

ISSN: 2231-8852

ABSTRACT

After the design, using Salsa20 cipher and the mode-of-operation of DOUBLE-A cryptographic hash function in “ITS DESIGN” paper, the analysis of it comes in this paper “ITS SECURITY”. This paper analyzes the security of DOUBLE-A cryptographic hash function and shows the durability of its sponge construction and Salsa20 cipher against most known attacks and security threats.

Keywords - Sponge function, salsa20, stream cipher, collision, security, hash function

1. Introduction

A cryptographic hash function used to verify the data integrity over the network peers and databases. Hash functions detect the data tampering and manipulating. Hash function digest should be one-way property such that going back through its operations should be impossible. That is why most secured databases using hash functions for sensitive data such as passwords and files. If the attacker succeeds in breaking the system and got the database, compromising these original sensitive data will be impossible. Furthermore, the attacker should not be able to forge another valid data that its digest is equals the valid stored one in the database. These are the characteristics of the ideal hash function called “Random function”. Random function should follow the basic three hash function security criteria (pre-image resistance, second pre-image resistance and collision resistance).

Hash functions follow vary constructions with using different ciphers. It depends on the designer consideration and thought.

DOUBLE-A hash function follows sponge construction – where the security of the digest is separated from its length - and uses Salsa20 cipher (Daniel J. Bernstein, 2005). It consists of 1600-bit state divided into bitrate 576-bit -where the hash function performs its operations and 1024-bit capacity which is a security parameter (Daniel J. Bernstein, 2005).

After the security analysis of DOUBLE-A, DOUBLE-A showed its durability as well as its high diffusion.

DOUBLE-A hash function resist to pre-image, second pre-image and collision. DOUBLE-A achieves the minimum claimed security level for 512-bit output size against the hash function security criteria and distinguishers.

This paper is the complimentary of “DOUBLE-A – A NEW CRYPTOGRAPHIC HASH FUNCTION - ITS DESIGN”. It shows the security analysis of it for most known attacks and distinguishers with proofs.

2. Paper Outline

Table 1. Terminologies

Term	**Definition**
#	Number of
C	Capacity parameter
S	Hash state
R	Bitrate
X	Input message
Z	Hash Digest
N	Output size
M	Message
F	Function – permutation [on *DOUBLE-A*]
R	The number of message bits processed per block permutation
S`	Suffix
IV	Initial Value
R_n	R_n is the result of $P_1 \oplus K$. (section 4.8)
Inner collision	collision after inserting the message - after the first XORing
State collision	Collisions after one permutation - Absorbing phase. It is easy to get state collision from inner collision.
Random Oracle (RO) \ Random sponge(RS)	Ideal hash function

This paper analyzes the security of DOUBLE-A hash function. This paper starts from Inner and state collision to differential cryptanalysis with diffusion test. Then shows the general idea if each attack, how the attack fits DOUBLE-A hash function and the resistance of DOUBLE-A against the attack. At the end of this paper, result box showing the minimum security level for all analyzed attacks. Finally, “the conclusion”. Table 1 presents the terminologies used.

3. Inner and State Collisions

3.1 Inner Collisions

Inner collisions are found after inserting the message while *DOUBLE-A* is using the exclusive-or operation.

3.2 State Collision

State collisions come after the hash function permutations (ARX). A random sponge is distinguishable from a random oracle by the presence or absence of inner collisions. Once the attacker gets inner collisions, producing a state collision is inevitable. It is not necessary to get inner collisions from state collisions. Many attacks rely on making collisions on these values. When c>2n, inner collisions are unlikely.

From the state collisions, attackers will find output collisions after the end of the squeezing phase. Suppose that there is a collision, the attacker does not know which part will be truncated at the output. So *DOUBLE-A* seems to be secured against output collisions.
Inner collision → State collision → Output collision

4. The Security Analysis

4.1. Length Extension Attack

Hash functions can be used as a Message Authentication Code to verify the data over the peers. Since MAC consists of H (Secret\\Data), the attacker is able to extend the message without knowing the secret itself, but the length of the secret should be known in order to make the extension equal to the same length of the secret. By that, the attacker still forges a valid MAC.
Original MAC:
H (Secret + M) = 6d5f807e23db210bc254a28be2d6759a0f5f5d99
Modified MAC
H (M + padding + Extension) = 0e41270260895979317fff3898ab85668953aaa2

Attackers extend the message by inserting a slightly different message to the original one while the server still sees it as a valid MAC. In the case of hash functions, the attacker tries to get to the internal states from the digest in order to know the length of the secret that has been prepended to the original message. After that the attacker inserts the appended message. Now, the server makes its calculations and sees it a valid MAC. Note that, if there is a collision between tow messages' digest, then MAC collision is inevitable. The best known countermeasure for length extension attack is HMAC construction. HMAC = (key ∥ H (key ∥ message)).

This attack is hard to succeed in *DOUBLE-A*, because in each operation, the state is updated and does not present any biases to any other state. Furthermore, going back to previous states is infeasible because here the attacker tries to make preimage attack which is protected by salsa20 cipher. Even though attacker succeeds in building the algorithm for computing the previous states of the output, the output is truncated and the attacker does not know the full digest and which part was truncated.

4.2. Collision Attacks

Two known inputs produce the same hash value. H(x) = H(y). The attacker is able to insert new messages inside the blinder and get the same x's digest value. Attackers use the mathematic birthday problem in probability theory which is for example, 25 students in the same class. How many students may share the same birthday? Since the year has only 365 days, collisions must exist [Fig.1]. This reduces the complexity to about half, so sponge collisions resistance is $2^{c/2}$. Due to the fact that *DOUBLE-A* has c=2n, *DOUBLE-A* has a collisions complexity 2^{C} which achieving the security requirement of ideal hash function.

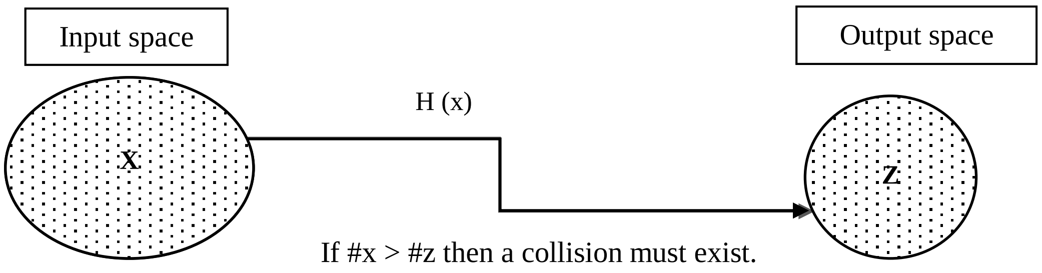

Fig. 1. Collisions

4.3. Pre-image Attacks

A known digest and an attacker tries to get the original message. Hash functions should be one-way functions, such that, an attacker is not able to go back to the original message through the intermediate chaining values. Pre-image can be obtained by binding output string to a state and subsequently finding a path to that state, then computing $T = f^{-1}(S)$ which leads to an inner state. Then, it goes back more to the original message. However, Salsa20 stream cipher uses modular addition (ARX) which the results do not have the exact reversal values that made the operations - to get the inputs.

Modular Addition operation makes the reversal $T^{-1}(f)$ operation is complicated, suppose x mod y = 1, the solution key here is that the attacker does not know the two values (x,y) that made the operations. Even if one of two values was known, an attacker is not able to detect the exact second value.

The total complexity of producing pre-image attacks on random sponges using brute force attack is $2^{c-r} + 2^{c/2}$. *DOUBLE-A* resistant to this type of attack due to the size of c = 2n and the Modular Addition operation used.

Briefly, a Pre-image attack is:

$H(M) \rightarrow$ 0a5d1f18c84b0c145f588a60121da7

Attacker tries to get M from the digest.

4.4. Second pre-image Attack by Kelsey-Schneier

Two inputs produce the same hash function. An attacker tries to find the second input which leads to the same digest with first input. Second pre-image can be obtained if the attacker has the ability to find a second path to the inner state with total complexity 2^{c}. If the attacker succeeds in finding collisions at intermediate chaining values, then the attacker is able to find an output collision in order to disclose the second message.

Since Double-A has c=2n, it will cost 2^{2c} as a general attack complexity. Attackers might use the brute force attack tool that uses the mathematic birthday paradox to establish the attack with complexity of 2^{C}.

Briefly, second Pre-image attack is:

$H(M_0) \rightarrow$ 0a5d1f18c84b0c145f588a60121da7

$H(M_1) \rightarrow$ 0a5d1f18c84b0c145f588a60121da7
Attackers try to find M_1.

In *DOUBLE-A* this generic attack is prevented by using 2n of capacity which makes it hard for the attacker to get the second Pre-image (J. Kelsey and B. Schneier, 2005).

4.5. MultiCollision Attacks by Joux

Cascading hash functions is a combination between two hashes ($H_1(M_1) \backslash\backslash H_2(M_1)$) or any form of this type (A. Joux, 2004).

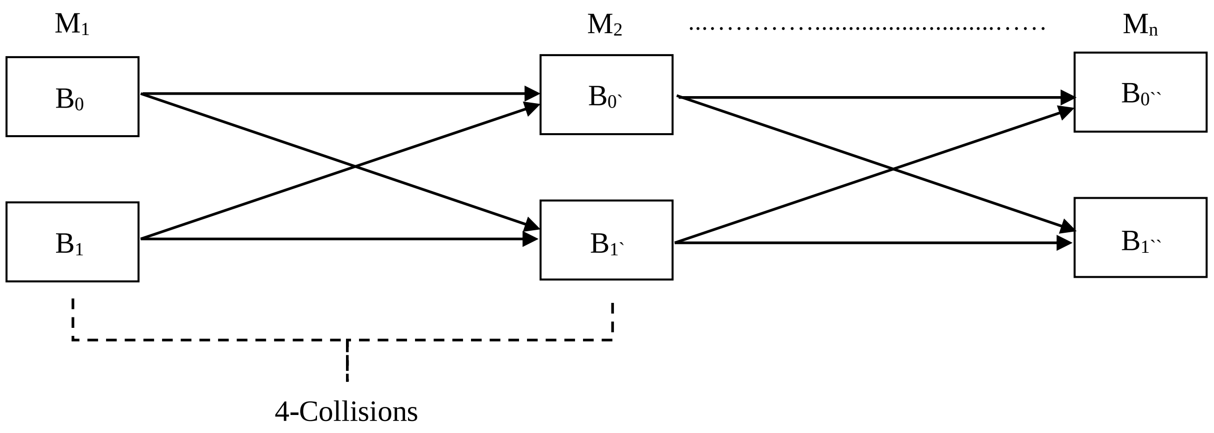

Fig. 2. Four-collision

Cascaded hashes have a complexity of $2^{c/2} + 2^{c/2}$ for collisions. A multicollision attack was first introduced by Joux in Joux Multicollision attack paper [6]. The original Joux paper studied cascaded hash functions and showing its security level. Joux proved that cascading hashes are not safer than pure one hash algorithm and the complexity of it remains the same for single hash algorithms against the basic security criteria, even though H_1 and H_2 are totally independent hash functions output. The four collision attacks obtained by making two calls of collision finding machine C and comparing all the messages block pairs to obtain four collisions [Fig.2]. Giving IV that produce two different blocks of a message such that $f(IV, B) = f(IV, B`)$. C should work properly for all chaining values or at least on a fixed proportion of them. It may use birthday attack or any specific attack based on a weakness of f (A. Joux, 2004).

Briefly, Multicollision attacks rely on finding state collisions which *DOUBLE-A* has the required resistance of it - as described above. Even incase an attacker succeeds on finding state collisions, the attacker does not know which part of the output will be truncated (A. Joux, 2004).

4.6. Herding Attack by Kelsey-Kohno

The attacker presents a digest Z and then for any message M attacker is able to find M2 such that H(M|M2) = Z. The idea of this attack is the production of diamond structure, which is a pre-computed data structure. This attack is parametrized by some positive integer K (Diamond width) (J. Kelsey and T. Kohno, 2006).

$$K = ((C - 5)/3) \quad (1)$$

For Alice repeatedly applies a collision-finding attack against a hash function to build a diamond structure, which is a pre-computed data structure reminiscent of a binary tree that takes 2^k intermediate hash states which iteratively converge to the same final digest value. Attackers exhaustively search a string S` such that P||S` collides with one of the diamond structure's intermediate states. This step requires trying 2^{c-k} possibilities for S`. Attackers expect to try about 2^{c-k} trial messages in order to find a linking message (J. Kelsey and T. Kohno, 2006).

DOUBLE-A [1024bit] Herding cryptanalysis

- $K = (1024-5)/3 = 339.6$
- Binary tree = $2^{339.6}$
- Attacker needs to try $2^{1024 - ((1024 - 5) / 3)}$ trial which is very big.

This attack is based on producing collisions at intermediate chaining values which is briefly another way to produce a second pre-image. For the diamond structure, an attacker needs collisions between two messages starting with different IVs. In *DOUBLE-A*, this attack is prevented by a strong inner and state collision resistance as above [section 2]. Furthermore, the IV is fixed for all messages.

4.7. Distinguishers

Distinguishers have become very popular against recent hash functions and are applicable to stream and block ciphers as it can take any form of cryptanalysis. Distinguishers can reveal the encryption method used some information about the encrypted message and refine the potential key space[1]. It studies the existence of relations between different outputs, or between inputs and outputs, which can be used by an attacker, for example to find (a part of) the input.

This section shows the resistance of *DOUBLE-A* and its cipher against all known distinguishers. *DOUBLE-A* should provide the minimum security level for the maximum output size with complexity of 2^{512}

4.8. Slide Attack

It is a cryptanalysis technique that attackers use it to uncover some parts of the key used at the cipher. The attacker makes queries M_i and receives H $(K||M_i)$, then tries to get some non-trivial information from the secret key or manages to forge another MAC (A. Biryukov and D. Wagner, 1999; A. Biryukov and D. Wagner, 2000).

The attack occurs in three steps:

1. Finding and detecting slid pairs of messages.
   - Good pair.
   - Bad pair; attackers should move to the next pair[discussed later on]
2. Recovering the internal state.
   - Exposing the original message.

[1] Set of all possible keys that can be used to produce a key. It is a very important attribute that determines the strength of any cryptosystem. It used to prevent the attacker from using a brute-force attack to find the key used to encipher a message.

3. Uncovering some part of the secret key.
   - In order to break cipher

Finding slid pair requires $2^{c/2}$ pair for detecting a good pair that satisfies the requirements for finding the key (from sub-keys to main key) (A. Biryukov and D. Wagner, 1999; A. Biryukov and D. Wagner, 2000).

Briefly attack explanation :(Fig.3)

After applying known plaintext-ciphertext attacks and finding the first sub-key, an attacker tries $P_0 \oplus K = R_1$ [see Table 1], R_1 should equals p_1. In addition, the last input for the last f should equal to C_0. If it is, the attacker is in a good situation. The attacker then succeeds on guessing the first part of main K [consist of sub-keys (k_0, k_1)] (which it is K_0) then calculating $f_3$sbox $\oplus$ $f_4$sbox = corresponding key (k_1). Now, an attacker is succeeded in breaking the cipher and can use the calculated main key to decrypt the message. If the pair does not satisfy the previous state, then this is considered a bad pair and the attacker should move to the next pair. In hash functions case, there is no key. This can be the message blocks. From the previous idea, attackers are able to get the state and subsequently can reach to the original message which means producing second pre-image. This attack requires known plaintext-ciphertext attack.

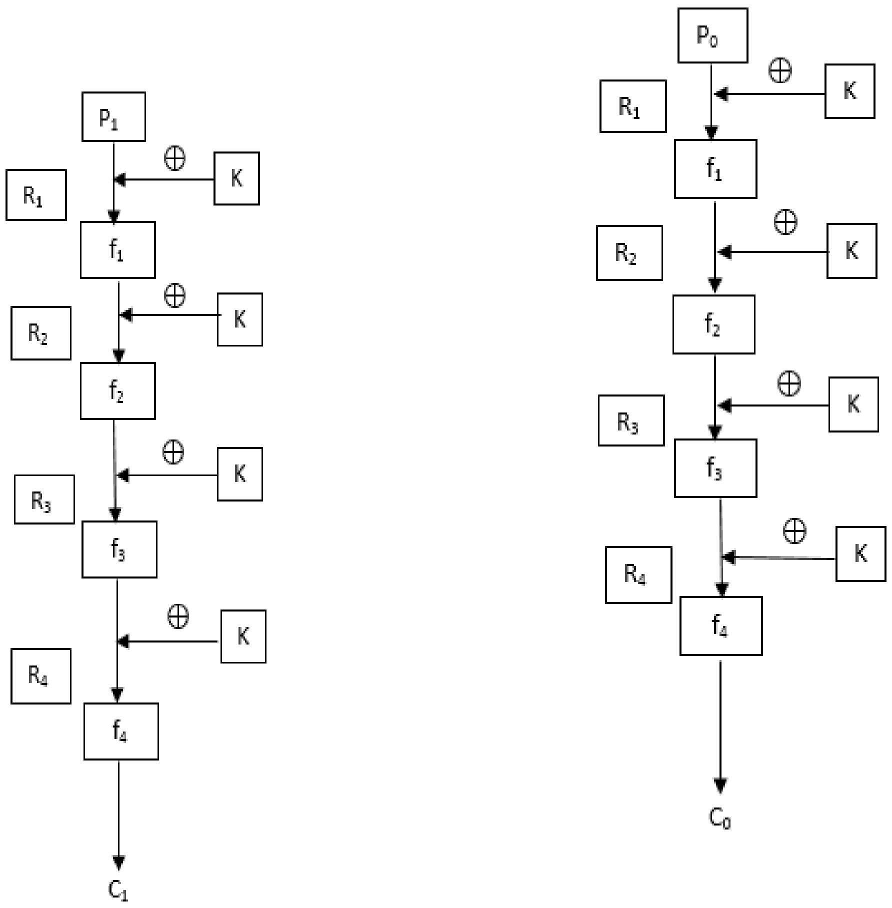

Fig. 3. Slide Attack on pair of plaintexts

This attack works in case the same key is used for all f. that means the same state. However, for *DOUBLE-A,* after each operation the state is updated. Furthermore, salsa20 relies on Modular Addition, which blocks the reversal operation. Also guessing all possible pairs for *DOUBLE-A* with using birthday paradox is impractical 2^{512} messages pairs!

4.9. Rotational Cryptanalysis

Cryptanalytic attack against algorithms that rely on three operations: modular addition, rotation and XOR. This attack relies on obtaining rotational pairs $(x,\overleftarrow{x})$. These pairs disclose information about rotated bits which exposes the state. If it happens, the attacker somehow will be able to compute $T=f^{-1}(S)$ in order to obtain Pre-image (D. Khovratovich and I. Nikolic, 2010).

Rotational cryptanalysis works like the following algorithm:

- Generate a random plaintext P and encrypt it on K;
- Compute P` and encrypt K`;
- Check whether (Ek(P), Ek`(P`)) is a rotational pair.
    - The plaintext P` is computed by the following rule:
        - P`i = P $\oplus$ di

Key can be translated into message block. Once the attacker discloses it, preimage will occur.

RX operations preserve rotational pairs with probability 1. Modular operation making it 0.375. Therefore, the rotational attack suffers from the number of Modular Additions operation in permutations. The rotated bits amount does not effect. On DOUBLE-A the probability of finding rotational pair is bounded away 1 due to **A**RX operations (D. Khovratovich and I. Nikolic, 2010).

4.10. Rebound Cryptanalysis

Attackers use rebound attack to find collisions on hash functions. It can be done by using techniques such as rotational or differential cryptanalysis. Attackers need to know the internals of the function attacked. Furthermore, rebound attacks combine advanced differential cryptanalysis techniques with "meet in the meddle strategy". By attacking the permutations (P) in two phases, attackers optimize the exploit of freedom degrees searching for inputs that conform a differential characteristic (F. Mendel, et al., 2009).

The basic idea of the attack is to observe a certain differential characteristic in a cipher; in a permutation or another type of primitive. Finding values fulfilling the characteristic is achieved by splitting the primitive E into three parts (subciphers) such that $E= E_{fw}$ o E_{in} o E_{bw}. E_{in} is called the inbound phase and E_{fw} and E_{bw} together are called the outbound phase. The basic attack strategy of rebound attacks is to find at least one starting point i.e. one paired values which will follow the characteristics of an inbound phase. After getting the values which will follow the characteristics of the inbound phase, the attacker finds one starting point and checks whether or not it satisfies the truncated differential trail. After that, the attacker moves these values forward and backward the states. Attackers precompute big State's table with differential characteristic (F. Mendel, et al., 2009).

4.10.1. Inbound phase (match in the middle)

At this phase, there is a mathematic system that uses equations to perform the purpose. Inbound phase covers the part of the differential characteristic that is difficult to satisfy in a probabilistic way i.e. the attacker tries to get differential characteristic. At this phase the attacker will carry out the following steps:

- Tries to find a large number of differential active bytes in input and output State (or layer (state after one single operation [A-R-X]) of intermediate chaining values).
- Compare these values with the attacker's Sates table looking for matches (Match in the middle).
- If attacker is able to obtain some active bytes, he may able to obtain full active state (more starting points).

Inbound phases may repeat several times until it finds a sufficient number of starting points that make it successful.

4.10.2. Outbound phase

The outbound phase tests these starting points in order to find paired values that satisfy the truncated differential path of the outbound phase. If it is satisfied, the matches of the inbound phase are computed in forward and backward directions through E_{fw} and $E_{b}w$ to obtain the desired collisions

DOUBLE-A has a 5x5byte state. The permutation is on 3x3 (Fig.5). In case the bitrate is fully active, the capacity is nonactive 1024bit which mean that the best possible number of starting points will be 2^{72} of 200 byte(1600bit full state) and hence, the active bits on the bitrate is carried on nonactive bits(not pure active bits).

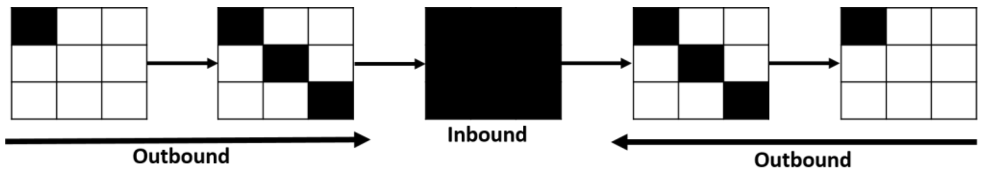

Fig. 4. Simple view of DOUBLE-A bitrate

4.11. Differential and Linear Cryptanalysis

Differential cryptanalysis is the study of how difference an input can affect result difference at the output. It aims to expose the key used. To establish this attack, the attacker should be able to obtain the ciphertext for some set of plaintext of his choosing and apply it on reduced round cipher.

In analyzing the *DOUBLE-A* security against Differentials and Linear cryptanalysis, *DOUBLE-A* shows a high security level.

4.11.1. Linear Cryptanalysis

Linear cryptanalysis aims at obtaining linear approximations related to the plaintext and the states of the ciphers prior to the last round and biases. The probability of ideal approximation should be bounded away from ½ because the number of zeros and ones is equally large - obtained from linear approximations (M. Matsui, 1993).

Let ‘I’ denotes “input”, ‘I-th’ denotes the input’s elements, ‘O’ denotes the output and ‘O-th’ denotes output’s elements.

$I=(I_1,I_2,I_m)$ $O = (O_1 , O_2 , O_m)$

$$I_1 \oplus I_3 \oplus I_4 \oplus O_2 \oplus O_4 \oplus O_5 = 0 \qquad (2)$$

This is an expression for an ideal approximation for the cipher. The goal of the linear cryptanalysis is to find linear approximations like the one above (2). The attacker takes *known* pairs of plaintext and ciphertext and checks them in order to see if they satisfy what the attacker is looking for. The attacker should obtain linear approximations relating plaintext to ciphertext with a probability of ½. The attacker has a large number of plaintext-ciphertext pairs and applies “known plaintext-ciphertext attack”. Then the attacker starts guessing the last rounds keys (corresponding key [$P(i_n) \oplus C(i_n) = K(k_n)$]) and decrypting the ciphertext to obtain the state previous to the last round (Discussed in slide attack). In hash functions case, there is no key used; this key can be translated into a message block. This might expose the internal states, exposing the message as mentioned above [slide attack - Fig.3] (M. Matsui, 1993).

The best linear bias found of 4 rounds of Salsa20 is $2^{-3.6758}$[13](bounded away 0.5). Thus, for hundreds of active bits[15], the attack costs a large amount of plaintext-ciphertext pairs. Furthermore, the high diffusion on DOUBLE-A makes linearization worse (M. Matsui, 1993).

4.11.2. Differential Cryptanalysis

The differential attack is defined as small differences in input states that will have small differences after the first computation step, the second step, etc. the attacker tries to get differentials between the inputs and traces the output’s state after the computations (Biham and A. Shamir, 1991).

Table 2. Test vectors for DOUBLE-A-512

DOUBLE-A ("The five boxing wizards jump quickly.")
96DA45779F8CBA4B0D5147A0610AA6814F4731F5929AA0163B6017EEB1BA AD77FEACD777A24B1F2D796B15965DC5216B0D5147A0610AA68DD6889BA 8BD8319AA
DOUBLE-A ("The five boxing wizards jump quickly")
F226D22B6918B3B73FC37A7627D60295C3E0F5A42E4046005EFC7F49675B806 13E0F3345C8EB5B47C8C4D7BCBE10EF8D3FC37A7627D60295F681FA212A27 38A0
DOUBLE-A (“ ”)
DB8ADF56E71612BC2BF88FA71AD71300B10A1704232D0CD12647F5D55FAA 08A01E6527E6BA749B16DB8ADF56E71612BC4B41ECED86930A12FC4CF182 0BD53266

The Salsa20 author (Daniel J. Bernstein, 2005) showed the cipher security against differentials. Salsa20 takes 16bytes input, 64bytes output and 32byte key; there are 2^{512}

choices of (n,n`,k), so there is no prior reason to believe that any of the choices have the 128-bit quantity n`⊕n and the 512-bit quantity Salsa20$_k$(n`)⊕Salsa20$_k$(n) both being "small.".

This is clear in Fig.4-5 and Table.2. There are 2^{512} pairs to perform the attack which achieves the claimed security level for 512-bit output of *DOUBLE-A.* Nicky Mouha and Bart Preneel proved that full Salsa20 with 128-bit key is secured of differentials (Nicky Mouha and Bart Preneel, 2013).

First Round (**Bolded** *italic* for Active value)

0x00000000	0x00000000	0x00000000	0x00000000
0x00000000	0x00000000	0x00000000	0x00000000
0x0000000***1***	0x00000000	0x00000000	0x00000000
0x00000000	0x00000000	0x00000000	0x00000000

Second Round (**Bolded** *italic* for Active value)

0x***8***00***40***00**1**	0x00000000	0x00000000	0x00000000
0x00000000	0x00000000	0x00000000	0x00000000
0x0000000***1***	0x00000000	0x00000000	0x00000000
0x0000***e***000	0x00000000	0x00000000	0x00000000

Third Round (**Bolded** *italic* for Active value)

0x***edc5e***0***a9***	0x0***2***0000***c***0	0x***381f83***0***c***	0x***3***0***4888dc***
0x00000000	0x00000000	0x00000000	0x00000000
0x0000000***1***	0x0000***6***000	0x***8***00***c***000***1***	0x00000000
0x0000***e***000	0x0***1c***00000	0x0***4***0000***d8***	0x0***12***00***f***00

Fifth Round (**Bolded** *italic* for Active value)

0x***39545d5e***	0x0***cc16***0***d8***	0x***3***0***1fb***0***3***0	0x**a**0***52***0***8dc***
0x***a24***0***cc8b***	0x***24e***0***12***0***c***	0x***2a***0***3***0***dc7***	0x***abeeb94e***
0x***39ea4***0***9b***	0x0000000***f***	0x***cf3bb828***	0x***1c2***0***5f6d***
0x***c6612ba5***	0x0***1c***0***6a***00	0x0***2***000***18***	0x**6745c36b**

Fourth Round (**Bolded** *italic* for Active value)

0x***f5eebb6a***	0x***79a3e194***	0x***52e3644f***	0x***28fc33dd***
0x***cbfe2c2e***	0x***a***0***ce9f57***	0x***fa23cf***02	0x***2f549d35***
0x***2b1af315***	0x***7af4976b***	0x***a1***00***a15f***	0x***86f42***0***f1***
0x***29***00***cc14***	0x***8dcbf124***	0x***90611242***	0x***61fdabbe***

Fig. 5. Five Rounds Salsa20/5

DOUBLE-A differential cryptanalysis test [Table2].

By testing two slightly different messages on DOUBLE-A [second and last row in Table2]. It clearly shows that there is a big difference between the two digests while the two messages differ only in the presence of a full-stop character at the end. The Salsa20 author's study showed that after four rounds of Salsa20, diffusion seems to be optimal (Daniel J.

Bernstein, 2005). Furthermore, there are 2^{512} pairs (n,n`) which achieve the claimed security for 512bit output.

M1-Diffusion Example of Salsa20/5

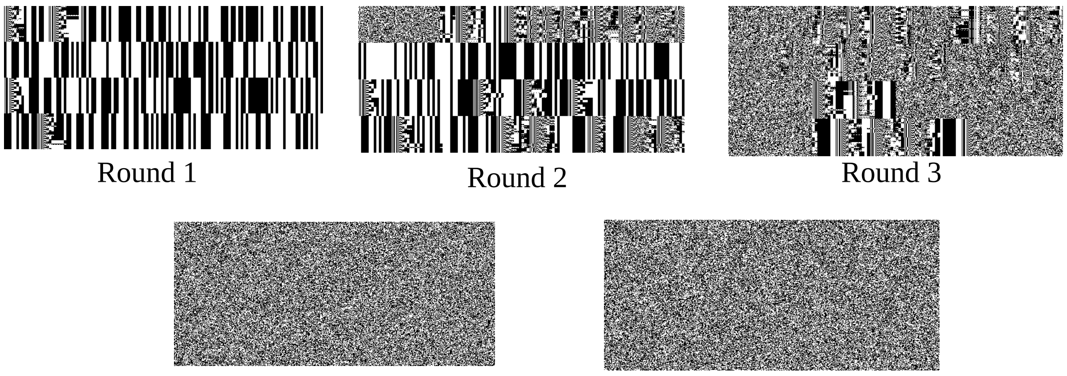

Fig. 6. Salsa20 Diffusion

5. Result Box

Table 3. DOUBLE-A Cryptanalysis Box

Cryptanalysis	Complexity
Length Extension	Intermediate chaining values collision resistance - Discussed at 4.1
Preimage	$2^{c/2}$
Second Preimage	2^{c}
Collision	$2^{c/2}$
Herding	$2^{1024 – ((1024 - 5) / 3)}$ messages trail - Discussed at 4.6
Slide cryptanalysis	2^{512} Message pairs - Discussed at 4.8
Rotational	0.375 Approximation for finding pair – Discussed at 4.9
Rebound	Discussed At 4.10
Linear	Linear Approximation 0.0782 - Discussed on 4.11.1
Differential	2^{512} pairs – Discussed in 4.11.2

6. Conclusion

After the analysis of DOUBLE-A hash function, it is clear that DOUBLE-A 512-bit output achieved the minimum claimed security level for all mentioned attacks. It has the pre-image resistance, second pre-image resistance and collision resistance as well as its high diffusion. As described above, the sponge construction of DOUBLE-A separated the security measurement from the digest's length. Therefore, there are no extra threats for small outputs length.

ACKNOWLEDGMENT

Special thanks to Dr. Mohammed Al-Ahmed for his notes, hopeful help and supervising. Thanks for anonymous people on the internet who has helped us.

REFERENCES

G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche. On the indifferentiability of the Sponge construction. In N. P. Smart, editor, EURO- CRYPT, volume 4965 of Lecture Notes in Computer Science, pages 181– 197. Springer, 2008.

Daniel J. Bernstein. Salsa20 design. Department of Mathematics, Statistics, and Computer Science. The University of Illinois at Chicago. Chicago, 2005.

J. Kelsey and B. Schneier. Second preimages on n-bit hash functions for much less than 2n work. In R. Cramer, editor, EUROCRYPT, volume 3494 of Lecture Notes in Computer Science, pages 474–490. Springer, 2005.

A. Joux. Multicollisions in iterated hash functions. application to cascaded constructions. In M. K. Franklin, editor, CRYPTO, volume 3152 of Lecture Notes in Computer Science, pages 306–316. Springer, 2004.

J. Kelsey and T. Kohno. Herding hash functions and the Nostradamus attack. In S.Vaudenay, editor, EUROCRYPT, volume 4004 of Lecture Notes in Computer Science, pages 183–200. Springer, 2006.

A. Biryukov and D. Wagner. Slide attacks. In L. R. Knudsen, editor, FSE, volume 1636 of Lecture Notes in Computer Science, pages 245–259. Springer, 1999.

A. Biryukov and D. Wagner. Advanced slide attacks. In B. Preneel, editor, EUROCRYPT, volume 1807 of Lecture Notes in Computer Science, pages 589–606. Springer, 2000.

D. Khovratovich and I. Nikolic. Rotational cryptanalysis of ARX. In Hong and Iwata, pages 333–346, 2010.

F. Mendel, C. Rechberger, M. Schla ̈ffer, and S. S. Thomsen. The Rebound attack: Cryptanalysis of reduced Whirlpool and Grøstl. In O. Dunkelman, editor, FSE, volume 5665 of Lecture Notes in Computer Science, pages 260–276. Springer, 2009

M. Matsui. Linear cryptoanalysis method for DES cipher. In T. Helleseth, editor, EUROCRYPT, volume 765 of Lecture Notes in Computer Science, pages 386–397. Springer, 1993.

Heuristic Search for Non-Linear Cryptanalytic Approximations. Juan M. E. Tapiador, Julio C. Hernandez-Castro, and John A. Clark

Biham and A. Shamir. Differential cryptanalysis of DES-like cryptosystems. J. Cryptology, 4(1):3–72, 1991.

Nicky Mouha and Bart Preneel. A Proof that the ARX Cipher Salsa20 is Secure against Differential Cryptanalysis. IACR Cryptology ePrint Archive, 2013:328, 2013.

Daniel J. Bernstein. Salsa20 security. Department of Mathematics, Statistics, and Computer Science (M/C 249) The University of Illinois at Chicago Chicago, 2005.